CASTOR

Building Trust
in the Computing Continuum

D2.1

Operational Landscape, Requirements and Reference Architecture - Initial version

Project number:	101167904
Project acronym:	**CASTOR**
Project title:	Continuum of Trust: Increased Path Agility and Trustworthy Device and Service Provisioning
Project Start Date:	1st October, 2024
Duration:	36 months
Programme:	HORIZON-CL3-2023-CS-01
Deliverable Type:	Report
Reference Number:	HORIZON-CL3-2021-CS-01-101167904/ D2.1 / v1.0
Workpackage:	WP2
Due Date:	30th September, 2025
Actual Submission Date:	1st December, 2025
Responsible Organisation:	UBITECH
Editor:	Nikos Fotos, Thanassis Giannetsos
Dissemination Level:	Public
Revision:	1.0
Abstract:	This deliverable defines the CASTOR technical requirements, alongside the specification of the conceptual reference architecture, the functional components, and interfaces between them. It also provides an analysis and point of reference for CASTOR in relation to the use cases and reference scenarios including an analysis on the interfacing between the use case application software stack and the CASTOR framework as well as preliminary Key Performance Indicators that need to be considered for engraining trust-enabled routing policies, while also respecting the operational profile of the envisioned use case scenarios. The purpose of this document is to define the parameters for the rest of the project and provide the necessary input for the design and implementation of all security enablers and models towards trust-aware, dynamic traffic engineering provisioning.

Funded by EU's Horizon Europe programme under Grant Agreement number 101167904 (CASTOR). Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI).

Funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee.

Copyright Notice

Project Funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable	R*	
	Dissemination Level	
PU	Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)	X
SEN	Sensitive, limited under the conditions of the Grant Agreement	
Classified R-UE/ EU-R	EU RESTRICTED under the Commission Decision No2015/ 444.	
Classified C-UE/ EU-C	EU CONFIDENTIAL under the Commission Decision No2015/ 444	
Classified S-UE/ EU-S	EU SECRET under the Commission Decision No2015/ 444	

* R: Document, report (excluding the periodic and final reports)
DEM: Demonstrator, pilot, prototype, plan designs
DEC: Websites, patents filing, press & media actions, videos, etc.
DATA: Data sets, microdata, etc.
DMP: Data management plan
ETHICS: Deliverables related to ethics issues
SECURITY: Deliverables related to security issues
OTHER: Software, technical diagram, algorithms, models, etc.

Editor

Nikos Fotos, Thanassis Giannetsos (UBITECH)

Contributors (ordered according to beneficiary numbers)

Nikos Fotos, Sofianna Menesidou, Panagiotis Banavos, Thanassis Giannetsos (UBITECH)
Iasonas Sakellariou, Stelios Kazazis, Symeon Tsintzos (QUBITECH)
Fabian Schwarz, Meni Orenbach (NVIDIA)
Yalan Wang, Liqun Chen (SURREY)
Alexandru Coleș, Ioan Constantin (ORO)
Vlad Chiriac, Ciprian-Romeo Comsa (TUIASI)
Riccardo Orizio, Michael McElligott, Stelios Basayiannis (COLLINS)
Vangelis Kosmatos, Panagiotis Pantazopoulos (ICCS)
Kostas Latanis (SUITE5)
Christos Dalamagkas, Ioannis Boukas, Evangelos Syrmos (K3Y)
Vasiliki Lamprousi, Aristi Galani, Sokratis Barmpounakis (WINGS)
Pablo Martinez, Antonio Skarmeta (UMU)
Alexandros Fakis, Kostas Maliatsos (FERON)
Gergely Kovacs, Andras Edelmayer (COMMSIGNIA)
Anuj Pathania, Andy Pimentel (UvA)
Jamie Pont, Budi Arief, Theo Dimitrakos (UKENT)

Disclaimer

Executive Summary

CASTOR extends the state of the art in the **secure data transmission by engraining trust metrics as part of the Traffic Engineering (TE) process**, which is the project's central vision towards establishing "trustful service-graph-chains". Routing decisions determine how information flows across the Internet and shape the performance and quality of globally-deployed services. In traditional networking, routing affects how data packets travel from their source to their destination across topologies composed of interconnected network elements (i.e., routers). In IP networks, data is segmented into packets labelled with headers that include the destination IP address. Routers inspect each packet's address and select the next hop using continuously updated routing tables. This process scales to thousands or even millions of devices, ultimately enabling the global-scale delivery of Internet data flow.

Each selected path can be shaped by diverse criteria (such as hop count, bandwidth, and delay) establishing paths of the shortest distance, or achieving the fastest or the least congested Internet route to the destination. However, **security mechanisms for converging such high-degree of network efficiency with the establishment of secure data channels is only currently considered as an add-on and not inherently engrained as one of the core metrics that can shape the construction of "*trustful service unions*"**. If data transmission is not sufficiently protected, organizations will sooner or later find it difficult to protect sensitive information and ensure business continuity and operations - especially, considering the rise of cyber-threats including BGP hijacking and DDoS attacks. Accordingly, the **CASTOR vision seeks to establish a new Internet architecture providing organizations with secure data transmission paths on public networks: By advancing the selection of the corresponding Internet paths by introducing a quantified trust criterion to the corresponding routing decisions.** In this context, CASTOR capitalises different technologies such as trusted computing, trust assessment and optimization in order to manifest multipath control over zero-trust routing elements. This approach allows us to define trust in a non-intrusive manner and with enhanced scalability.

CASTOR is envisioning to be the **first-of-its-kind to ensure continuous secure data transmission over trusted paths** - going beyond the somewhat static trust properties (primarily integrity) that is considered by today's solutions (i.e., including the IETF recommendations on Trusted Path [27] Routing for enhanced scalability and path control). It extends the Trusted Path Routing (TPR) concept ensuring the only attested and trustworthy network devices are included in routing decisions. In such a model, considering the heterogeneity of the underlying infrastructure and (routing) computing resources, is becomes apparent that **trust levels vary**. Towards this end, the existence of various network operators, that correspond to different trust domains, over which a service may be deployed (necessitating for service continuity), requires the implementation of mechanisms for evaluating the level of trust for each party involved. This evaluation should take into account the dynamic nature of the environment along with its heterogeneity, particularly in relation to activities involving lifecycle management (i.e., secure enrolment or deployment) **supporting for global but heterogeneous trust**. Therefore, to prevent the formation of deceptive or impractical trust assumptions, it would be advantageous to implement a system founded on the concept of below-zero-trust, which not only entails a mindset of "*never trust, always verify*" but also can cope with frequent fluctuations on the trust level of a routing element which, in turn, might results to frequent (but trustworthy) routing updates. This approach mandates that every entity involved, whether physical or virtual, must provide substantiating evidence to establish its trustworthiness, irrespective of its location within the system. CASTOR adopts the "*never trust, always verify*" approach, since it im-

plements security measures at all network and infrastructure levels, regardless of the user or resource's location. It treats any user, device, or application attempting to access resources as untrusted and trust is continuously evaluated based on evidence.

This **continuum-wide trust quantification** requires advancements in both the infrastructure and network layer for exposing a computing base (set of security functions) capable of continuous trust characterizations (so as to be able to enforce trust) while maintaining the existing network agility. The goal is to identify the optimal network-management decisions over a set of pre-established path profiles (adhering to different network and trust characteristics) so as to be able to recommend the optimal set of forwarding paths featuring the required network agility over routing compute elements that can verifiably guarantee the required (from the SP) level of end-to-end assurance.

The present deliverable sets the foundation for the whole concept of runtime trusted path routing. The architectural framework revolves around three core layers: (i) the **Orchestration** that exposes the traffic engineering calculation and enforcement services; (ii) the **Network** that exposes the security functions and trust extensions; and (iii) the **Infrastructure**/**Routing** which manifests the dynamic (evidence-based) trust assessment based on a custom Trusted Computing Base, lightweight enough to be instantiated in each one of the routing elements. More specifically, D2.1 provides the conceptual CASTOR architecture and the functional and non-functional requirements along with the requirements derived from the four use cases that will guide the project's validation activities. The overall purpose of D2.1 is to serve as a reference document for the CASTOR project and as a key input to the technical work packages.

This deliverable establishes the technical foundation for CASTOR's innovations and contributions. It documents the project's vision by stating the problem that tackles along with the literature review on several relevant areas including orchestration, segment routing, trust and risk assessment, runtime monitoring, cryptographic mechanism, auditing through blockchain infrastructure and optimization techniques. D2.1 also introduces the project's system model and the CASTOR conceptual architecture, focusing on the different phases and component interconnection. A more granular architecture, including thorough descriptions of interfaces, message types, and information exchange, will be furnished in the subsequent deliverables of WP3, WP4, and WP5.

The document also provides the Service Provider's high-level requirements for mixed-criticality services as it pertains to route control agility. It highlights both the functional and non-functional requirements integral to the proposed CASTOR framework. Special attention is given to key aspects such as security, trust-assessment, operational assurance, trust-aware service assurance as well as traffic engineering requirements, that are necessary to establish an optimized and trusted communication path.

Moreover, the document goes beyond requirements by providing a comprehensive definition of four use cases and detailed user stories and scenarios. This use-case-driven approach, in conjunction with the identified requirements, serves as a foundation for eliciting the timeline for the development of the core CASTOR integrated framework. In addition, an initial set of KPIs per use case has been devised, hinting at business and technical indicators that will be tested under the piloting WP of the project.

Contents

1 Introduction **3**
1.1 Towards Dynamic Trust Assessment in the Compute Continuum . . . 3
1.2 Scope and Purpose . . . 4
1.3 Relation to other WPs and Deliverables . . . 4
1.4 Deliverable Structure . . . 4

2 CASTOR Vision & Background in Unlocking Scalability, Control and Trustworthiness Agility in Next-Generation Networks **6**
2.1 Vision and Problem Statement in Converging Network Agility with Trusted Path Routing . . . 6
2.2 Research Pillars and State-of-the-Art Analysis . . . 9
2.2.1 Service- and Network-Aware Resource Orchestration . . . 9
2.2.2 Orchestration . . . 11
2.2.3 Routing Protocols and Source Routing in Segment Routing . . . 13
2.2.4 Dynamic Trust Assessment and Governance . . . 14
2.2.5 Risk Assessment and Required Trust Level Calculation . . . 16
2.2.6 Establishing Trust in Network Devices with Secure Runtime Monitoring . . . 17
2.2.7 Support for Global but Heterogeneous Trust . . . 20
2.2.8 Service Certification and Auditing through Blockchain Infrastructure . . . 23
2.2.9 Complex Multi-Constraint and Multi-Objective Optimization Process . . . 24

3 System Model and Assumptions **26**
3.1 Conventions and Definitions . . . 26
3.2 CASTOR as a *Trusted* Routing Path Extension towards Secure, Reliable Connectivity . . . 28
3.3 Threat Model . . . 30

4 Extending Trusted Path Routing: Manifesting Evidence-based Theory for Runtime/Explicit Trust Assessment **32**
4.1 Definition of Trust and Trustworthiness . . . 33
4.1.1 Overall Principles . . . 33
4.1.2 Trust Properties of Interest in CASTOR . . . 34
4.2 Elevating Trust metrics as a core enabler in Traffic Engineering Provisioning . . . 35
4.2.1 Current Considerations in Trusted Path Routing . . . 35

4.2.2 Trust objectives in Service-Level Agreements . . . 36
4.3 Subjective Logic as a Foundation for Evidence-Based Trust Assessment . . . 37
4.3.1 From Evidence to Opinions . . . 37
4.3.2 Discounting and Indirect Evidence . . . 38
4.3.3 Fusion of Multi-Source Evidence . . . 38
4.4 CASTOR TAF high-level description . . . 39
4.4.1 The Architecture of the Trust Assessment Framework (TAF) . . . 40
4.4.2 The Local TAF agent . . . 41
4.4.3 The Global TAF . . . 43
4.5 Open Questions Relating to Trust Characterisation of Routers, Links and Paths . . . 45
4.5.1 Information Sharing and Trust Models . . . 45
4.5.2 Managing Computational Dependencies and Discounting . . . 46
4.5.3 Modelling Uncertainty . . . 47
4.5.4 Challenges in the composition of trust propositions to achieve link and path-level trust 47
4.5.5 Subjective Logic Fusion . . . 48

5 Multi-Path Control & Agility for Optimal {*Network, Trust*}-Aware E2E Path Construction 50
5.1 Explicit Path Identification . . . 51
5.1.1 A Generic Example of Multi-Constraint Optimization Problem Definition . . . 53
5.2 Control Plane: Beaconing for Optimal Forwarding Path Identification . . . 53

6 CASTOR Conceptual Architecture and Functional Components 57
6.1 CASTOR Conceptual Architecture . . . 57
6.1.1 Preparedness phase . . . 59
6.1.2 Service Registration phase . . . 60
6.1.3 Proactive phase . . . 61
6.1.4 Reactive phase . . . 62
6.2 CASTOR Functional Components . . . 64
6.2.1 SLA Translation & Decomposition . . . 66
6.2.2 CASTOR Orchestration . . . 67
6.2.3 Distributed Ledger Technologies . . . 71
6.2.4 CASTOR Risk Assessment Engine . . . 76
6.2.5 Global and Local Trust Assessment . . . 81
6.2.6 On-board Finite State Machine Analyser . . . 86
6.2.7 Optimization Engine . . . 88
6.2.8 Trustworthy Platform Attestation and TNDI Onboarding/Runtime . . . 90
6.2.9 CASTOR Tracing Capabilities . . . 93
6.2.10 Composite Attestation . . . 96
6.2.11 Crypto Structures & Building Blocks . . . 99

7 CASTOR Methodology **102**

7.1 Methodology for MVP Design . . . 102
7.1.1 Requirements Definition Process . . . 103
7.2 Requirements Elicitation Methodology . . . 103
7.2.1 Technical Requirements Specification Process . . . 104
7.2.2 Use Case Requirements Specification Process . . . 104

8 CASTOR Use Cases **107**

8.1 High-Level Introduction of the CASTOR Use Cases Towards Trusted Traffic Engineering process . . . 107
8.2 Highly Available & Secure Airspace Monitoring in Urban Air Mobility (UAM) Environments . 110
8.2.1 System Model, Communication Interfaces, and Protocols . . . 111
8.2.2 “As-is” Scenario . . . 114
8.2.3 Collins Use Case needs from CASTOR . . . 115
8.2.4 To-be Reference Scenario 1: On-Airport Trusted-Routing Loop for Real-Time Surveillance . . . 116
8.2.5 To-be Reference Scenario 2: Collaborative Airport Operational Control Centres for Agile Decision Making . . . 118
8.2.6 Reference Scenario 1 User Stories . . . 119
8.2.7 Reference Scenario 2 User Stories . . . 126
8.3 Trustworthy Communications of First Responder Mobile Units and the Compute Continuum 134
8.3.1 “As-is” Scenario . . . 135
8.3.2 System Model and Communication . . . 136
8.3.3 Scenario Needs from CASTOR . . . 139
8.3.4 To-be Reference Scenario 1: Connectivity to V2X PKI over cross-domain path provisioning . . . 140
8.3.5 To-be Reference Scenario 2: OTA Updates over trustworthy paths . . . 140
8.3.6 Reference Scenario User Stories . . . 142
8.4 Priority-based Trusted Messaging & Scalable Performance for CCAM Applications . . . 149
8.4.1 “As-is” Scenario . . . 149
8.4.2 System Model . . . 149
8.4.3 Scenario Needs from CASTOR . . . 150
8.4.4 Improved model using CASTOR’s framework . . . 152
8.4.5 Reference Scenario User Stories . . . 153
8.5 Future-Proofing Next-Generation Unmanned Aerial Vehicles Communications towards Critical Infrastructure Sustainability . . . 157
8.5.1 System Model . . . 157
8.5.2 “As-is” Scenario . . . 160
8.5.3 Use Case needs from CASTOR . . . 161
8.5.4 ”To be” Reference Scenario 1: CASTOR in the Data Network . . . 162

8.5.5 ”To be” Reference Scenario 2: External Risk Indices as input to CASTOR in Data Network . . . 162
8.5.6 Reference Scenario User Stories for Scenarios 1 and 2: CASTOR in the Data Network and Risk-aware Path Selection . . . 162
8.5.7 ”Nice-to-have” Scenario User Stories: CASTOR in the Shared Back-haul Infrastructure . . . 170
8.6 Trust-Aware UAV Data Delivery Across Mobile Edge Attachments . . . 173

9 CASTOR Framework Requirements 176
9.1 Overarching Security Requirements . . . 178
9.2 Functional and Non-Functional Requirements . . . 195
9.2.1 Trust Assessment Requirements . . . 195
9.2.2 Router Operational Assurance . . . 201
9.2.3 Trust-aware Service Assurance . . . 204
9.2.4 Traffic Engineering Requirements . . . 218

10 Summary and Conclusions 221

References 225

List of Figures

1.1 Relation of D2.1 with other WPs and Deliverables . . . 5

2.1 CASTOR Layered Architecture for Extending the Vision of Trusted Path Routing . . . 8

3.1 System Model and key CASTOR values in the Traffic Engineering landscape numbered from 1 to 9. . . . 29

5.1 Routing topology with network and trust attributes and possible paths satisfying different policies . . . 52

6.1 High-level phases in CASTOR framework . . . 57
6.2 CASTOR high-level architecture . . . 59
6.3 Translation of intents to enforceable policies . . . 66
6.4 Orchestration flow of actions when there is an intent-based service order . . . 68
6.5 Orchestration flow of actions when a node joins the network . . . 68
6.6 Orchestration flow of actions when a node leaves/drops the network . . . 69
6.7 PCE Extension . . . 71
6.8 CASTOR DLT flow of actions - Storage of SLAs and SSLAs . . . 74
6.9 CASTOR DLT flow of actions - Recording of Trust Policies . . . 75
6.10 CASTOR DLT flow of actions - Abstraction of Trust Capabilities . . . 76
6.11 Phase 1: CASTOR router type onboarding & RTL Calculation . . . 79
6.12 Phase 2: Topology Integration & Cascading Attack Analysis . . . 81
6.13 CASTOR TAF design phase . . . 83
6.14 CASTOR TAF router deployment phase . . . 85
6.15 CASTOR TAF runtime phase . . . 86
6.16 Onboard Finite State Machine analyser flow of actions . . . 88
6.17 Optimization Engine action flow . . . 89
6.18 TNDE and TNDI flows during join and onboarding phases (attestation and configuration) . . . 91
6.19 TNDE and TNDI flows during the runtime phase (trust assessment and data sharing) . . . 93
6.20 CASTOR's multi-level tracing architecture proposal . . . 94
6.21 The workflow of composite attestation . . . 97
6.22 The structure of layered attestation . . . 98
6.23 The workflow of ORE in CASTOR for different domains . . . 99

8.1 Deployment of Key System Components . . . 112
8.2 Flight Plan Registration Information Flow . . . 113
8.3 Unmanned Airspace Surveillance Information Flow . . . 114
8.4 Identification of airspace threat potentially compromised . . . 116
8.5 The Surveillance Continuum . . . 117
8.6 Multiple airspace domains require coordination and cooperation . . . 118
8.7 Current Inter-Zone Data Exchange Model . . . 119
8.8 Exchanging trust assessments with data can improve operational security and efficiency . 119
8.9 Normal behaviour without CASTOR . . . 120
8.10 Attack behaviour without CASTOR . . . 121
8.11 Attack prevented from starving CISP of attack-revealing data . . . 124
8.12 Significant ATL degradation results in operator notification . . . 125
8.13 Zone B has no insights into the trust provenance of data consumed from the CISP . . . 128
8.14 Zone B now has insights into the trust provenance of data consumed from the CISP . . . 130
8.15 Instead of applications tagging with trust indicator, receiver can query origin trustworthiness 133
8.16 The big picture of C-Roads topology of European C-ITS systems . . . 134
8.17 End-to-end communication topology of vehicles and C-ITS service access . . . 137
8.18 Functional architecture of hybrid C-ITS network adopted to CASTOR . . . 139
8.19 C-ITS service access enhanced by CASTOR . . . 141
8.20 UC2.US1a Workflow . . . 144
8.21 UC2.US.2 Workflow . . . 147
8.22 UC2.US3 Workflow . . . 148
8.23 Illustration of the network path for CAM and DENM messages . . . 151
8.24 Message routing using CASTOR's framework . . . 152
8.25 UC3.US1 Workflow . . . 154
8.26 UC3.US2 Workflow . . . 155
8.27 UC3.US3 Workflow . . . 156
8.28 UAV-related System Model . . . 158
8.29 The communication flow in UC4 . . . 159
8.30 Sequence Diagram of UC4.US1a - Nominal operation . . . 164
8.31 US4.US1b: Fallback SSLA . . . 166
8.32 UC4.US1c: SSLA compliance report . . . 168
8.33 UC4.US2a: Risk Index . . . 169
8.34 UC4.US3a: Nominal Operation . . . 171
8.35 Workflow of UC4.UC3b . . . 172
8.36 PoC Scenario . . . 173

List of Tables

4.1 Contextual properties of trust. . . . 35

6.1 CASTOR Artifacts - Naming convention . . . 65
6.2 Comparison of the Trace Units considered for CASTOR. . . . 95

8.1 UC1 - Highly Available & Secure Airspace Monitoring in Urban Air Mobility (UAM) Environments . . . 108
8.2 UC2 - Trustworthy Communications of First Responder Mobile Units and the Compute Continuum . . . 109
8.3 UC3 - Priority-based Trusted Messaging & Scalable Performance for CCAM Applications . 109
8.4 UC4 - Future-Proofing Next-Generation Unmanned Aerial Vehicles Communications towards Critical Infrastructure Sustainability . . . 110
8.5 Short Glossary of Terms from Airspace Management . . . 110
8.6 Network and Trust properties as service-level objectives . . . 116
8.7 Reference Values for Use Case 1, Scenario 1 . . . 121
8.8 CASTOR KPIs for Use Case 1, Scenario 1, User Story 1(b) . . . 122
8.9 CASTOR KPIs for Use Case 1 - User Story 1(c) NetOps Alerting . . . 126
8.10 Reference Values for Use Case 1, Scenario 2 . . . 128
8.11 CASTOR KPIs for Use Case 1, Scenario 2, User Story 2(b) . . . 130
8.12 CASTOR KPIs for Use Case 1, Scenario 2, User Story 2(c) . . . 133
8.13 Network and Trust properties as service-level objectives . . . 140
8.14 CASTOR KPIs for User Story UC2.US1a . . . 144
8.15 CASTOR KPIs for User Story UC2.US1b . . . 145
8.16 CASTOR KPIs for user story UC2.US2 . . . 147
8.17 CASTOR KPIs for user story UC2.US3 . . . 149
8.18 Cause description triggering generation and transmission of DENM message [4] . . . 150
8.19 Network and Trust properties as service-level objectives . . . 153
8.20 Quantitative KPIs for user story UC3.US1 - Traffic Operator . . . 156
8.21 Qualitative KPIs for user story UC3.US1 - Traffic Operator . . . 156
8.22 Quantitative KPIs for user story UC3.US2 - Emergency Operator . . . 156
8.23 Qualitative KPIs for user story UC3.US2 - Emergency Operator . . . 157
8.24 Quantitative KPIs for user story UC3.US3 - VRU . . . 157

8.25 Qualitative KPIs for user story UC3.US3 - VRU . . . 157
8.26 Network and Trust properties as service-level objectives . . . 161
8.27 Reference Values for user story UC4.US1a . . . 164
8.28 CASTOR KPIs for user story UC4.US1b . . . 166
8.29 CASTOR KPIs for user story UC4.US1c . . . 168
8.30 CASTOR KPIs for user story UC4.US2a . . . 170
8.31 Reference Values for user story UC4.US3a . . . 171
8.32 CASTOR KPIs for user story UC4.US3b . . . 173

9.1 Functional & Non-functional Requirement Categories . . . 177
9.2 SR.1 Device-support for Hardware-based Root of Trust . . . 178
9.3 SR.2 HW-based Isolation of CASTOR's device-side TCB Components . . . 178
9.4 SR.3 Secure Device Key Management with Platform and TNDI Binding Support . . . 179
9.5 SR.4 Runtime Configuration Integrity Check of Routing Plane Sw/Hw Stack . . . 180
9.6 SR.5 Dynamic Security Function Placement . . . 182
9.7 SR.6 Onboarding of Network Devices and their TNDIs into CASTOR . . . 183
9.8 SR.7 Secure E2E CASTOR-to-Device Control and Data Channels . . . 184
9.9 SR.8 Secure and Efficient Cryptography . . . 185
9.10 SR.9 Secure Reporting of Traces and Trustworthiness Data . . . 186
9.11 SR.10 Secure Link Establishment between TNDIs . . . 187
9.12 SR.11 Secure Runtime Tracing Support of TNDIs . . . 188
9.13 SR.12 Composition of Trustworthiness Evidence . . . 189
9.14 SR.13 Runtime Operational Assurance and Process Execution Integrity Checks . . . 190
9.15 SR.14 Ordering of Attestation Evidence . . . 191
9.16 SR.15 Secure Data Handling and Provenance . . . 192
9.17 TAF.R.1 Generalizability . . . 195
9.18 TAF.R.2 Correctness . . . 196
9.19 TAF.R.3 Robustness and Resilience . . . 197
9.20 TAF.R.4 Flexibility of Trust Sources . . . 198
9.21 TAF.R.5 Performance of trust evaluations . . . 199
9.22 TAF.R.6 Scalability . . . 200
9.23 TNDE.R.1 Management of Network Device TNDIs . . . 201
9.24 TNDE.R.2 Dynamic Setup and Configuration of the TNDE and TNDIs (Re-/Programmability) 201
9.25 FSM.R.1 Model optimisation and specialisation . . . 202
9.26 FSM.R.2 Model explainability . . . 203
9.27 OSS.R.1 Secure Remote Asset Management and Reconfiguration Effectiveness . . . 204
9.28 OSS.R.2 Trust- and Policy-driven Orchestration and Service Placement . . . 205
9.29 OSS.R.3 Accurate and fresh synchronization of the network topology attributes and trust assurance reports . . . 206

9.30 OPT.R.1 Trusted Path Optimization . . . 207
9.31 OPT.R.2 Network and attestation information . . . 207
9.32 OPT.R.3 Multi-path computation . . . 208
9.33 OPT.R.4 Re-optimization . . . 208
9.34 OPT.R.5 Optimization Scalability . . . 209
9.35 RA.R.1 RTL Derivation and Management . . . 209
9.36 RA.R.2 Cascading Attack Detection . . . 210
9.37 RA.R.3 Black-box risk analysis . . . 212
9.38 POLICY.R.1 Intra-domain translation to SLA (SSLA) and policies . . . 213
9.39 POLICY.R.2 Cross-domain translation to SLA (SSLA) and policies . . . 213
9.40 DLT.R.1 Secure Storage of Security Claims . . . 214
9.41 DLT.R.2 Authentication & Authorization in CASTOR Blockchain . . . 215
9.42 DLT.R.3 Secure Oracle . . . 216
9.43 DLT.R.4 Network Trust Exposure Capability . . . 217
9.44 TE.R.1 Automated Traffic Engineering TE capabilities (to cope with network conditions dynamicity) . . . 218
9.45 TE.R.2 Constraint-based path computation capabilities (for domain-wide traffic) . . . 218
9.46 INTER-DOM.TE.R.1 Trust summary exchange . . . 219

Versioning and contribution history

Version	Date	Author	Notes
v0.1	13.2.2025	Nikos Fotos (UBITECH)	Table of Contents
v0.2	28.2.2025	Michael McElligott (COLLINS), Vlad Chiriac (TUIASI), Gergely Kovacs (COMMSIGNIA), Ioannis Boukas, Evangelos Syrmos (K3Y)	First input on use cases (AS-IS Scenario and TO-BE scenario)
v0.3	17.3.2025	ALL	First input on Threat Modelling (Chapter 3)
v0.4	31.3.2025	ALL	First consolidated input on State-of-the-Art analysis
v0.5.1	28.4.2025	ALL	First round of Sequence Diagrams in Chapter 6
v0.5.2	5.5.2025	ALL	Revised input in use cases: AS-IS Scenario descriptions, and identification of use case needs from CASTOR
v0.6.1	19.5.2025	ALL	Updates to Sequence Diagrams in Chapter 6
v0.6.2	26.5.2025	Michael McElligott (COLLINS), Vlad Chiriac (TUIASI), Gergely Kovacs (COMMSIGNIA), Ioannis Boukas, Evangelos Syrmos (K3Y)	First input on use cases (AS-IS Scenario and TO-BE scenario) & Updates to use case descriptions and trust assessment properties of interest per scenario
v0.6.3	9.6.2025	Anuj Pathania, Andy Pimentel (UvA), Theo Dimitrakos (UKENT)	First input on problem statement for optimization in traffic engineering. Complete first consolidated problem statement for Trust Assessment.
v0.7.0	30.6.2025	Yalan Wang (SURREY), Fabian Schwarz (NVDIDIA)	Updates to SotA and in-router CASTOR architecture and crypto primitives description.
v0.7.1	14.6.2025	Iasonas Sakellariou, Symeon Tsintzos (QUBITECH)	Review problem statement in Chapter 4, and provide comments to the document.
v0.7.2	21.6.2025	Anuj Pathania	Resolve comments and update problem statement in Chapter 4 and description of optimization-related sequence diagrams in Chapter 6.
v0.7.3	14.7.2025	Alexandru Coleș (ORO) Reviewed System Model and conventions in Chapter 3.	
v0.7.4	28.7.2025	ALL	Updates to sequence diagram descriptions based on latest advancements in architecture.
v0.8.0	8.9.2025	Nikos Fotos, Thanassis Giannetsos (UBITECH)	Provide updates to the architecture diagram based on the incorporation of the four CASTOR phases. Include refinements to the interactions with the Optimization Engine.
v0.8.1	22.9.2025	ALL	Final refinements on all sequence diagrams in Chapter 6. Updates to System Model (Threat Model) in Chapter 3.
v0.8.2	6.10.2025	Nikos Fotos (UBITECH)	Included template for Functional Requirements to include relevant background and definition information in Chapter 9.
v0.8.3	13.10.2025	ALL	First input on Functional Requirements, and a first set of KPIs per technical artifact
v0.8.4	20.10.2025	Michael McElligott (COLLINS), Vlad Chiriac (TUIASI), Gergely Kovacs (COMMSIGNIA), Ioannis Boukas, Evangelos Syrmos (K3Y)	Final polishing on the Use Case User Stories
v0.8.5	27.10.2025	ALL	Final descriptions on Functional Requirements, and consolidated KPIs in Chapter 9.

0.8.6	31.10.2025	ALL	Resolve comments from internal review in Chapters 2 and 6.
v0.9.0	3.11.2025	Nikos Fotos	Update Use Case User Story template to incorporate KPIs.
v0.9.1	10.11.2025	Nikos Fotos, Thanassis Giannetsos (UBITECH), Jamie Pont (KENT), Yalan Wang, Liqun Chen (SURREY)	Refinements to open challenges of the Trust Assessment problem statement in Chapter 4. Revision to composite attestation schemes: update references to for multi-ordered signatures.
v0.9.2	17.11.2025	Nikos Fotos, Sofianna Menesidou, Thanassis Giannetsos (UBITECH)	Polishing on the System Model description: Revised Figure and assumptions/considerations in Chapter 3. First draft of Problem Statement and Vision of CASTOR in Chapter 1.
v0.9.3	24.11.2025	Michael McElligott (COLLINS), Vlad Chiriac (TUIASI), Gergely Kovacs (COMMSIGNIA), Ioannis Boukas, Evangelos Syrmos (K3Y)	Final Updates to KPIs on Engineering Stories
v1.0	24.11.2025	(Nikos Fotos, Thanassis Giannetsos, Sofianna Menesidou (UBITECH)	Final polishing round on Architecture and final updates in PoC description
1.1	1.12.25	Daphne Galani (UBITECH)	Final Review & Submission

Chapter 1

Introduction

1.1 Towards Dynamic Trust Assessment in the Compute Continuum

The advancement of on-device edge technologies is enabling substantial computational capacity across a wide range of devices, including network equipment and routers. This shift makes it possible to combine on-device edge resources with other forms of edge computing and diverse cloud services within collaborative computing environments. However, integrating heterogeneous computing resources with varying network capabilities requires intelligent orchestration—one that can also optimize for security and sustainability. As hybrid networks, edge computing, and full cloud migration reshapes connectivity infrastructure, both network equipment and services are at risk [62].

Since we are currently transitioning toward these Connected Collaborative Computing (3C Network) environments, it is essential to investigate the emerging technological challenges associated with this transformation. This transformation stimulates new applications that need intelligence and strong requirements regarding the content delivery networks in the edge and far edge to run in a completely decentralized and distributed manner. These demands have accelerated the transition toward shared infrastructures, thereby the adoption of virtualized software functions. We are moving away from the traditional segmentation model, where each administrative domain maintained its own routing topologies, infrastructure, and security solutions, to shared infrastructures where multiple network operators deploy services and routing capabilities through a virtualization layer (i.e., virtual network functions).

All these developments intensify the need for Trusted Path Routing (TPR). This problem, already acknowledged within the IETF, aims to engender trust as a core dimension that influences the traffic engineering process. The goal is to ensure that services traverse infrastructure that provides not only the required network flexibility but also strong guarantees of trustworthiness. In general, most works that consider trustworthiness focus only on the integrity. However, integrity is not the only trust property, but several other properties exist such as safety, reliability, resilience, etc. In addition, capturing such an extensive set of properties is not possible with traditional cryptography.

Towards this direction, CASTOR significantly advances the state of the art by introducing a set of mechanisms that act as verticals, enabling a more effective capture of the traffic engineering process. Its innovation is two-fold: (a) adapting to trust changes and modelling the trust state, and (b) recommending an optimal set of paths. Deliverable D2.1 focuses on the first innovation and outlines the architecture that captures all relevant building blocks and their interdependencies required to adapt to dynamic trust changes. Consequently, D2.1 provides network and security mechanisms that complement routing solutions to support device connectivity over trustworthy routers and related infrastructure.

1.2 Scope and Purpose

This deliverable plays a foundational role within CASTOR, setting forth the essential technical requirements that have shaped the architectural framework of CASTOR. Beyond functional requirements, CASTOR incorporates non-functional requirements, particularly concerning security, trust, operational and traffic engineering aspects across the Compute Continuum.

This deliverable dives deep into the conceptual architecture, functional components, and their vital interconnections, offering an extensive exploration of the CASTOR framework's design capturing also all operational flows. The research carried out in this document not only provides in-depth insights into CASTOR but also highlights its significance in various use cases, reference scenarios and the user stories to be investigated within these scenarios.

This first deliverable concentrates, due to the complexity, primarily on intra-domain aspects to fully document the operational flows, while sets the scene for the next iteration in D2.2, which will address the remaining open questions related to inter-domain operational flows.

1.3 Relation to other WPs and Deliverables

As the reference architecture and technical requirements deliverable, this deliverable serves as the basis for all later WPs and deliverables. This first initial version of the deliverable includes the complete documentation of the CASTOR requirements of T2.1, the CASTOR reference architecture of T2.2, the classification and analysis of the threat landscape of T2.3 and the analysis of the type of secure elements existing for the heterogeneous type of continuum elements (from the far-edge to the edge and to the routing plane of T2.5 until M12.

D2.1 provides to WP3 the definition of requirements and the high-level CASTOR architecture of the in-router artifacts, including the CASTOR Trust Sources. This will assist the detailed threat model analysis and the mapping of threats to the respective evidence that needs to be collected by the CASTOR Trust Sources and to be documented in D3.1. Also, D2.1 provides to WP4 the definition of overarching CASTOR TAF challenges, and requirements that need to be addressed towards a compute continuum-wide and federated trust decisions based on the evidence-based ATL and the risk-aware Required Trust Level (RTL) values. Finally, WP5 and WP6 will also be fed with the requirements and challenges towards the optimal selection of trust-aware traffic engineering policies and the use case definition and requirements respectively. In addition, D2.1 constitutes the baseline for Milestone MS1 - Availability of CASTOR Reference Architecture & Operational Landscape to be met by the CASTOR framework, as it delivers the architectural specification of the CASTOR architecture.

The final version of this deliverable (D2.2) will be provided during M24 and will include the complete developments of T2.2-T2.5, including the harmonized Trustworthiness Profiles capturing the trustworthiness controls required and potential updates on the requirements, KPIs and the reference architecture considering also the cross-domain end-to-end service provisioning. The inter-dependencies among the CASTOR WPs are shown in Figure 1.1.

1.4 Deliverable Structure

Chapter 2 provides the vision of CASTOR, documenting the problem that tackles along with the background and state-of-the-art analysis on several relevant areas such as orchestration, segment routing, trust and risk assessment, runtime monitoring, cryptographic mechanism, auditing through Blockchain infrastructure and optimization techniques. **Chapter 3** describes the system model and threat landscape

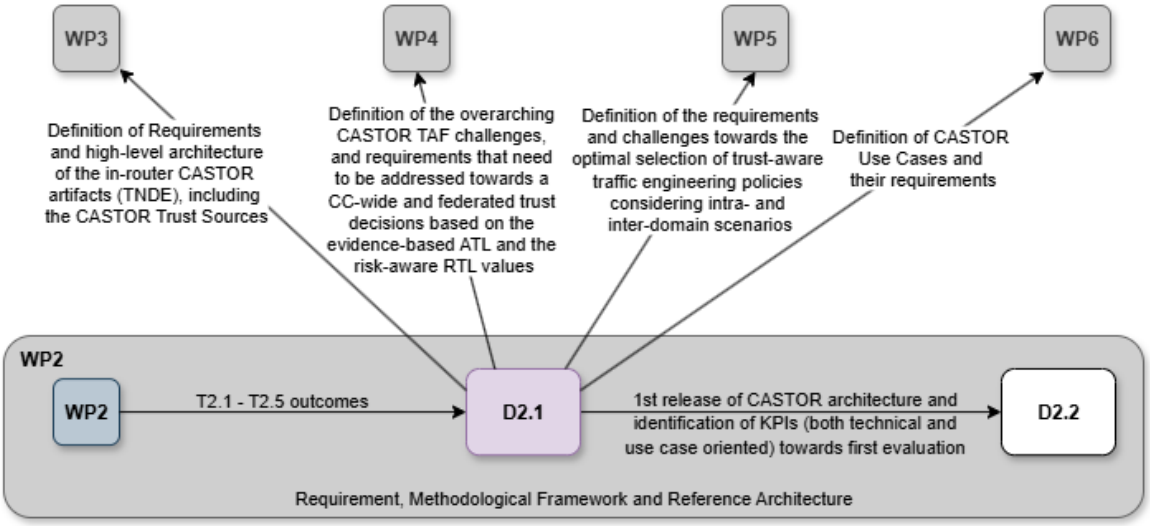

Figure 1.1: Relation of D2.1 with other WPs and Deliverables

analysis in order to set the scene for the key contributions of CASTOR in the context of traffic engineering and the underlying threats to be considered. **Chapter 4** focuses on the concepts of trust and trustworthiness and the CASTOR Trust Assessment Framework (TAF) and discusses the key challenges and open questions for bringing such a framework in the routing plane. **Chapter 5** elaborates on the problem of trusted path routing optimization and discusses the key challenges and open questions on how to solve multi-objective problems that include both trust and network metrics. **Chapter 6** building on top of the system model, describes in detail the conceptual architecture including the different phases, the functional components and the core functional objectives of the project. The overall methodology of defining the CASTOR Minimum Viable Product (MVP) along with the requirements is presented in **Chapter 7**. **Chapter 8** outlines the four use cases of different application domains in detail, including the reference scenarios adopted by each demonstrator. The user stories from which the requirements will be drawn as part of the scenarios are described in detail and will govern the demonstrations of the CASTOR. This chapter also sets the scene for the cross domain evaluation.**Chapter 9** serves with detailed lists of the functional and non-functional technical requirements (e.g., security, trust, operational assurance and traffic engineering) that will assist on the detailed functional specification that will be done on the technical deliverables, and **Chapter 10** concludes the deliverable.

Chapter 2

CASTOR Vision & Background in Unlocking Scalability, Control and Trustworthiness Agility in Next-Generation Networks

2.1 Vision and Problem Statement in Converging Network Agility with Trusted Path Routing

Routing decisions determine the way information flows throughout the Internet and shape the performance and quality of globally-deployed services. Routing traditionally amounts to the way that network data packets are steered from their origin to their destination over topologies made up by interconnected network devices (i.e., routers) [89]. Typically, in IP networks, the original data are broken down to packets tagged with headers containing the destination IP address. To steer the flow of packets, network routers examine each packet's address and determine the next hop to the desired destination based on appropriately-updated routing tables. The process scales across numerous (at the order of thousands or millions) network devices to finally offer the global-scale delivery of Internet data flow.

Each selected path can be shaped by diverse criteria (such as hop count, bandwidth, and delay) establishing paths of the shortest distance, or achieving the fastest or the least congested Internet route to the destination. However, **security mechanisms for converging such high-degree of network efficiency with the establishment of secure data channels is only currently considered as an add-on and not inherently engrained as one of the core metrics that can shape the construction of "*trustful service unions*"**. If data transmission is not sufficiently protected, organizations will sooner or later find it difficult to protect sensitive information and ensure business continuity and operations - especially, considering the rise of cyber-threats including BGP hijacking and DDoS attacks. Accordingly, the **CASTOR vision seeks to establish a new Internet architecture providing organizations with secure data transmission paths on public networks: By advancing the selection of the corresponding Internet paths by introducing a quantified trust criterion to the corresponding routing decisions.**

While in early days of (local) networks, routing was statically determined by network administrators who manually configured the routing tables, the explosion of Internet (at around three decades ago) rendered this approach obsolete, if not infeasible; a single link failure or a topology update required human intervention leading to unacceptably slow convergence. Dynamic routing protocols such as the Distance Vector instances [e.g., RFC 1058: Routing Information Protocol] were then introduced. The concept suggests that routers periodically exchange a copy of the entire routing table with their immediate neighbours. This simple-to-implement approach allowed for a distance metric (i.e., hopcount), often proved insufficient, while suffering from slow convergence and issues such as the "count-to-infinity" pertaining to the potential occurrence of routing loops after a link failure.

Link state protocols such as Open Shortest Path First (OSPF) [IETF RFC 1131] and Intermediate System to Intermediate System (IS-IS) [IETF RFC 1195 and ISO 10589] were developed and introduced in late 80s and early 90s. The involved concept had each router broadcasting information about its directly connected links (marking its "state") to every router located within its administrative domain, typically defined as Autonomous System (AS); the network area where the same routing protocol is used. Availing information for the complete network map, each router can independently participate to complex algorithms (like Dijkstra's algorithm) and calculate the "best path" to every network destination. Their fast convergence and increased scalability properties render that kind of protocols as the dominant solutions for routing in enterprise and Internet Service Provider (ISP) networks.

Moving beyond a single Autonomous System (AS) constituted the latest development of the Internet routing fundamentals. The need to interconnect thousands of independent ASes, shaping the global Internet has been addressed by the Border Gateway Protocol (BGP) [IETF RFC 4271: A Border Gateway Protocol 4]. It is an instance of Path Vector protocols, having the entire sequence of ASes that should be crossed to reach the destination (AS), advertised across domains. BGP allows for policy-driven routing shaped by business agreements (e.g., on transit costs) between ISPs, rather than solely technical metrics. Sharing reachability information through BGP offers the glue ingredient that holds the entire public Internet together.

The CASTOR approach seeks to capitalise on the aforementioned (mature) routing technologies, adopting a more NFV-like approach. The increasingly pervasive network virtualisation technologies call for a "transformation" of proprietary network hardware operations to Virtualized Network Functions (VNFs) that would automate the entire lifecycle of the CASTOR (virtualized) routing services. With virtualization technologies, a router's control plane (which runs the routing protocols like OSPF, IS-IS, and BGP) is decoupled from the data plane (or forwarding plane) that operates for forwarding the packets. An orchestration software can offer the ability to automate (Section 6.2.2) and centralize (Section 6.2.2.1) the complex CASTOR logic that would (traditionally) require to be manually engineered.

CASTOR is envisioning to be the **first-of-its-kind to ensure continuous secure data transmission over trusted paths** - going beyond the somewhat static trust properties (primarily integrity) that is considered by today's solutions (i.e., including the IETF recommendations on Trusted Path [27] Routing and SCION [22] protocol for enhanced scalability and path control). It extends the Trusted Path Routing (TPR) concept ensuring the only attested and trustworthy network devices are included in routing decisions. In such a model, considering the heterogeneity of the underlying infrastructure and (routing) computing resources, is becomes apparent that **trust levels vary**. Towards this end, the existence of various network operators, that correspond to different trust domains, over which a service may be deployed (necessitating for service continuity), requires the implementation of mechanisms for evaluating the level of trust for each party involved. This evaluation should take into account the dynamic nature of the environment along with its heterogeneity, particularly in relation to activities involving lifecycle management (i.e., secure enrolment or deployment) **supporting for global but heterogeneous trust**. Therefore, to prevent the formation of deceptive or impractical trust assumptions, it would be advantageous to implement a system founded on the concept of below-zero-trust, which not only entails a mindset of "*never trust, always verify*" but also can cope with frequent fluctuations on the trust level of a routing element which, in turn, might results to frequent (but trustworthy) routing updates. This approach mandates that every entity involved, whether physical or virtual, must provide substantiating evidence to establish its trustworthiness, irrespective of its location within the system. CASTOR adopts the "*never trust, always verify*" approach, since it implements security measures at all network and infrastructure levels, regardless of the user or resource's location. It treats any user, device, or application attempting to access resources as untrusted and trust is continuously evaluated based on evidence.

Each forwarding element is evaluated by a Verifier prior to its inclusion in a trusted network domain. Evidence about the device's integrity is assessed to determine its eligibility for participation in the routing topology. **While this enrolment-time verification establishes a baseline of trust, it does not account for the fact that a device's trustworthiness may change over time.** If a device becomes

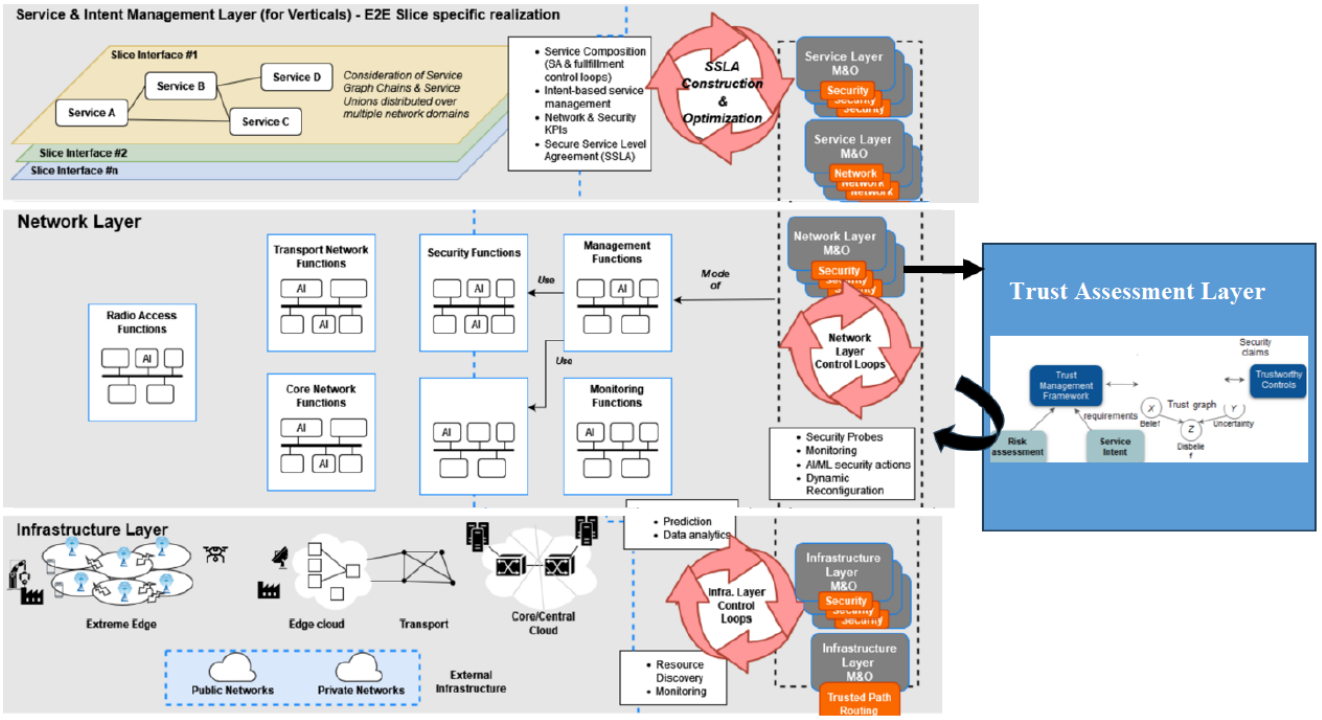

Figure 2.1: CASTOR Layered Architecture for Extending the Vision of Trusted Path Routing

misconfigured, compromised, or enters a degraded trust state after initial enrolment, this change should be reflected in the trusted path routing decisions. The TPR model [27], as currently defined, provides no mechanism to detect or respond to such changes. Extending it with a runtime trust assessment phase raises several open issues that we need to resolve. This constitutes one of the core challenges that CASTOR aims to address (Section 2.2.7.2) for the employment of a generic trust assessment methodology capable of continuous (real-time) trust evaluation and quantification. Trustworthiness is one of the main research challenges in the flagship effort of the 3CN Project (unlocking the vision of "*Collaborative, Cognitive Computing over Shared Infrastructures*") and it constitutes a basic foundation towards fulfilling the vision of a complete zero trust security model for building service-graph-chains and service unions with high Level of Trust (LoT) and Level of Assurance (LoA). This can be manifested by CASTOR capabilities for assessing the trust level of all elements (Figure 2.1) expanding throughout the service application and network layers (assessing the LoA of all network infrastructure elements including hardware and virtualization platforms as well as all VNFs) all the way down to the infrastructure layer for enabling trustworthy network decisions. This **continuum-wide trust quantification** requires advancements in both the infrastructure and network layer for exposing a computing base (set of security functions) capable of continuous trust characterizations (so as to be able to enforce trust) while maintaining the existing network agility. The goal is to identify the optimal network-management decisions over a set of pre-established path profiles (adhering to different network and trust characteristics) so as to be able to recommend the optimal set of forwarding paths featuring the required network agility over routing compute elements that can verifiably guarantee the required (from the SP) level of end-to-end assurance.

As will be seen later on (Section 2.2), we have witnessed advancements with respect to trust evaluations in both cloud environments (e.g., evaluating the trustworthiness of Virtual Network Functions in central B5G/6G deployments) and far-edge ecosystems (e.g., ensuring the trustworthiness of the data integrity used as part of V2X cooperative communication). However, one of CASTOR's key objectives is to bridge the two ends of the Compute Continuum by introducing the concepts of continuous and dynamic trust evaluations in the routing plane.

2.2 Research Pillars and State-of-the-Art Analysis

2.2.1 Service- and Network-Aware Resource Orchestration

2.2.1.1 Services and Intents

The concept of ”**intent**” was first introduced in RFC 7575 [107] within the context of Autonomic Networking, where it is defined as “an abstract, high-level policy used to operate the network”. According to this definition, an intent represents a user-supplied directive (e.g., policy) that guides an Autonomic Network, which would otherwise function without human intervention. However, to prevent the term from being used merely as a synonym for policy, it is necessary to establish a clear distinction between intent and other policy types. An intent expresses the desired state of a system and is used to describe an expected network or service. Intents neither specify concrete configurations nor prescribe the management tasks to be performed by a system. Instead, they enable consumers to request networks and services without requiring detailed knowledge of how those outcomes will be realized. This implicitly assumes that the system can infer the behavior of networks and services, applying intelligence and automation to satisfy the intent. This not only relieves the consumer of the burden of knowing implementation details but also provides flexibility allowing the producer to explore alternative options to find optimal solutions. An intent is a set of expectations including requirements, goals, conditions and constraints given to a system, without specifying how to achieve them. The main characteristics of an intent are:

- An intent is typically **understandable by humans**, and needs to be **interpreted by a machine** without any ambiguity.
- An intent focuses on describing **”what” needs to be achieved**, and not ”how” the outcomes should be achieved. This is in contrast to a rule (focus on ”how”) which specifies the explicit logics or formula to be executed, and to a policy (focus on ”what” + ”how”) which specifies the action(s) to be taken and the conditions under which they should be taken.

Intent-Based Management aims to lead towards networks that are significantly simpler to manage and operate, requiring only minimal external intervention. Even autonomic networks are not clairvoyant: they cannot inherently discern operational objectives or determine which service instances they should support. In other words, a network has no intrinsic awareness of the provider’s goals—the intent that gives the network its purpose. These goals must therefore be communicated explicitly as intent. That being said, the concept of intent is not limited just to autonomic networks, such as networks that feature an Autonomic Control Plane [37], but applies to any network. Examples like the hierarchical schema to support different roles related to 5G networks and network slicing management defined in 3GPP, where different intents could be considered, being used for supporting different interactions, specifically:

- **Intent-CSC**: from Communication Service Customer (CSC) to the Communication Service Provider (CSP) to express properties of a communication service, e.g. ‘Enable a V2X communication service for a group of vehicles in specific time with low latency’.
- **Intent-CSP**: from CSP to a Network Operator (NOP) to express properties of the CSP’s desired network, e.g., ‘a network slice supporting V2X communications.
- **Intent-NOP**: from NOP to a Network Equipment Provider (NEP) to express characteristics of a RAN and/or 5GC network, e.g., specifying ‘coverage requirements and UE throughput requirement in certain area’.

Intent can also be used to manage and control of closed-loop automation, which means the intent can be translated to policies and management tasks.

However, intents traditionally overlook the notion of trust, leaving a gap in how systems evaluate the conditions under which actions should be taken.

2.2.1.2 Service Models

A service model represents a service delivered by a network to a user. As defined in RFC 8309 [142], such a model describes the service and its parameters in a portable, implementation-agnostic manner, enabling its use independently of the underlying equipment or operational environment in which the service is realized. Two subcategories are distinguished:

- a "**Customer Service Model**" describes an instance of a service as provided to a customer, possibly associated with a service order
- and a "**Service Delivery Model**" describes how a service is instantiated over existing networking infrastructure.

Realizing service models requires a system, typically a controller, that implements the provisioning logic. This includes decomposing high-level service abstractions into lower-level device constructs, identifying and allocating the necessary resources, and orchestrating the sequence of provisioning actions. Orchestration is generally performed using a "push" model, in which the controller/manager initiates operations, distributes the required configurations to devices, and verifies that the resulting operational or derived states align with the intent/desired state. Beyond initial instantiation, the system must also support updating, modifying, and decommissioning service instances. The device itself typically remains agnostic to the service or the fact that its resources or configurations are part of a service/concept at a higher layer.

Instantiated service models map to corresponding instances of lower-layer network and device models (e.g., specific path instances or particular port configurations). A service model also captures the dependencies and layering of services over the lower-layer networking resources that support them. This facilitates management by allowing to follow dependencies for troubleshooting activities and to perform impact analysis in which events in the network are assessed regarding their impact on services and customers. Services are typically orchestrated and provisioned top to bottom, which also facilitates keeping track of the assignment of network resources (composition), while troubleshooted bottom up (decomposition). Service models might also be associated with other data that does not concern the network but provides business context. This includes things such as customer data (such as billing information), service orders and service catalogues, tariffs, service contracts, and Service Level Agreements (SLAs), including contractual agreements regarding remediation actions. An example of a data model that provides a mapping for customer service models (e.g., the L3VPN Service Model) to Traffic Engineering (TE) models (e.g., the TE Tunnel or the Abstraction and Control of Traffic Engineered Networks Virtual Network model) is the Service Mapping YANG Data Model [92]. Like intents, service models provide higher-level abstractions of network functionality. They are often accompanied by mappings that capture the relationships between service-level constructs and underlying device or network configurations. Unlike intents, however, service models do not define a desired outcome that an Intent-Based System (IBS) can automatically maintain. Instead, their management relies on the design and implementation of sophisticated algorithms and control logic by network providers or system integrators.

Building on the concept of intents, service models require a language, such as the YANG data model, that apart from the network metrics incorporate also the notion of trust.

2.2.1.3 Intent-Based Networking - Functionality

Intent-Based Networking (IBN) ([48]) is a paradigm in which network behavior is guided by high-level objectives, or intents, rather than low-level configurations. In IBN, users or applications specify the desired outcomes for the network, and the system automatically translates these intents into the appropriate network policies, configurations, and actions. This approach abstracts away implementation details, enabling networks to operate with minimal human intervention while maintaining flexibility and adaptability. By focusing on what the network should achieve rather than how to achieve it, IBN supports automated provisioning, dynamic optimization, and more effective management of complex network environments. IBN involves a wide variety of functions that can be roughly divided into two categories:

- **Intent Fulfilment** provides functions and interfaces that allow users to communicate intent to the network and that perform the necessary actions to ensure that intent is achieved. This includes algorithms to determine proper courses of action and functions that learn to optimize outcomes over time. In addition, it also includes functions such as any required orchestration of coordinated configuration operations across the network and rendering of higher-level abstractions into lower-level parameters and control knobs.
- **Intent Assurance** provides functions and interfaces that allow users to validate and monitor that the network is indeed adhering to and complying with intent. This is necessary to assess the effectiveness of actions taken as part of fulfilment, providing important feedback that allows those functions to be trained or tuned over time to optimize outcomes. In addition, Intent Assurance is necessary to address ”intent drift.” Intent is not meant to be transactional, i.e., ”set and forget”, but is expected to remain in effect over time (unless explicitly stated otherwise). Intent drift occurs when a system originally meets the intent, but over time gradually allows its behaviour to change or be affected until it no longer does or does so in a less effective manner.

The key challenge is bridging the worlds of IBN and trust, enabling networks to reason about and act upon trust-aware intents. While the incorporation of trust into intents lies beyond the current scope of CASTOR, the project will develop languages to translate and quantify trust trust as part of the service model, providing crucial stepping stones toward this goal.

2.2.2 Orchestration

The orchestration layer offers great potential to seamlessly manage diverse domains and, due to the wide range of possibilities it provides, it has attracted the attention of the scientific community. In [39], an orchestration model is proposed, focusing on providing high availability for service function chains, where multiple Virtual Network Functions (VNFs) that perform the same function but with different roles (Master or Slaves) are instantiated on distinct servers, generating multiple paths for delivering the same service. Service orchestration in virtualized networks can leverage NFV-MANO framework to automate lifecycle management of network services, from instantiation to scaling and healing. ETSI Network Functions Virtualization (NFV) specifications ([6],[7]) formalize the decomposition of orchestration into NFVO, VNFM, and VIM, enabling control over virtualized resources. In [143] is shown that intent-based orchestration can enrich MANO with semantic statements, allowing optimized service placement and dynamic adaptation under varying conditions. Furthermore, the integration of Application-Layer Traffic Optimization (ALTO) ([133],[122]), which provides abstract network map, cost map (related to performance information and SLA-based metrics), path and endpoints information (such as location, bandwidth, capabilities, resource availability), with NFV-MANO, can enable service orchestration decisions to be informed by real-time, topology- and performance-aware metrics, supporting multi-domain orchestration and SLA compliance [102].

Recently, lightweight forms of orchestration have gained significant attention due to their shorter deployment time and ease of portability and scalability. In [38], a survey on container orchestration is presented, where among the identified gaps are container monitoring capabilities, tools, and autonomous features in orchestration frameworks that enable self-healing and self-optimization. In the study by Salhab et al. [127], an open-source Core Network (CN) was orchestrated using Docker Swarm as the manager. Their approach involved deploying multiple nodes with a load balancer and a floating IP to establish a highly available 5G core (5GC). However, this work primarily leveraged Docker's existing capabilities to demonstrate their applicability in the context of 5G. The orchestration lacked dynamism, and the security measures implemented were rudimentary.

Surveys of distributed systems and container-orchestration platforms consistently highlight key challenges in observability, autonomous operations, and resilience [137], [86]. For example, [33] emphasize the critical need for end-to-end runtime instrumentation to ensure reliability in microservice deployments. Modern stacks, Kubernetes[1], Prometheus[2], and Istio[3], help bridge these gaps. Kubernetes implements declarative configuration and self-healing, enabling systems to detect and automatically recover from failures. Prometheus provides a robust metrics-based monitoring ecosystem, while Istio's service mesh adds secure traffic control, identity-based policy enforcement, and zero-trust architecture principles to the orchestration landscape. At the networking layer, container network interfaces (CNIs) like Calico[4] and Cilium[5] extend orchestration capabilities into Layer 3, supporting granular network policies. Cilium, in particular, leverages extended Berkeley Packet Filter (eBPF) to embed network observability, traffic shaping, and security policies directly into the Linux kernel.

Among recent contributions, Lombardo et al. [98] proposed extending Kubernetes networking with segment routing over IPv6 (SRv6), enabling native support for Traffic Engineering (TE) within the orchestration layer. Their work integrates SRv6 support into the Calico-VPP plugin, introducing a scalable, feature-rich overlay for Kubernetes clusters deployed across distributed multi-datacenter environments. The proposed solution allows dynamic creation and configuration of SRv6 tunnels and supports IPv4 and IPv6 workloads. Importantly, it achieves feature parity with traditional overlays, while enabling advanced capabilities such as multi-tenant VPNs and centralized traffic engineering via Segment Routing (SR) policy injection—either through BGP or Kubernetes-native ConfigMaps. Nevertheless, cloud-based VNF orchestration efforts remain fundamental, as they establish the foundation for NFV orchestration in 5G and B5G networks.

Next generation networks will strongly rely on a multi-stakeholder infrastructure. Seamless resource sharing based on AI automation will enable service interconnection across diverse domains, in the so-called Cloud-in-Continuum. Cooperation between stakeholders strengthens End-to-end (E2E) service delivery by addressing competition and trust models in service provisioning. To this aim, future network establishes well-known technological pillars to fulfill these needs. E2E network slicing, leveraging NFV and SDN paradigms, logically groups and isolates resources and services, while NFV-MANO enables the automation of the NFV life-cycle management. Lastly, DLT-based solutions enhance trustworthiness in information-sharing processes and establish these as three key pillars. However, of the three aforementioned, slicing management through orchestration often relies on human intervention for decision making and responding to context evolution, which reduces its effectiveness. Current State-of-the-Art (SotA) solutions typically address this challenge by introducing a unified orchestration abstraction, such as intent-based interfaces or intermediary orchestrators. These tools translate high-level policies and requirements into actionable commands across diverse domains [56, 59]. To mitigate the limitations of manual interventions, an abstraction layer is proposed as a solution. As a consequence, ETSI defined the Zero-touch network and Service Management (ZSM) standardization working group , oriented toward

[1] https://kubernetes.io/
[2] https://prometheus.io/
[3] https://istio.io/
[4] https://docs.tigera.io/calico/latest/about/
[5] https://cilium.io/

providing autonomous governability of cross-domain service networks, taking humans out of the loop. ZSM is driven by closed loops that coordinate logical steps. The closed loops defined in ZSM enable interaction between system components via intents, aiming to deliver concrete tasks such as slice deployment. ZSM divides the 5G architecture into different Security Management Domains (SMDs), such as RAN, Edge, Transport, Cloud, or Core. Intra-domain management is addressed by the ZSM architecture , which defines functional modules that rationally split functionalities (e.g., orchestration, Policy Management, Analytics). These modules and the domain's infrastructure are connected by a closed loop, driven by an intent. Furthermore, ZSM defines an E2E domain that ensures cross-domain integration. A global perspective of the system allows the E2E domain to manage the lifecycle of E2E slices via intents. They ensure consistent policy enforcement, resource alignment, and coherent lifecycle management. Specifically, intent-based mechanisms enhance interoperability by abstracting technical differences and providing a common semantic language. This approach enables cohesive cross-layer orchestration while reducing the operational complexity often associated with manual or disparate orchestration systems. The SotA techniques leveraging closed-loop automation, driven by intent-based policies, facilitate dynamic and autonomous orchestration that responds in real-time to evolving network conditions. Technologies frequently employed to implement the orchestration across different layers include:

- **Intra-domain Orchestration**: Flexible Algorithm (Flex-Algo), Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), Routing Information Protocol (RIP), etc.
- **Inter-domain Orchestration**: Border Gateway Protocol Link-State (BGP-LS), BGP-LS Extensions for Flexible Algorithm (RFC 9351), Control Exchange Points (CXP), etc.

The orchestration layer offers great potential to seamlessly manage diverse domains and, due to the wide range of possibilities it provides, focusing on providing high availability for service function chains, where multiple VNFs that perform the same function but with different roles (Master or Slaves) are instantiated on distinct servers, generating multiple paths for delivering the same service. Recently, lightweight forms of orchestration have gained significant attention due to their shorter deployment time and ease of portability and scalability.

Many of aforementioned approaches address only specific or isolated aspects, often overlooking the involvement of multiple actors in different domains involved in the holistic management of 5G systems. In contrast, the CASTOR approach bridges these gaps by introducing autonomous, policy-intent-based orchestration. This enables seamless enforcement of security measures and countermeasures, both for standalone VNFs and for E2E network slices spanning the entire 5G ecosystem. Further, there is currently no standardized architecture for integrating service orchestration with security orchestration. While various proposals exist that highlight the benefits and limitations of different approaches, a harmonized methodology at the orchestration layer, capable of managing and providing the necessary evidence to enable trust assessment, has yet to be established. Additionally, it remains an open question how network-related exposure functions or services can be leveraged to disseminate trust-related information effectively.

2.2.3 Routing Protocols and Source Routing in Segment Routing

Virtual Private Networks (VPNs) have emerged as a cost-effective solution to securely connect various Systems-of-Systems among Service Providers in the Computer Continuum . VPNs can be implemented at different layers of the OSI model, resulting in Layer 1 (L1), Layer 2 (L2), and Layer 3 (L3) VPN architectures, with each layer implemented using its own set of protocols [69]. Multiprotocol Label Switching (MPLS) has enabled Service Providers to offer a broad selection of QoS-enabled VPN services [46]. Traditionally, Layer 3 and Layer 2 VPNs were built using Border Gateway Protocol (BGP) and Label Distribution Protocol (LDP) respectively. IS-IS or OSPF served as the Interior Gateway Protocols (IGPs)

to ensure fast next-hop convergence and LDP, BGP or Resource Reservation Protocol for Traffic Engineering (RSVP-TE) for transport label distribution. The number of different protocols and the emergence of Software-Defined Networking (SDN) has led to the development of Segment Routing (SR) defined in RFC 8402 [65].

Segment Routing (SR) introduces a source-routed forwarding paradigm in which the ingress router encodes the end-to-end path as a label or segment stack and is available in two main data-plane instantiations: SR-MPLS and SRv6. IGPs have been extended to carry segment (label) information in Link State Advertisements (LSA), eliminating the need for separate protocols such as LDP and RSVP-TE for label distribution. In a single-domain, the ingress router can compute the full segment/label stack to encapsulate and transport VPN traffic by itself, since it has a complete view of the topology, while in multi-domain or hierarchical networks, the ingress router may not have visibility of all destinations. In a nutshell, with SRv6 route summarization and redistribution can be used to reach destinations, while with SR-MPLS an external helper like an SDN controller or Path Computation Element (PCE) is needed to compute paths and segment/label stacks. The SR PCE receives topology state from IS-IS, OSPF, or BGP Link State Address Family. The Segment Routing Policy Architecture documented in RFC 9256 [64] introduces full Traffic Engineering capabilities to Segment Routing (SR-TE). However, all aforementioned works do not consider trust for routing. Recent research emphasises that the next major step in trust governance involves shifting from node-level trust to path-level trust. Systems such as ICING [108] introduce cryptographic path-verification mechanisms, where packets carry proofs that they have traversed approved paths. FABRID [87] extends this with remote attestation in inter-domain routing, demonstrating that trust evidence can be exposed and used for trustworthy path selection. Architectures such as SCION [45] and ongoing work in the IETF NASR group further demonstrate the need for verifiable, attested forwarding paths that allow clients to constrain their traffic to devices and domains they trust, while SCION [45] is the first one that considers the notion of trust. Since, SCION is the only initiate that solves the same problem as CASTOR, a comprehensive evaluation in terms offered functions will be done in the context of D5.1.

These approaches reveal that most routing infrastructures lack strong mechanisms for path-level trust, reinforcing the necessity of trust governance and trust-plane mechanisms for enabling secure interactions across heterogeneous and dynamic compute-continuum environments. CASTOR is complementary to SCION, focusing on the path-level trust and on how to exchange trust-related data between entities in intra- and inter-domain scenarios to facilitate the construction of a trust-plane.

2.2.4 Dynamic Trust Assessment and Governance

In the complex and heterogeneous compute continuum, comprising vast numbers of devices and diverse data sources, the integration of trust-assessment mechanisms is crucial. In this context, trust assessment mechanisms play a central role in not only establishing and quantifying the extent of trust existing between entities, based on both their own opinions as well as the opinion of others without a central authority, but further enabling the provision of trust evaluations even for devices that lack inherent trustworthiness. Achieving this demands a multifaceted approach that uses a wide range of properties and criteria, with trust relationships grounded primarily in integrity-related attributes. As the compute continuum ecosystem becomes increasingly interconnected, two key intricacy factors further underscore the importance of dynamic trust assessment: the sheer volume of incoming data and the potential conflicts and inconsistencies within this data. Together, these challenges highlight the critical role of adaptive trust-assessment mechanisms in managing data flow and ensuring the reliability and safety of advanced compute continuum ecosystems.

Understanding, classifying, measuring, and assessing trust have been fundamental research challenges in trust management in the last twenty years. Trust assessment has a focal point in sociology, technology, and computing, including e-commerce, access control, and security risk analysis. The authors in [88] propose a trust-assessment method for cloud ecosystems in which multiple Cloud Service Providers (CSPs)

collaborate. In their approach, each CSP forms a subjective trust opinion of its peers based on Service Level Agreements and established reputation, enabling it to select appropriate partners. Similarly, the work in [68] presents a trust-assessment approach for vehicle platooning, where a host vehicle evaluates the trustworthiness of its predecessor by comparing its sensor-derived position with the contents of received V2X messages over time; the resulting trust opinion is then used to adjust the safety distance. In a different vein, the authors in [44] apply epistemic logic to quantify trustworthiness in multi-agent systems, with an observer monitoring agent behaviour and updating trust opinions accordingly. Across all these works, trust is evaluated between two directly communicating peers, meaning trust is assessed only within the scope of a specific bilateral relationship. However, this is insufficient in the compute continuum, where trust must be inferred over complex trust networks composed of numerous interdependent relationships.

Emerging scenarios and applications across various domains, including compute continuum systems, are introducing new challenges for trust-assessment research. At the same time, modern systems are becoming increasingly interconnected and collaborative, forming complex Systems-of-Systems that enable capabilities no system in isolation could achieve on its own. In response, new and more sophisticated requirements for trust assessment must be considered to adequately capture the characteristics of such complex and heterogeneous environments.

First, due to the interconnected and collaborative nature of these systems, trust can no longer be assessed solely within a single trust relationship (i.e., between one trustor and one trustee). Instead, trust must be evaluated across complex trust networks consisting of multiple interdependent trust relationships. Here, a trust relationship refers to the construction of atomic propositions expressing whether relevant trust properties (e.g., integrity, safety, resilience) hold for that specific interaction. Assessing trust across networks enables entities to base trust decisions on input from multiple cooperating peers, increasing confidence compared to assessments derived from only a single source.

Second, trust levels must be derived from incomplete and subjective information, often provided by entities that may themselves be unreliable. Evidence may be unavailable in a direct trust relationship, or such a relationship may not exist at all. Consequently, trust assessment must support transitive trust, obtained through a single or a chain of referral relationships within a trust network. The works in [131] and [82] similarly distinguish between direct, indirect, and transitive trust, noting the importance of transitivity when direct trust is absent.

Third, compute-continuum systems are highly dynamic, with trust relationships forming and dissolving at run time. Thus, trust models must allow trust relationships between entities to be created and removed on-the-fly to reflect this dynamism. Several studies and white papers have highlighted the need for such dynamic trust, integrating temporal aspects into trust assessment [131][63].

Fourth, the entire trust assessment process should be fast and robust to be used at run-time for real-time and safety-critical applications.

Fifth, trust assessment must cover not only entities (nodes) but also the data they produce. This requires support for both node-centric and data-centric trust relationships, enabling a more comprehensive evaluation of trustworthiness.

Sixth, complex trust networks consist of heterogeneous nodes whose trustworthiness depends on multiple, evolving sources of evidence. Trust assessment must therefore incorporate inputs from diverse trust sources, which may change over time. Misbehaviour in this context typically involves the transmission of incorrect data (e.g., incorrect position information), meaning the veracity of data plays a central role. Based on detector outputs, a node's trustworthiness may be increased or decreased accordingly.

Seventh, because compute-continuum systems are distributed and ubiquitous, and aligned with Zero-Trust principles, centralised trust-assessment solutions are insufficient. Instead, decentralised and distributed approaches are required, inherently introducing subjectivity into trust assessment. Subjective trust has been previously acknowledged in works such as [66] and [82]. Therefore, trust assessment

must incorporate belief ownership, allowing subjective beliefs from multiple entities to be merged, producing a more accurate reflection of the objective world than any single opinion could provide. This aligns closely with the first requirement, where we emphasise the cooperative construction of trust across multiple entities.

There are various mathematical theories that have been proposed in the literature to ascertain information in uncertain and unpredictable conditions that could be potentially used for trust assessment: Probabilistic Logic, Fuzzy Logic, Bayesian Probability, Dempster-Shafer Theory and Subjective Logic [54]. In essence, Subjective Logic uniquely provides the ability to integrate subjective beliefs from multiple agents, fuse contradictory evidence, and apply trust transitivity, allowing trust to be evaluated across entire trust networks rather than individual pairwise relationships.

2.2.5 Risk Assessment and Required Trust Level Calculation

In the computing continuum, spanning IoT devices, edge nodes, and cloud infrastructure, ensuring secure and trustworthy communication is critical for dynamic, heterogeneous networks. The risk assessment and the calculation of the required trust level (RTL) are critical to securing the computing continuum, where dynamic and heterogeneous networks of IoT devices, edge nodes, and cloud systems demand robust security mechanisms.

Risk assessment identifies, analyzes, and mitigates threats to data and infrastructure, while RTL calculation quantifies the minimum trust needed to meet application-specific security, privacy, and operational requirements, often formalized in Security Service Level Agreements (SSLA). Existing risk assessment approaches often lack adaptability to real-time, multi-domain environments, and trust calculation methods focus narrowly on integrity, overlooking resilience and privacy. Similarly, trust-level calculations often focus narrowly on integrity, neglecting other properties such as resilience and availability.

In distributed systems, risk assessment, involves identifying threats, vulnerabilities, and their impacts, but scalability challenges arise in large, dynamic networks like IoT and edge computing, where device mobility and heterogeneous protocols increase complexity. Real-time risk assessment is limited by the need for continuous monitoring across diverse domains. Trust-level calculation employs methods like Bayesian networks and Subjective Logic to quantify trust based on evidence such as device behaviour or security claims. Existing risk assessment methods lack real-time adaptability to the dynamic nature of the computing continuum, where node mobility and cross-domain interactions require continuous threat assessment. The integration of risk assessment with trust quantification for path selection is limited, as most approaches do not address the varying SSLA trust requirements.

The definition of RTL threshold, along with a consistent approach to risk assessment, forms the foundation of the overall assessment methodology. A key challenge lies in translating the semantics of identified risks into an RTL expression that can be directly compared against the actual level of trust (ATL). This gap, how to map risk-derived requirements into measurable trust levels, is well-recognized in other domains, such as the automotive sector [11]. In the context of 5G networking, threat modelling is inherently complex due to the presence of multiple layers, such as the routing, data, and link layers, each with its own distinct threat landscape. Although definitions for these layer-specific threat models exist, they also serve as a foundation for deriving RTL.

Performing trust assessments within this multi-layered and highly dynamic environment requires a generalized methodology capable of consistently identifying RTL values and enabling meaningful comparisons with ATL. Developing such a methodology remains challenging, as several obstacles must be addressed, including the diversity of threat models, heterogeneity of network components, and the evolving nature of 5G architecture and services.

2.2.6 Establishing Trust in Network Devices with Secure Runtime Monitoring

CASTOR's goal is to provide trusted path routing enforced by a set of trustworthy network devices (e.g., physical or virtual routers). Therefore, CASTOR aims at enabling the network domain orchestrator to dynamically assess the trust levels of all network devices within the domain to define trusted paths, adjusting them whenever a trust level has changed. In order to do so, CASTOR needs to establish trusted components within each device which allow for secure runtime monitoring and attestation of the device states, as a basis for the trust assessment. Furthermore, CASTOR's trusted device components need to securely share the local trust levels (and associated attestation evidence) with the network orchestrator to enable a network-wide trust assessment and calculation of the trusted forwarding paths.

Establishing Trust in Network Devices. The IETF Trusted Path Routing (TPR) standard [27] proposes to establish trust in network devices by incorporating Trusted Platform Module (TPM) crypto (co-)processors into each device. That way, verifiers can leverage TPM-based remote attestation protocols to verify the boot-time state of each router before including it in the network domain for trusted path routing. However, in contrast, CASTOR envisions to perform a dynamic trust assessment based on runtime attestation evidence generated at each device.

Alcatraz [21] and TrustedGateway [129] establish trust in network routers by augmenting them with CPU-based trusted execution environments (TEEs), such as Intel SGX or Arm TrustZone, in order to enable secure traffic routing and forwarding. Alcatraz establishes end-to-end-encrypted (E2EE) tunnels between the TEEs of each router for secure hop-to-hop traffic encryption, while TrustedGateway designs a trusted network I/O path with secure routing, firewalling, and NIC interaction that is isolated from other services running on the routers. However, neither of them performs secure runtime monitoring of the router's components to collect evidence for a dynamic trust assessment in the context of trusted path routing. Furthermore, they do not consider a network orchestrator performing a network-wide trust assessment and path calculations.

The TDISP protocol [2] of the PCI-SIG standard allows a TEE VM (TVM) to establish trust in PCIe devices and include them in their TCB, enabling secure interaction. More precisely, TDISP binds TEE Device Interfaces (TDIs) of an attested PCIe device (e.g., GPU, NIC, or SSD), e.g., representing virtual functions (VFs) of the device, to a TVM and continously monitors the TDI's configuration and state information to guarantee secure operation. That way, TDISP enables secure interaction between the TVM and TDI and can revoke the binding whenver the TDI enters and insecure state. In CASTOR, we want to bind network devices to the trusted domain orchestrator and continuously monitor them in order to asses their trust levels based on runtime evidence. However, TDISP is applicable only to PCIe devices and therefore cannot be directly applied to remote devices interfacing via the network (e.g., routers).

CASTOR envisions to address the above challenges of establishing runtime trust into network devices. CASTOR designs new TEE-based trust extensions, called trust network device extensions (TNDE), that are securely instantiated in each network device taking part in the trusted path routing. The TEE-based design roots the protection of the extensions in hardware and enables the secure execution of a runtime monitoring and trust assessment framework locally on each network device—strongly isolated from the remaining device components (e.g., router network OS). To form a trusted network domain, CASTOR envisions to transfer and extend the concepts of TDISP beyond PCIe devices towards network devices managed by a trusted orchestrator, in the context of trusted path routing. The CASTOR device extensions expose so-called trust network device interfaces (TNDIs) that represent CASTOR-enabled network devices (e.g., a physical or virtual router). The orchestrator can remotely attest the device extensions as part of a secure enrolment process into the network domain and use a new TNDI security protocol (TNDI-SP) to set up the dynamic runtime monitoring and trust assessment of each network device. A detailed description of the CASTOR architecture will

be provided in chapter 6.

Runtime Monitoring of Device Evidence. CASTOR needs to perform secure runtime monitoring of each network device to collect evidence for the dynamic trust assessments. Many relevant tracing frameworks for system monitoring have been proposed in the context of host-based intrusion detection and malware analysis, including kernel-based, hypervisor-based, as well as hardware-assisted tracers. In addition, control flow attestation schemes (CFA) have proposed static or dynamic instrumentation of target services for fine-grained control flow tracing at a basic block level in embedded systems.

Kernel-based approaches [90, 1, 3] typically introduce new kernel modules and/or build on pre-existing kernel facilitates (e.g., Linux tracepoints or eBPF) to monitor process interactions and resources. Examples for monitored artifacts include system calls, I/O resources (sockets, files), and process memory regions. The tracers' direct integration into the kernel isolates them from user space attackers and provides direct access to the kernel data structures. However, they are not protected against a kernel compromise and must be tightly bundled with the vendor specific OS, which can be challenging in setups with devices from multiple vendors, as in CASTOR.

Hypervisor-based introspection frameworks [67, 147, 130] allow for a memory-based inspection of virtual machines (VMs) or a single virtualized host OS. VMI frameworks are strongly isolated from the target even under a kernel compromise and enable the non-intrusive inspection of kernel and user data structures by leveraging existing memory forensic techniques [96]. However, as VMI inspects the target from outside (in contrast to kernel-based tracers), VMI-based tracers must bridge the semantic gap [79] to correctly interpret the inspected memory and locate relevant data structures, as well as avoid performance and security pitfalls, especially when targeting live systems.

Hardware-assisted designs are often based on CPU extensions or Direct Memory Access (DMA)-capable peripherals [70, 113] which allow for high-performance tracing. These approaches are typically tailored to a specific type of inspection data (e.g., DMA-accessible memory or execution traces) and—depending on the tracer design—can face similar deployment and security challenges as presented above.

CFA schemes choose between different types of tracers to monitor the control flow of the target [14, 19]. Typically, they apply static or dynamic instrumentation of the target service or leverage CPU extensions (cf. above) to trace the control flow at a basic block level (BBL). However, due to the performance impact of BBL-level tracing and the associated deployment challenges of static instrumentation, CFA schemes typically focus on a small number of user space applications and embedded systems without high performance requirements.

To the best of our knowledge, no existing work has specifically looked into the challenges of performing secure and efficient runtime tracing of network devices—especially, in the context of trusted path routing. Therefore, CASTOR aims at providing a secure runtime tracing layer as part of the device extensions. CASTOR will tailor the tracing to specific evidence relevant to dynamically monitor and asses the trustworthiness of the network devices for the enforcement of trusted routing paths. CASTOR plans to follow a secure multi-level tracing approach by exploring a TEE-isolated memory inspection, as well as kernel or hardware-assisted tracing methods that can provide complementary tracing data for a more fine-grained assessment on demand. Furthermore, CASTOR will address the challenge of securely sharing the trust levels and evidence via the network with the global orchestrator for a network-wide trust assessment. Therefore, CASTOR will build on concepts of TDISP and extend them towards network devices managed by the domain orchestrator, as mentioned above. We will describe CASTOR's tracing framework as part of the architecture in chapter 6.

Runtime attestation and implicit verification. As already mentioned, continuous verification of device and data trustworthiness is important to prevent unauthorized access and ensure reliable communication

across domains. Composite attestation is a concept in digital identity and verification systems that allows multiple attestations (verifiable claims or credentials) to be combined, reused, and built upon to create new, complex attestations. Specifically, an attestation is a digitally signed statement about a subject (person, organization, or thing) made by an issuer, while ”composability” means these attestations can be (a) combined with other attestations to form more comprehensive credentials; (b) selectively disclosed (revealing only certain parts); and (c) derived to create new attestations without requiring the original issuer. From the aspect of cryptography, in composite attestation, (a) there are multiple provers and multiple verifiers; (b) multiple proofs can be aggregated/combined one by one; and (c) any verifier can verify part of the aggregated/combined proofs.

In parallel, an aggregatable signature can act as a credential for a user, which can be verified and offers unforgeability. Most importantly, multiple aggregatable signatures associated with multiple users can be combined as one signature, proving identities of all corresponding users. There are some possible solutions. The first one is Boneh–Lynn–Shacham (BLS) signature [32], which is based on type III pairings and has short signature size, efficient signing and verification. The second one is using a structure called Merkle tree [106], each user's credential (a normal signature) is considered as a leaf of the Merkle tree. The Merkle tree is maintained by a trusted third party, which can aggregate multiple signatures. Any verifier can do the verification by checking certain authentication path in the Merkle tree.

CASTOR's runtime attestation will be based on aggregatable signatures to achieve composite attestations. Also, considering we are in a transition from traditional cryptography to post-quantum ones, we plan to adopt a post-quantum signature algorithm with post-quantum zero-knowledge proof techniques, which allows continuous proof/verification (recursive proof) in zero-knowledge.

State machine and evidence verification. State-machine formalisms (finite-state machines—FSMs, labelled transition systems—LTS/IOLTS, communicating FSMs—CFSMs) remain the canonical abstraction for modelling network protocols and distributed communication services. Protocol roles (endpoints, routers, controllers) are captured as communicating automata exchanging typed messages, while safety/liveness properties are expressed in temporal logics (LTL/CTL) and verified via model checking [119], [47], [75], [34]. For time-critical behaviours (retransmission timers, heartbeat/keep-alive, exponential backoff), timed automata extend FSMs with clocks and guards, enabling exhaustive analysis with tools such as UPPAAL [18], [25]. In practice, engineers combine IOLTS/ioco conformance theory for specification-based testing, PROMELA/SPIN for asynchronous message-passing verification, and, where appropriate, SDL/MSC or LOTOS-style notations for protocol structure and interactions, to systematically explore race conditions, deadlocks, livelocks, and timeout pathologies under adversarial schedulers [47, 75] [136].

At the same time, complementing white-box modelling, FSM learning could provide black-box models of protocol implementations that are otherwise opaque (closed-source stacks, firmware). Active automata learning—rooted in Angluin's L*—interacts with a system under learning via membership/equivalence queries to infer minimal DFA/Mealy models, with counterexample-guided refinement; open-source frameworks such as LearnLib operationalise these algorithms for I/O-rich systems via abstraction/caching/mappers to manage real data domains and nondeterminism [20], [112], [78]. In parallel, passive approaches learn Moore (or Mealy) machines from input–output traces without oracle queries, yielding artefacts well-suited for conformance testing and downstream verification of protocol I/O behaviour [72]. Pairing learned models with ioco-style testing, model checking and runtime validation has exposed undocumented states and state-dependent vulnerabilities in production protocol stacks (e.g., TLS), demonstrating the practical security value of the learn-then-analyse pipeline [72], [53].

Recent work tightens the loop between learning and verification evidence through learn→verify→refine workflows: a learned model is checked for temporal properties (e.g., trust atomic propositions). Emerging directions include probabilistic/stateful learning (towards MDPs or stochastic I/O automata for congestion

control and wireless MACs), timed learning (recovering clock constraints to reason about timeouts and jitter), and hybrid approaches fusing active learning with fuzzing or symbolic execution to accelerate coverage of hard-to-reach protocol states. In safety-critical networking contexts, these techniques enable evidence-driven assurance cases over both (a) hand-crafted specifications and (b) learned surrogates of flight-critical communication stacks, with conformance and regression suites guarding against regressions introduced by performance optimisations or hardware offloads [47], [18], [78], [72].

Moore machines will be employed in CASTOR, with a focus on abstracting the behavior of communication nodes at the local level, and the global as part of CASTOR orchestration engine.

2.2.7 Support for Global but Heterogeneous Trust

CASTOR's goal is to establish trusted path routing in a trustworthy network, in which there are a set of devices, e.g., physical or virtual routers. To achieve trusted path routing, in one domain, every router needs to generate a proof of their evidence (trust level) to next router *in sequence*. An orchestrator (server) in one domain needs to collect all routers' proof to generate one proof on behalf of all routers. In this process, CASTOR needs to ensure all aggregated routers are valid or can find some rouge routers. Therefore revocation or linkability is necessary. Based on this kind of requirement, a composite attestation scheme is required. For the cross-domain setting, during achieving the composite attestation scheme with revocation, CASTOR also needs to compare different trust levels and give the lowest one without revealing the real values of trust levels. A crypto primitive, called Order-revealing encryption (ORE) can achieve orders of plaintexts by comparing corresponding ciphertexts without revealing the real values of plaintexts. There are numerous solutions to achieve composite attestation and order-revealing encryption schemes.

2.2.7.1 Public Key Infrastructures (PKIs) and Beyond

The correctness and reliability of attestation evidence is a fundamental step and a key enabler for the envisioned trusted path routing and traffic engineering applications, as it ensures a certain level of security and trust through authenticity among the network elements.

Traditional Public Key Infrastructure (PKI) serves as the foundational layer of trust for the entire CASTOR project by establishing a secure and verifiable link between public keys and the real-world identities of all entities involved, including physical or virtual routers and domain orchestrators. This is achieved through the issuance of digital certificates by a trusted Certificate Authority (CA), which cryptographically binds each device's identity to its public key. The use of digital certificates is used in order to authenticate routers and preventing form attacks. Standardized architectures, such as those outlined by the IETF's Remote Attestation Procedures (RATS), further build upon this concept by mandating PKI-signed platform certificates. These certificates not only authenticate a device's identity but also anchor it to a trusted computing base, creating a robust framework for initial device registration and verification. This mechanism of trusted identity bootstrapping is a critical prerequisite for all subsequent secure operations within the CASTOR ecosystem.

In the PKI approach, a set of certification authorities (CAs) provide credentials to the network elements. In the general case, there is a set of different authorities with distinct roles:

- **Root Certificate Authority (RCA):** Serves as the trust anchor of the PKI and is responsible for issuing certificates to subordinate CAs. Its own certificate is self-signed.
- **Enrolment Certification Authority (ECA):** Handles the registration of network elements and issues long-term enrolment certificates. Entities holding such certificates may then request additional certificates from other CAs, such as pseudonym certificates from the PCA.

- **Pseudonym Certification Authority (PCA):** Issues certificates that contain no identifying information, thereby supporting privacy-preserving operations.
- **Certificate Revocation CA (CRL-CA):** Responsible for generating and distributing certificate revocation lists for all types of certificates.

However, due to the sensitivity of attestation evidence, it raises the need to protect privacy as well. Current approaches are based on PKI-based solutions with privacy friendly authentication services through the use of short-term pseudonyms. However, such architectures are based on centralized infrastructure entities for the support of services such as authenticated router registration, certificate revocation, etc. Thus, traditional centralized PKI solutions are insufficient, as they fail to capture the decentralized Self-Sovereign Identity concepts across all layers that are necessary to realize the vision of a network of trust.

2.2.7.2 Layered and Composite Attestation

In order to have guarantees for trust on the routers and devices there is the concept of confidential computing. Although numerous research efforts have explored remote attestation, an open challenge remains in the assumptions we provide to the verifier (e.g., expected system state). Existing schemes are based on the concept of a single prover and a single verifier. However, in the context of routing process this needs to be elevated to multiple proves and multiple verifiers, either why the infrastructure element has multiple internal building blocks (e.g., process, VNFs) or a path that is constitute of multiple routers. For instance, in CASTOR, each device, whether a physical or virtual router, can generate evidence and present it to the next node. The receiving node verifies the evidence, appends its own proof (which incorporates the previous node's proof), and forwards it further. This process can be repeated hop by hop, ultimately establishing a trusted and authenticated path in which the order of devices is preserved. Additionally, the system must be able to identify and isolate any rogue device. From a cryptographic perspective, this scenario leads to several key requirements: (a) multiple provers must be supported; (b) each prover must be able to combine several proofs; (c) combined proofs must implicitly encode and allow verification of the sequence of provers; and (d) rogue nodes must be detectable, meaning effective revocation mechanisms must be in place. Thus, is clearly understandable that there is a need to shift from the concept of single attestation to the concept of layered and composite attestation.

Layered attestation focuses on capturing evidence from the different layers within a single device (e.g., the internal layers of a virtual router), whereas composite attestation addresses scenarios involving multiple provers and verifiers, such as several routers forming a network path. However, layered attestation does not account for runtime behavior, and research on composite attestation is still at an early stage. To elevate layered attestation into a full composite attestation model, additional dimensions, such as the ordering and aggregation of evidence across entities, must be captured.

More specifically, when attestation involves multiple provers and requires aggregating proofs across them, layered attestation offers a suitable approach. Layered attestation is a security paradigm that constructs a composite, verifiable proof of integrity for a complex system by sequentially aggregating evidence from its individual components or layers. Rather than producing isolated, independent reports, it creates a chain of trust, where each layer—such as a bootloader, operating system, application, or in CASTOR's case, a network router—attests to its own state and then cryptographically incorporates the proof received from the previous layer. This produces a single cumulative proof demonstrating not only that each component is trustworthy on its own, but also that the components operate together in the correct sequence and configuration. A final verifier receiving this layered proof can therefore validate the integrity of the entire chain, from the root of trust to the final layer. This makes layered attestation well suited for verifying multi-step processes such as secure boot sequences or, in CASTOR, establishing a trusted path across multiple routers [125, 145, 91, 121, 99, 144, 148].

In addition to layered attestation, ordered multi-signatures [30] provide another cryptographic primitive that can achieve composite attestation. An ordered multi-signature scheme extends traditional multi-signatures by enforcing a specific sequence on the signing process. In a standard multi-signature, a group of signers collectively produces a single compact signature on a shared message, and the order in which they sign is irrelevant. By contrast, an ordered multi-signature scheme cryptographically encodes and verifies the exact sequence in which each signer contributed. The result is a single, constant-size signature that proves two things simultaneously: (a) all required signers endorsed the message, and (b) they did so in one—and only one—specific order. This property, known as signer order integrity, is essential in scenarios where the sequence of actions or approvals reflects a workflow, logical process, or physical path, such as a network routing procedure in which data traverses routers in a predetermined order (defined path).

The technical mechanism to achieve this often involves sequential aggregation. The process begins with the first signer in the sequence generating an initial signature. This signature is then passed to the second signer, who does not merely create their own independent signature but instead performs an aggregation operation that cryptographically combines the first signature with their own, creating a new aggregate signature. This new aggregate is then passed to the third signer, and the process repeats. Critically, each signer's operation is dependent on the aggregate signature created by the immediate predecessor. This chain of cryptographic dependencies makes it computationally infeasible to reorder the signers after the fact or to generate a valid final signature without having followed the exact prescribed sequence. After [30], there are some work focusing on ordered multi-signatures [134, 24]. [134] proposed a more efficient ordered multi-signature scheme. However, the state-of-the-art scheme [24] is more efficient that during verification no pairing operations. But in this scheme, an aggregated signature over different users is signed on the same message, which is not suitable for CASTOR scenario. Moreover, these schemes do not involve revocation function, i.e., finding a rogue user whose proof was aggregated in a signature. The target for CASTOR is to design a more efficient layered attestation scheme or ordered multi-signature with revocation. Then apply it to CASTOR.

Beyond the schemes used to generate verifiable evidence for trust assessment, the output of attestation must also be shared in a privacy-preserving manner. During cross domain communication, each router creates a piece of self-attested evidence, which will be used to compute the trustworthiness score either 1/0 (indicating trusted/untrusted) or a percentage (indicating a certain trusted level). The router encrypts its level of trust, which could involve its identity, public key associated with a certificate, attestation evidence, network position, and more. To join a network or to confirm its existence in the network but protect its trusted level, the router sends the ciphertext to a domain orchestrator. In a domain, the domain orchestrator can access to all trust levels associated with all nodes in this domain. The orchestrator should be able to compare different trust levels and give the lowest trust level. As discussed in the previous section, traditional PKI cannot achieve this. Order revealing encryption (ORE) schemes, however, can enable the exposure of attestation evidence and trust semantics in a privacy-preserving way, for example, by revealing only a minimum trust level without disclosing any additional sensitive details.

ORE is an encryption scheme that shows the order of plaintexts via comparing the ciphertexts without revealing information about plaintexts. ORE was introduced by Boneh *et al.* [31], aiming to improve order-preserving encryption, which suffers from inference attacks. However, this scheme [31] is impractical because it is based on multilinear maps. After that, Chenette *et al.* [43] proposed the first ORE based one pseudorandom function (PRF), which is efficient and achieves a simulation-based security notion. However, this scheme revealed the first differing bit, which is the most significant differing bit. This can lead to security issues to the ORE scheme. Thereafter, there are some ORE schemes from PRF, aiming to improve efficiency and security [93, 101, 117]. Further, there are some ORE schemes achieving some features, such as delegation [95]. Nevertheless, how such mechanisms can be integrated as extensions into existing network exposure services remains an open research question.

Overall, existing attestation flows assume a relatively stable model in which the Verifier initiates evidence collection, evaluates it against a fixed set of reference values, and produces an Attestation Result. This model presumes a single moment of assessment, uniform evidence semantics, and a clear Verifier-Attester separation. Introducing runtime trust monitoring exposes a broader space of design questions that challenge these assumptions, as already identified and highlighted by CASTOR in IETF TPR discussions [76].

Heterogeneous and Weighted Evidence: One unresolved issue concerns the nature and diversity of evidence during runtime. Devices may include multiple sources of evidence related to runtime state, including integrity monitors, process isolation mechanisms, or configuration compliance checkers. These sources may differ in reliability, frequency or precision. Therefore it cannot be assumed that all evidence is of equal weight. Some measurements may be conclusive, while others may be advisory or context-dependent. So we need to extend current attestation frameworks to represent or process such weighted, multi-source evidence over time.

Evidence Change and Notification Models: Dynamic trust monitoring also means that we need to address how changes in evidence during runtime should be handled. Devices may transition between trust-relevant states, and a model, where the Verifier initiates attestation on demand, offers no way to detect or respond to such transitions in a timely manner. A way forward is to define a mechanism for an Attester to track internal evidence changes, determine when a new trust assessment is needed and notify a Verifier accordingly. This also raises architectural questions about how notifications are structured and secured, and how Verifiers might subscribe to receive updates tied to evolving evidence. In scenarios such as Trusted Path Routing, the Verifier might even reside at the orchestration layer in order to receive notifications from multiple routers and dynamically recompute trusted forwarding paths when device trust conditions change.

Source-Level Validation and Binding: We also need to define how a Verifier should validate the origin and binding of individual evidence components when they are collected from multiple sources within an Attester. Each evidence source may operate under different trust boundaries and may require individual validation with respect to its provenance, protection domain, and association with the Attester's identity. This creates a need for mechanisms that allow internal components to be explicitly and securely bound to the attestation process. In practice, this implies cryptographic binding between evidence sources and the attestation function, supported by key management and endorsement models that enable composability and structured verification.

2.2.8 Service Certification and Auditing through Blockchain Infrastructure

The convergence of Blockchain technology with the compute continuum constitutes a promising paradigm for service certification and auditing. In the context of networking, Blockchains can support capturing and managing violations in SLAs/SSLAs, acting as a bridge for information exchange, in a controlled manner, between different administrative domains. However, Blockchains lack built-in capabilities to retrieve data from or send data to external systems. This limitation is known as the Blockchain oracle problem, which highlights the inability of smart contracts to verify the veracity of off-chain information [58]. A key challenge in this domain is bridging the trust gap between the on-chain, decentralized ledger and the off-chain, real-world data. This is where a Blockchains oracle plays a substantial role, acting as the trusted intermediary to store authentic data into the Blockchain [115]. In the context of service certification and auditing, the role of the Blockchain oracle is not only to transfer data, but to also guarantee its origin, timeliness and immutability. The oracle's role is to bring proof of service execution onto the Blockchain, while any entity with the appropriate access rights can audit the service's performance by examining the on-chain records. The Blockchain acts as a single source of truth, where the history of all certified services is permanently recorded. The principles of Blockchain oracle and auditing are particularly relevant to secure path routing in the compute continuum. A relevant academic work by Ghodichor et al [71] demonstrates how a blockchain-based approach can authenticate nodes and improve network security by creating an

immutable ledger of routing information.

However, traditional centralized oracles are attractive targets for attackers. Decentralized oracles mitigate the single point of failure inherent in centralized oracles, yet they often suffer from performance limitations [42]. In this regard, researchers are focusing on different mechanisms, to build decentralized oracles, but there is still room for improvement in terms of trustworthiness, efficiency, and privacy [23]. Thus, trusted hardware technologies (e.g., Intel SGX or ARM TrustZone) are integrated with decentralized oracle systems to form a more robust and secure "trusted oracle" protocol. Prominent examples are the Phala Network [110] and the Fabric Private Chaincode [35]. Moreover, secure oracles are incentivized to provide accurate data through a reputation system and staking mechanisms. For instance, oracles that submit truthful data earn rewards and build reputation, while those that submit malicious or inaccurate data lose staked collateral. Thus, it is ensured that the data an oracle feeds into the Blockchain is highly resistant to manipulation, thereby establishing a foundation of trust for subsequent auditing processes. However, what is still missing is the concept of a secure oracle layer to expose all the interfaces and capabilities independently from the underlying technology to manage all the trust related information data set.

The emergence of Blockchain technology has created new possibilities for secure and decentralized service certification and auditing. However, as aforementioned, several challenges still need to be addressed. Towards this direction, secure decentralized oracles, especially when combined with trusted hardware, provide a promising pathway for service certification and auditing. On top of that, CASTOR will offer a secure oracle layer exposing all the necessary interfaces and functionalities. Such a layer is essential for managing the complete set of trust-related information in a consistent and technology-agnostic manner.

2.2.9 Complex Multi-Constraint and Multi-Objective Optimization Process

The proliferation of edge computing and the Internet of Things (IoT) has led to the emergence of highly distributed, dynamic, and heterogeneous networks. These networks consist of resource-constrained devices that operate under stringent application-level requirements, including low latency, energy efficiency, and secure, trustworthy communication. These devices often function in dynamic environments, where they may be mobile or communicate without pre-existing infrastructure, thereby forming Mobile Ad Hoc Networks (MANETs). They engage in communication using various radio technologies such as Bluetooth (IEEE 802.15.1) and ZigBee (IEEE 802.15.4). Additionally, they may connect to the Internet via cellular technologies (e.g., 4G/5G) or Wi-Fi (IEEE 802.11), collectively contributing to the paradigm known as the IoT. IoT further exacerbates the challenges of network due to the massive scale of interconnected devices and data. These networks are inherently multi-layered and span multiple Autonomous Systems (ASs), comprising a wide range of devices that employ diverse technologies and operate under varying domain-specific policies. This complexity gives rise to both intra-domain and inter-domain routing challenges. Existing technologies, such as LEACH [84], provides foundational mechanisms for establishing topologies within an AS, including the selection of cluster heads (CHs) that act as representatives and communication gateways to external networks on behalf of member nodes. Inter-domain routing involves the integration of topology, routing, and policy information across multiple ASs. Modern network architectures primarily rely on distributed best-effort protocols such as BGP [123], offering limited control over end-to-end (E2E) traffic. However, traffic can also be routed based on Quality of Service (QoS) requirements, supported by technologies such as Multi-Protocol Label Switching (MPLS) [124].

In the context of path establishment and TE process currently the focus is in single-objective optimisation which is generally a network metric (e.g., latency). Integrating trust as a foundational component in IoT networks necessitates the exploration and development of novel, trust-aware network architectures. Regarding multi-objective optimistaion, techniques like Multi-Objective Dijkstra have been adapted to compute Pareto-optimal paths, while meta heuristic algorithms such as Ant Colony Optimization (ACO) and

Genetic Algorithms (GA) have been used for multicast routing and dynamic network scenarios [128] [118] [73] [103]. More recently, Deep Reinforcement Learning (DRL) has been explored to optimize multiple objectives simultaneously in deterministic networking environments [126] [146]. Despite these advances, classical methods face critical limitations when applied to large-scale, dynamic networks. The scalability issue is prominent, as the complexity of multi-objective routing grows exponentially with the number of nodes and objectives, making problems like multi-objective routing optimization problems (MORP) NP-hard [140]. On top of that, all these works are based mainly on heuristic algorithms, but such solutions **lack guaranteed optimality and often converge to suboptimal solutions** [80][85][17], while ML-based approaches require extensive training data and exhibit slow convergence, limiting their adaptability in real-time scenarios [109]. These challenges underscore the need for advanced techniques, such as quantum and quantum-inspired optimization, to efficiently handle multi-objective routing in next-generation networks [140].

In contrast to classical approaches, quantum computing offers new ways to tackle complex optimization by leveraging quantum parallelism and tunnelling to address complex optimization problems. Quantum algorithms have been applied to address MORP and multi-objective routing and spectrum assignment (MO-RSA) in next-generation networks. Quantum Annealing (QA) has been applied to shortest-path and optimal routing problems by mapping them to QUBO or Ising models suitable for D-Wave systems [104][13]. Since many NP-hard combinatorial problems can be reformulated as Ising models [100], QA is a natural fit and has recently been used for multi-objective quadratic optimization as well [16]. Gate-based quantum computing approaches such as the Quantum Approximate Optimization Algorithm (QAOA) have also shown promise. QAOA aggregates multiple routing objectives (e.g., cost and delay) into a single QUBO formulation [116] and alternates cost and mixer Hamiltonians, with parameters optimized classically (e.g., via COBYLA) [36]. Case studies demonstrate that QAOA can compute Pareto-like solutions for small dual-objective routing graphs [74]. Hybrid approaches combining Variational Quantum Eigensolve (VQE) and QA have also been proposed for trust-aware routing in IoT and metaverse contexts [74]. Although current implementations remain constrained by Noisy intermediate-scale quantum (NISQ) hardware [50], they demonstrate the feasibility of quantum methods for NP-hard multi-objective problems. However, scaling to realistic network topologies remains an open challenge. As near-term quantum computers remain limited, quantum-inspired algorithms—such as the Coherent Ising Machine (CIM) and the Simulated Bifurcation (SB) algorithm—have emerged as practical alternatives. Both are designed to find low-energy states of an Ising model efficiently, which corresponds to optimal or near-optimal solutions of the combinatorial problem encoded (e.g. the routing optimization).

In summary, trusted path routing optimization extends the traditional shortest path problem by considering both communication network metrics and trust/security metrics. Classical multi-objective optimization techniques (like multi-criteria Dijkstra or evolutionary algorithms) have laid the groundwork for handling multiple network objectives. Building on that, quantum computing approaches – notably quantum annealing – offer a novel way to encode and solve the routing problem on quantum hardware, potentially finding better solutions through quantum parallelism and tunneling. Furthermore, quantum-inspired algorithms such as CIM and simulated bifurcation bridge enable practical large-scale optimization, solving the same formulations efficiently on classical or specialized hardware. Emerging literature suggests that combining these advanced techniques can significantly enhance routing decisions – reducing latency and packet loss while ensuring paths traverse trusted nodes. As research progresses, we expect to see “trust-aware” routing algorithms that leverage both quantum computing and quantum-inspired optimizers to meet the dual goals of efficient communication and secure, reliable data delivery.

Chapter 3

System Model and Assumptions

As aforementioned, the main vision of CASTOR is to achieve **secure data transmission by engraining trust throughout the traffic engineering process.** In this context, the contributions of the CASTOR framework span across the Compute Continuum - i.e., from network elements to the control plane - and provide the mechanisms for robust and adaptive-to-changes trust characterization. In what follows, we present the system model considered in the overall framework, highlighting the main entities in the path control pipeline and the core added values brought forth by the innovations of CASTOR. This section also outlines - highlighted in *italic* - the key assumptions that guide the first version of the architecture design described in Chapter 6. Finally, Section 3.3 builds on top of the System Model and introduces a first break down of the threat model that dictates the CASTOR trust assessment evaluations.

3.1 Conventions and Definitions

This section summarizes the core terminology adopted in CASTOR, building on the well-established conventions defined in the various IETF drafts on **Trusted Path Routing** [27] and **Control** [51] and **Data Planes** [52]. This short vocabulary is intended to lay the foundation on the core pillars/layers of CASTOR's architect (Chapter 6) and their associated functionalities towards the establishment of the novel trust- and network-aware (inter-networking) routing architecture. Particular focus is been given on those important specifications on the considered entities and endpoints that participate in the secure path establishment and enforcement - considering also the characterization and quantification of trust in complex environments such as the ones encountered in today's Internet architecture. *This vocabulary will act as the reference resource for all following chapters and will be enhanced throughout the project lifecycle (and in all subsequent technical deliverables) coalescing the concrete definition of each layer (as part of CASTOR's layered architecture depicted in Figure 2.1) as well as the specification of each building block exposing the necessary multi-path control capabilities*. This will allow all stakeholders have a common understanding of the characteristics that could be used to describe the trustworthiness of a data item or node as well as the methodology and concepts to be followed for allowing stakeholders to make a judgment if a service, function or entity can meet the required expectations - based on the trustworthiness requirements extracted by a service intent.

Autonomous System A collection of networks and router elements under common administrative control.

Control Plane The mechanisms that are responsible for the discovery, propagation and overall life-cycle management of routing paths. Any optimization tasks or communication overhead in the context of the overarching CASTOR Trust Assessment Framework should be done in a non-disruptive fashion, ensuring that the normal data plane traffic is not compromised.

Endpoint The source or destination of a path (as per [57]). Throughout the document the terms ingress and egress elements are used to signal the two endpoints of the path as well.

Forwarding Path An end-to-end path between two endpoints over which data-plane traffic is forwarded.

Path Profile A set of enforceable network and trust requirements that dictate the runtime characteristics of a forwarding path.

Path Profile Catalogue A set of the available path profiles that can be supported by an administrative domain. This constitutes a uniform trust environment that characterizes the underlying infrastructure layer - i.e., one or more ASes - that form the administrative domain.

Path Segment A list of segments that may span across multiple ASes or even administrative domains. This is intrinsically linked to the CASTOR optimization problem discussed in Chapter 5. A path segment can be translated into enforceable traffic engineering policies that form the forwarding plane that satisfies a concrete path profile as per the agreed (Secure) Service-level Agreements. Especially in the context of cross-domain service provisioning, a path segment culminates in the realization of a trust union across diverse ASes that can exhibit the required trust level. We have to highlight that this path segment is complementary to the foundational concept of ”Segment” in Segment Routing protocol. The latter considers a segment as a composition of elements that can execute instructions as part of a source-routed (Source Endpoint of a forwarding path) policy, whereas in CASTOR Path Segment abstracts this concept between ASes.

Traffic Engineering Policy Engine The main control plane infrastructure component (of CASTOR) offering control services for the path exploration process resulting to the enforcement of the optimal traffic engineering policies so as to enable continuous maintenance of (S)SLA compliance. This also extends to the enforcement of scalable trust- and network-aware routing updates, with high path freshness, for the re-establishment of SLA compliance in case of a propert violation.

Attestation Result Augmented Evidence Control plane trustworthiness evidence that provide cryptographically protected and verifiable evidence on the security posture of a routing element (the domain-agnostic concepts of IETF Remote Attestation Procedures are specified in [28]) In general, in CASTOR, trustworthiness evidence is essential for the overall lifecycle management of an underlying AS, and the service assurance based on agreed service level agreements. In CASTOR such evidence may be accumulated across a forwarding path, aiming to provide an efficient mechanism for evaluating the trustworthiness of links, paths or entire domains.

We have to highlight that in the context of ASes belonging to the same administrative domain, we assume that the network representation (capturing topology and temporal constraints) is maintained and kept up-to-date by the Orchestration layer towards controlling resource deployment. This is done by advancing MANO to understand and ac upon high level intents.). Declarative (northbound) network APIs are feeding the Orchestrator Service descriptors and features supporting the increased automation and continuous monitoring of the network state, respectively. This is an inherent feature needed for the optimal (inter-domain) path control and establishment based on a set of domains with known relationships by the Orchestrator. This information is then consumed by the Path Computation Element (PCE) as part of the traffic engineering process [138]. on the hand, in the case of intra-domain path establishment, service continuity is achieved through the employment of BGP-oriented routing protocols.

3.2 CASTOR as a *Trusted* Routing Path Extension towards Secure, Reliable Connectivity

Technology Strand #1: Trust- and network-aware TE process. Starting from the bottom left of the figure, we consider an Autonomous System (AS) that is controlled by a network operator and provides network connectivity in a single administrative domain. Regardless of the routing fabric (i.e., OSPF, IS-IS, MPLS) within each AS, one key requirement that is put forth in recent advancements in the context of serving sensitive workloads lies in the availability of security guarantees between the network elements that participate in the fulfilment of a service (e.g., EVPN, MPLS L3VPN). As per the IETF Trusted Path Routing (TPR) concepts, such guarantees would allow a network operator to ensure that workloads with certain trust requirements can be diverted through network links that have certain security capabilities. To this end, one primary added value of CASTOR (Added Value 1) relates to the provisioning of the in-router Trust Network Device Extension (TNDE) (Section 6.2.8). Building on top of the concepts of IETF TPR, the CASTOR TNDE is responsible for the secure monitoring (Section 6.2.9), collection and reporting of runtime trustworthiness evidence in a verifiable manner, allowing any verifier (be it an adjacent router or a controller entity) to attest to the network element's operational assurance.

The merits of the CASTOR TNDE is agnostic to the underlying hardware/software characteristics of the network elements provided that they posses certain root of trust (RoT) primitives (see Security Requirements in Section 9.1). For instance, a single router network element - if employed with the necessary RoT capabilities (e.g., equipped with a Trusted Platform Module) it is able to expose the capabilities envisioned in CASTOR. Even in the context of unmanaged networks, such TNDE capabilities could allow routers to autonomously enforce adjacency trust with no central coordination at runtime as part of the IETF TPR paradigm.
CASTOR Convention: Nevertheless, as we consider the advancement of complex and scalable networks that can cope with multiple traffic engineering requirements of mixed-criticality, *CASTOR incorporates into its architectural designs the concepts of network functions - e.g., forwarding rules, routing functionalities, route reflectors, path computation elements - being provisioned on top of virtualized infrastructure*. This allows for flexible resource allocation and on-demand provisioning of the necessary routing capabilities - e.g., even having multiple network elements treated as instances on top a common underlying infrastructure element - according to Service-level agreement needs.

Technology Strand #2: Trustful Service Union Establishment. Having laid out the key considerations for routing plane in Technology Strand #1, the rest of the CASTOR added contributions focus on the provisioning of the TNDEs in each infrastructure element and the transformation of the collected trust-related data into meaningful and actionable trust insights. First of all, one of the CASTOR added values lies in the design and implementation of the relevant TNDE functionalities that - depending on the threat model - can be enabled in order to ensure the secure collection of raw traces from the target in-router environment. Specifically, through novel tracing capabilities and their transmission to the envisioned Trust Sources, CASTOR aims to provide a complete pipeline for constructing trustworthiness evidence that characterize the trust posture of the network element with respect to its configuration integrity, but also its overall operational assurance.

Through the CASTOR-enabled secure and confidential channels, the aforementioned trustworthiness evidence is fed to the CASTOR Trust Assessment Framework (Added Value 2). Existing works have already made a preliminary attempt to tackle the concept of providing trusted paths in intra- and inter-domain scenarios [45], [27]. Despite the initial advancements towards establishing pre-defined trusted paths considering also a subset of the overall CC-wide threat model, they do not consider the emerging challenges of "Below Zero Trust" [139]. Specifically, considering the frequent fluctuations of the trust state of network devices, CASTOR envisions to provide robust, adaptive-to-changes trust evaluations that can reflect such changes in the calculated actual trust levels. Through continuous and dynamic trust evaluations, CASTOR is able to perform trust evaluations both at local (i.e., in-router) and global

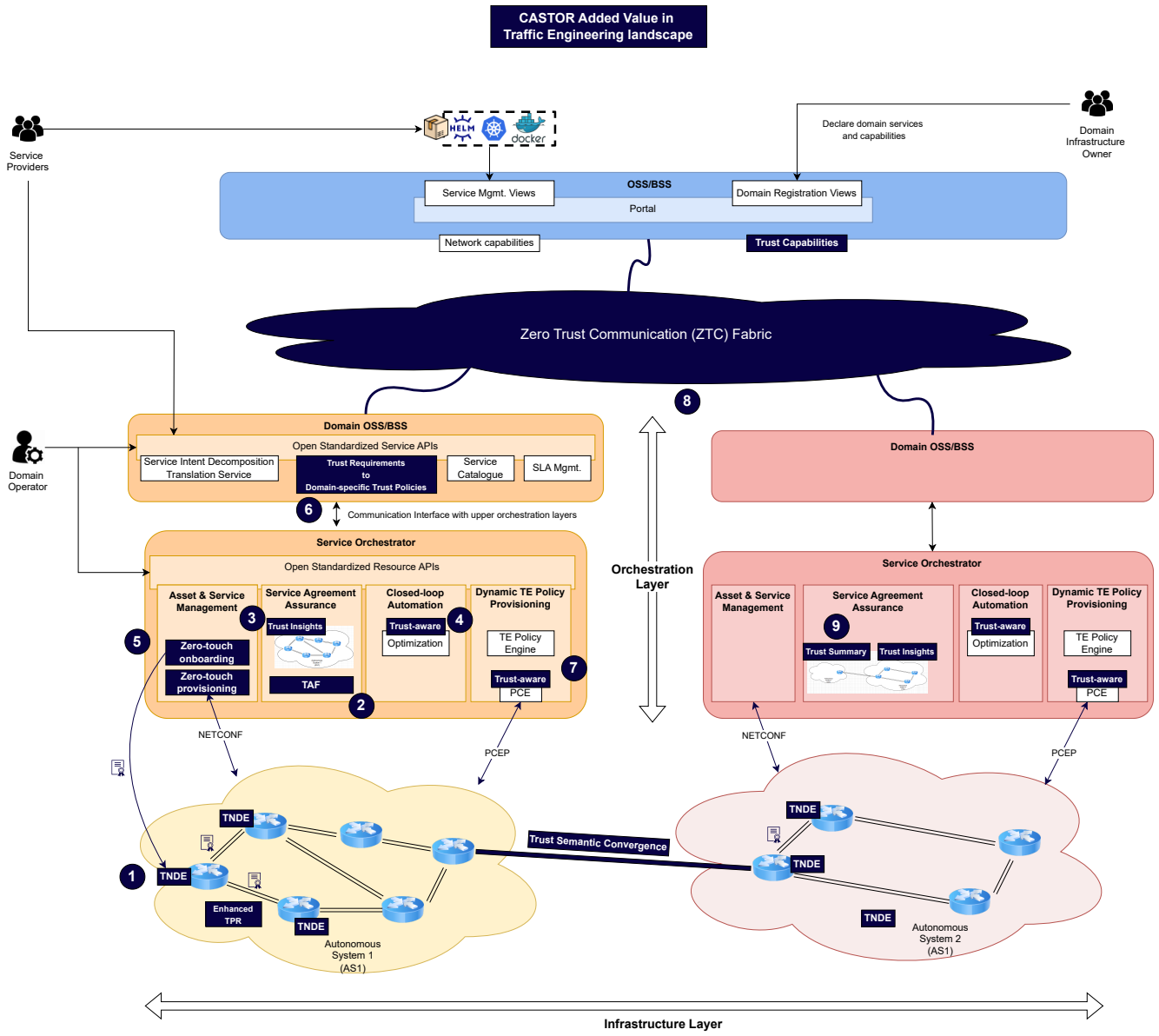

Figure 3.1: System Model and key CASTOR values in the Traffic Engineering landscape numbered from 1 to 9.

(i.e., controller) levels, allowing for the construction of rich trust insights that can guide service assurance and/or traffic engineering strategies (Section 4.4).

In addition, with the enhancement of the overall insights at the orchestration layer, CASTOR is able to construct a mirror of the topology graph at a controller layer for maintaining a fresh state of the network and trust posture of the underlying infrastructure elements (Added Value 3). Towards achieving trust-aware closed-loop automation, CASTOR leverages optimization techniques (Added Value 4; Section 6.2.7) to provide recommendations that satisfy both network and trust related requirements (i.e., objectives and/or constraints). These recommendations can then be converted into actionable and enforceable traffic engineering policies in order to ensure the adherence of agreed service-level agreements throughout the lifespan of a service.

CASTOR Convention: As an initial step toward incorporating trust requirements into traffic engineering pipelines, *CASTOR adopts a tiered approach to express the network and trust requirements that a domain can support. Using this approach, CASTOR maps SSLA clauses and objectives (defined by upper orchestration layers) to domain-specific network and, crucially, trust requirements.* Consequently, this tier-based service catalogue forms the basis of the path profile catalogue, which clusters service requirements according to a predefined set of policy templates specified by the domain operator.

Technology Strand #3: Agile Service and Transport Provisioning and Maintenance. All the aforementioned enablers spanned across the compute continuum culminate in the embedding of trust in the overall lifecycle management of traffic engineering policies. First, through the establishment of Trust Policies that define the minimum security requirements a network element must meet to participate in a path with specific trust capabilities, CASTOR provides trust-enabled management of all assets and services within an AS. From provisioning the necessary TNDE functionalities to securely onboarding new network elements—or even entire managed domains—CASTOR enables trust-aware decision making (Added Value 5) that determines when and how a network element or service should be reconfigured (Section 6.2.2).

The decision-making process within a managed domain is inherently determined by the requirements it can support and the capabilities it provides. In the context of intent-based networking - as illustrated in Section 2.2.1.3 - the derivation of the aforementioned requirements is the process of translating high-level requirements specified by the Service Provider into Service Level Agreements, and ultimately into domain-specific enforceable requirements and actions.
CASTOR Convention: *The entire service negotiation process that starts from the expression of the high-level service intents up until the realization of the established service-level agreement is orthogonal to the CASTOR project*. That being said, CASTOR's contribution (Added Value 6) on this fold is twofold: First, CASTOR envisions to extend existing, well-established, SLA schemas so as to incorporate trust objectives into an extended encoding, forming the concept of Secure SLAs (SSLAs). At the same time, CASTOR correlates SSLAs into concrete Trust Policies (Section 6.2.1) that can be enforced and validated at the infrastructure layer. Through this mechanism, CASTOR aims to resolve the multi-path control conundrum: translating the optimal path segments produced by the CASTOR optimization engine into enforceable control-plane instructions. This process highlights the flexibility of seamless traffic-engineering provisioning using trust-aware control-plane functionalities (e.g., the Path Computation Element; Added Value 7).

Finally, extending service provisioning beyond a single administrative domain (or AS) introduces additional challenges in ensuring end-to-end connectivity while meeting common network and trust requirements.
CASTOR Convention: *For the purposes of multi-domain service establishment, we assume that reachability and routing policy information between Autonomous Systems is already available through existing mechanisms for prefix announcement and propagation*. In this regard, CASTOR aims to expose the necessary interfaces that allow the participating domains to exchange abstract trust information without disclosing any sensitive - e.g., topological - information of each domain. Again, the contribution of CASTOR in this context is twofold. First, CASTOR enables to provide the necessary mechanisms - namely the Zero-Trust Communication Fabric - for sharing trust summaries between relevant stakeholders and ensuring trust semantic convergence across different administrative domains (Added Value 8). Second, CASTOR aims to leverage these common semantics across the domains in order to share trust summaries that can provide (cryptographically) verifiable information about the trust capabilities of a domain with the necessary level of abstraction (Added Value 9).

3.3 Threat Model

One of the CASTOR's main objectives is to enable trust evaluation across the Compute Continuum. Even starting from the transport network and the routing plane (as highlighted in Section 3.1), it is clear that the security concerns may span across different levels in the CC. Identifying and documenting the most prominent threats that may impact the CASTOR System Model is essential for advancing the collection of relevant trustworthiness evidence by the CASTOR Trust Source enablers. This information is required to enable the CASTOR Trust Assessment Framework to accurately evaluate the trustworthiness of a router element, link, path, or segment throughout their lifecycle. In the context of CASTOR, our primary focus

centres on ensuring the security and trustworthiness in the traffic engineering policy, allowing stakeholders to express both network- and trust-requirements when provisioning their service workloads.

One key point before delving into the threat model refers to the fact that there are various factors that could compromise the service level agreements of an established service. Apart from security implications, network defects (either at a router/switch or link level) or resource limitations are several examples of normal operation that may hinder the service fulfilment. Despite the fact that CASTOR Trust Sources (e.g., attestation process or operational assurance evaluations through FSM) may catch such events - which would lead to revised trust assessment they are not considered part of the envisioned threat model.

CASTOR's primary focus revolves around incidents that impact trust from a security standpoint (e.g., evaluating trust in terms of integrity, confidentiality, robustness). CASTOR's threat model spans across the Compute Continuum and takes into account the attacker capabilities as well as the potential cascading attacks that may lead to more complex compromises with heavy impact. Adhering to zero-trust principles, CASTOR's threat model encompasses attack vectors stemming from both local and network origins, including host-targeted denial-of-service attacks and routing-protocol compromises (e.g., BGP session hijacking, active wiretapping of BGP sessions), as well as broader classes of threats beyond these examples. As further illustrated in Deliverable D3.1, and considering the concept of virtualized router elements as presented above, it becomes apparent the need to distinguish between host-based threats and network-based threats. While the former may focus on attacks that may impact either the router function or the underlying host virtualization environment, the latter one relates to attacks that originate from the forwarding and data plane. All these attack vectors calls for robust tracing and detection mechanisms that need to catch such threats to the operational assurance of the network and result in a proportional decrease in the overall trust of the affected node, link and path.

Zero-trust, new threat actors, and dynamic environments necessitates a robust threat model. This strong adversarial model, encompassing security, privacy, and trust considerations, guides the requirements defined in Chapter 9 of this deliverable. The methodology employed for establishing the requirements, encompassing security, privacy, and trust considerations, is firmly grounded in the real-world security requirements articulated by key stakeholders in the context of real production ready scenarios. For instance, in the case of CASTOR use cases on smart automotive scenarios, the 5GAA and ETSI standards (see Section 8.4) serve as a guideline for the CASTOR requirements and their KPIs. To that extent, the consortium of CASTOR harnessed the expertise and input of relevant stakeholders, ensuring that the needs and specifications are accommodated. Furthermore, as we move towards cross-domain service provisioning, privacy considerations are also important to ensure that the participating administrative domains are able to establish end-to-end paths with certain trust guarantees, without exposing any sensitive topological information to the participating parties.

A detailed threat analysis as it pertains to specific threats and vulnerabilities which can be exploited to launch attacks against core security properties like confidentiality, integrity, availability will be elaborated in Deliverable D3.1. This threat analysis is also accompanied with a detailed mapping of the necessary trustworthiness evidence that needs to be collected by the CASTOR-enabled Trust Sources and evaluated buy the CASTOR Trust Assessment Framework so as to reflect the possible exploitation of a single threat to the appropriate trust re-calculations that will lead to an accurate and timely trust decision.

Chapter 4

Extending Trusted Path Routing: Manifesting Evidence-based Theory for Runtime/Explicit Trust Assessment

The Trusted Path Routing (TPR) architecture ensures that only **attested and trustworthy network devices are included in routing decisions.** In this model, each forwarding element is evaluated by a Verifier prior to its inclusion in a trusted network domain. Evidence about the device's integrity is assessed to determine its eligibility for participation in the routing topology. **While this enrolment-time verification establishes a baseline of trust, it does not account for the fact that a device's trustworthiness may change over time.** If a device becomes misconfigured, compromised, or enters a degraded trust state after initial enrolment, this change should be reflected in the trusted path routing decisions. The TPR model [27], as currently defined, provides no mechanism to detect or respond to such changes. Extending it with a runtime trust assessment phase raises several open issues that we need to resolve. This constitutes one of the core challenges that CASTOR aims to address for the employment of a Trust Assessment Framework capable of continuous (real-time) trust evaluation and quantification. As elaborated in Chapter 2, we have witnessed advancements with respect to trust evaluations in both cloud environments (e.g., evaluating the trustworthiness of Virtual Network Functions in central B5G/6G deployments) and far-edge ecosystems (e.g., ensuring the trustworthiness of the data integrity used as part of V2X cooperative communication). However, one of CASTOR's key objectives is to bridge the two ends of the Compute Continuum by introducing the concepts of continuous and dynamic trust evaluations in the routing plane.

By adhering to zero-trust principles, CASTOR's Trust Assessment Framework does not assume any trustworthiness on the routing plane. Instead, it relies on observable and timely evidence from the underlying network elements in order to evaluate their trustworthiness throughout their lifecycle. Routing elements are admitted into a trusted network domain based on integrity evidence collected at enrolment time. However, operational state may change after admission, potentially impacting the trustworthiness of a device. Without a runtime model, the Verifier cannot detect such changes or update routing decisions accordingly. This creates a need for mechanisms that enable continuous trust assessment and define how Verifiers or orchestrators interact with routing elements beyond initial attestation.

In fact, this is aligned with the emerging concepts of the IETF Trusted Path Routing specification, where router elements need to exchange trustworthiness evidence (e.g., that they have securely booted) prior to the provisioning of any communication channels. These pieces of evidence may have different characteristics depending on the property of interest under evaluation as well as the level of confidence (or uncertainty) that is associated with them. This highlights CASTOR's vision towards a continuous and dynamic trust characterization which enables - in its simplest form - the establishment and continuous maintenance of trust relationships in the routing plane while taking into consideration evidence coming

from multiple, diverse, and potentially conflicting trust sources.

By examining these concepts from a different perspective, the incorporation of trust requirements in the context of routing decisions introduces new challenges beyond the mere consideration of modern network-based Service Level Agreements. Specifically, starting from the node-centric trust evaluations of a single network element, CASTOR's overarching Trust Assessment Framework envisions enabling trust calculations that characterize communication links and eventually entire paths. In essence, this unlocks the incorporation of trust-aware decision making into the establishment Traffic Engineering policies that are able to satisfy enhanced Security Service Level Agreements (SSLAs) with both network- and trust-related objectives.

The aforementioned CASTOR objectives on the Trust Assessment Framework introduce a wide variety of challenges. First, this endeavour requires the establishment of precise and unambiguous definitions of trustworthiness and trust, as well as the careful selection of the underlying trust assessment methodology. A first specification of these concepts is documented in Deliverable D4.1 [40]. This chapter serves as an introduction to the main challenges that CASTOR aims to address in the context of trust assessment. In order to achieve this, it is essential to provide an initial sketch of the core principles and trust properties of interest that are relevant in CASTOR (Section 4.1). This allows to to shed light on the current landscape towards incorporating trust-aware decision making in the Traffic Engineering process (Section 4.2). In addition, Section 4.3 introduces the main characteristics that drive the decision behind the probabilistic logic - i.e., methodology - that needs to be employed in CASTOR, constituting the backbone of the overarching trust assessment framework (stemming, also, from the characteristics of the System Model presented in Chapter 3). Building on top of this, Section 4.4 describes the high-level components that comprise the Trust Assessment Framework, helping identify the key challenges applicable to each component. Eventually, to the culmination of the main challenges and research questions (Section 4.5) that will guide the design, implementation, and evaluation of the CASTOR's Trust Assessment Framework and its different modalities spanning across the Compute Continuum.

4.1 Definition of Trust and Trustworthiness

4.1.1 Overall Principles

Before delving into the concepts of **trust** and **trustworthiness**, it is important to begin with the participating actors: the *trustor* and the *trustee*. In this scenario, a trustor is an entity with a certain requirement, and an expectation that this requirement will be fulfilled by some other entity. The trustee, on the other hand, is the entity that aims to fulfil the expectation of the trustor.

Let us consider an indicative example in the context of the Trusted Path Routing paradigm, where two routers, say router A and router B aim to establish a secure connection between them. In this scenario, router A - i.e., the trustor - may expect router B - i.e., the trustee - to forward packets in a secure manner (i.e. an expectation of *confidentiality*) that have not been modified (i.e. an expectation of *integrity*) and within a reasonable time-frame (i.e. an expectation of *availability* and low *latency*). We now further explore the concepts of trust and trustworthiness.

- Trustworthiness can be defined as the likelihood of a trustee B to fulfil trustor A's expectations in a given context.
- *Trust* represents a decision (or disposition) by a trustor to place, or withhold, trust to a specific trustee. If a trustor decides to trust a given trustee, the trustor believes that, with high confidence, the trustee will fulfil the trustor's expectations. Alternatively, given the knowledge that an entity, A, trusts another entity, B, it can be said that "A trusts B" is equivalent to the fact that "A believes B to have the property of *trustworthiness*".

Note that in contrast to trust, which is a property of the trustor, trustworthiness characterizes the trustee. Trustworthiness quantifies the trustee's capacity to fulfil the trustor's expectations. It is vital that a trustor appropriately evaluates the level of trustworthiness of an entity such that it can be balanced with the trustor's expectations to ensure that trust is warranted.

Breaking down the concept of trustworthiness further, the degree to which a trustee can be considered trustworthy is based on both their ability to meet the expectations of the trustor, as well as the extent to which their capabilities are aligned with the trustor's goal.

Trustworthiness itself is confined to a specific *context*. In other words, a trustee is *only* expected to fulfil the goal of a trustor given a set of restrictions and/or circumstances. As an example, a router may be expected to verify the contents of a packet but only with authorised access permissions.

The expectation of a trustor can relate to *data*, or to the *behaviour* of the trustee itself. For example, there may be an expectation of data *correctness* (i.e. the data is accurate with respect to its intended meaning and without errors, such as a sensor reading faithfully representing its measured quantity), but there may also be an expectation relating to the functionality of the entity itself (for example, an entity may be expected to securely and consistently transmit data every specified time-frame).

The collection of, and interpretation of evidence, provides the trustor with the ability to assess the likelihood with which a trustee is able to fulfil its request. This is known as *trust verification*, and directly influences the decision making process. Evidence can exist in several forms, for example:

- Proof of the trustee's past behaviour, ideally in the same (or a similar) context to that of the desired task,
- Independent assessments made by other entities regarding the trustee's ability to achieve the goal,
- Information on regulatory restraints that may allow or prohibit the trustee from achieving the goal.

Although evidence is *objective*, trust verification is *subjective*. In practice, this means that two independent trustors may interpret the same evidence differently based on their own policies and ultimately produce different trust verification results (even to the extent that one trustor does not view the evidence as sufficient to grant trust, whereas the other is satisfied).

4.1.2 Trust Properties of Interest in CASTOR

In constructing this more contextually-aware representation of trust, these additional properties provide critical context that directly inform how trust is formed in practice, influencing evidence interpretation and, ultimately, the resulting trust decision. Table 4.1 defines the initial set of trust-related contextual properties used in CASTOR's trust modelling process.

Trust Properties in CASTOR	
Property	**Description**
Confidentiality	An assurance that sensitive information is protected from unauthorised access, solidifying the expectation of privacy in a given trust relationship. For example, vendor-specific information relevant to evidence collection should be encrypted to prevent unintended disclosure, and only shared on a need-to-know basis.

Integrity	A guarantee that data is not modified by unauthorised means during transmission and storage, and that any given entity in the network is faithfully reporting accurate, timely and legitimate evidence. For example, data (such as an attestation report) sent from one router should arrive at its destination exactly as it was sent. In CASTOR, the Local TAF only assesses trust within the context of integrity. However, additional properties of trust are also assessed at the level of the Global TAF.
Availability	An assurance that a network can maintain operational continuity to authorised users, even in the event of disruptions and/or attacks. For example, in the event that a link goes offline, degrades in performance or is compromised in some way, CASTOR should be able to efficiently establish a new path such that downtime is minimised/non-existent.
Robustness	A guarantee of a system's ability to adapt to and recover from disruptions and/or attacks to ensure operational continuity. For example, effective mitigation strategies should be implemented to handle issues such as packet loss or malicious attacks.

Table 4.1: Contextual properties of trust.

4.2 Elevating Trust metrics as a core enabler in Traffic Engineering Provisioning

4.2.1 Current Considerations in Trusted Path Routing

The IETF's approach to Trusted Path Routing (TPR) builds upon the foundational architecture defined in the Remote Attestation Procedures (RATS) framework. At its core, RATS enables a relying party (e.g., a verifier or orchestrator) to assess the trustworthiness of an attester (e.g., a router) by verifying evidence about its internal state, typically rooted in hardware-level security anchors such as Trusted Platform Modules (TPMs) or Trusted Execution Environments (TEEs). This verification process is intended to ensure that only devices with a verified, attested software and hardware configuration are permitted to participate in critical network operations.

In the current IETF TPR model, trust decisions are tightly coupled to the device onboarding process. Specifically, before a router is admitted into a trusted routing domain, it undergoes remote attestation. This is a process based on evidence. A verifier receives cryptographically signed evidence from the attester, such as measurements of firmware, kernel, or configuration. This is typically done during its boot or initialization phase. If the attestation is deemed valid, the device is issued a form of endorsement, such as a "trust passport," authorizing it to participate in trusted path computations. These trust passports may be periodically revalidated but are generally not tied to fine-grained, runtime monitoring.

This onboarding-centric trust assessment implies that a device is either trusted or not at the moment of domain admission, and this trust status persists unless explicitly revoked. While this approach is well-aligned with traditional, relatively static infrastructure deployments, it introduces a significant limitation for dynamic or multi-tenant environments such as the computing continuum or zero-trust architectures. In particular, it does not accommodate real-time fluctuations in trust posture due to software updates, behavioural anomalies, lateral movement of threats, or emerging vulnerabilities.

To compensate for this static model, TPR also explores the possibility of incorporating trust policies into routing protocols. For example, when a path is computed, devices may be excluded based on their attested trust level or associated trust attributes. However, this is still fundamentally bound to the trust state as observed during onboarding.

So we can summarize the RATS-Based approach in Trusted Path Routing as follows:

- Binary Trust Posture: Devices are typically considered either trustworthy or untrustworthy based on a point-in-time evaluation.
- Boot-Time Evidence Focus: The trustworthiness of a router is often inferred solely from its boot-time measurements, omitting behavioral or operational metrics.
- Lack of Runtime Adaptivity: Dynamic changes in a router's software, configuration, or runtime behavior are not reflected in the active trust model unless explicit re-attestation is triggered.

4.2.2 Trust objectives in Service-Level Agreements

The Trust Assessment Function (TAF) architecture introduced in CASTOR decouples trust inference from the forwarding plane by establishing a dual-layer assessment mechanism: a local TAF embedded in each node, responsible for measuring device integrity and runtime state, and a global TAF, which synthesizes network-wide observations into node and link-level trust metrics. The outcome is a set of dynamic trust indicators for nodes and links that reflect the real-time trustworthiness of the network and evolve in response to fresh evidence.

The dynamic nature of CASTOR's trust architecture—where evidence is continuously collected, processed, and evaluated by both local and global Trust Assessment Functions (TAFs)—demands that the control plane no longer treats path computation as a static process driven by stable, low-frequency updates. Instead, it must evolve to handle ephemeral, granular shifts in trust posture. This shift in architectural needs justifies a structural augmentation to the routing stack. Not only should routers advertise connectivity and performance characteristics, but also current trust scores, endorsements, and verified capabilities. Without these enhancements, routing logic will remain blind to the very attributes that define whether a path is usable under evolving SSLAs.

Current path computation processes are not equipped to consume, reason over, or prioritize trust scores as dynamic metrics. In CASTOR, a node deemed trustworthy at the time of initial path computation may no longer meet SSLA constraints minutes later due to trust degradation, even if its interfaces and metrics remain unchanged. To accommodate this, routing convergence must be extended beyond traditional triggers: changes in trust must be treated on par with link failures or performance degradation. That means we need to transition to trust-aware routing where a drop in ATL or RTL triggers revisions in the Traffic Engineering policies (e.g., Updates to Flex-Algo topologies or enforcement of updated SR policies), because the path is no longer admissible under trust-centric service guarantees. This redefines path engineering to incorporate evidence-based, runtime trust semantics.

Routing mechanisms - such as Flex Algo Definitions - allow the construction of overlay (virtual) topologies tailored to specific objectives. In an administrative domain, there can be multiple such overlay topologies, each constrained by inclusion or exclusion of links based on administrative tags (admin groups), prefix preferences, or metric types. This design is already used in scenarios where certain traffic classes must avoid low-bandwidth links, minimize latency, or maintain disjointness from other flows. In CASTOR, such mechanisms need to be enhanced to support per-profile trust topologies.

For example, in the context of Flex Algo, an instantiated Flex Algo topology could correspond to a trust level defined in the Service Catalogue of a domain; e.g., Flex Algo ID 128 corresponds to a high-assurance path profile, while Flex Algo ID 129 corresponds to a medium-assurance path profile. A Flex Algo topology designed to only use "green" links representing trusted and attested infrastructure, effectively slices the topology into a high-assurance virtual network, even if the physical underlay remains the same. Candidate paths computed by the PCE or by ingress routers can then select among multiple SR Policies, each aligned with a particular trust class. In CASTOR, the orchestrator may dynamically alter the preferred candidate path as the trustworthiness of infrastructure changes over time.

4.3 Subjective Logic as a Foundation for Evidence-Based Trust Assessment

In CASTOR, the concept of trust is not binary. Instead, trust is treated as context-dependent and built upon fragmented or incomplete evidence, which means there is always some uncertainty in the assessment. To handle this uncertainty systematically, we adopt Subjective Logic as the foundational calculus for reasoning over evidence and computing trust metrics such as the Actual Trust Level (ATL) and the Required Trust Level (RTL). Subjective Logic is particularly suited to decentralized systems, where trust decisions rely on both direct and referral evidence of varying credibility and scope.

Subjective Logic is a probabilistic logic framework designed to express beliefs in the presence of uncertainty. Its core abstraction, the opinion, extends classical probability by attaching degrees of belief (b), disbelief (d), and uncertainty (u) to a given proposition, along with an optional base rate (a). This makes it ideal for trust systems where assertions about system integrity or policy compliance are rarely black-and-white.

4.3.1 From Evidence to Opinions

Each Local TAF within CASTOR aggregates diverse sources of trust evidence: secure boot measurements, attestation reports, violation events on the behavioural assurance, and potentially even neighbour assessments. However, these sources differ in reliability, coverage, and semantics. Instead of merging them through ad hoc scoring rules, CASTOR uses Subjective Logic to formalize this fusion.

Trust propositions such as “the router has booted correctly” or “the link is confidential” are evaluated into opinions, e.g., ω = (b=0.7, d=0.1, u=0.2), which concisely encode both belief and remaining uncertainty. This is essential: in order to take decisions, this uncertainty must be explicit and measurable.

In principle, the CASTOR Trust Assessment Framework requires the aggregation of diverse trustworthiness evidence: e.g., secure boot measurements, guarantees on the runtime operational assurance of a router and potentially even neighbouring assessments as part of the secure link establishment in the context of Trusted Path Routing. These sources differ in reliability, coverage, and semantics. Rather than fusing them using ad hoc scoring rules, CASTOR adopts a *formal evidence-to-opinion methodology* grounded in Subjective Logic.

This methodology aligns with recent advances in dataset trustworthiness assessment, where trust propositions are explicitly defined (e.g.,“the dataset is not biased”) and evaluated using quantifiable evidence mapped to belief, disbelief, and uncertainty components. In CASTOR, trust statements such as “the router booted securely” or “the link is confidential” are similarly formulated as *trust propositions*, each mapped to a Subjective Logic opinion $\omega = (b, d, u, a)$. This enables CASTOR to transform heterogeneous raw evidence into a normalized opinion space, making reasoning and aggregation tractable even in conflicted, incomplete, or federated evidence scenarios.

Moreover, just as the dataset trust methodology incorporates *different quantification models* (e.g., constant-uncertainty or evidence-weighted) based on context, CASTOR may select the appropriate opinion formation mechanism depending on the nature of the evidence (e.g., deterministic attestations vs. behavioural heuristics). This preserves both *semantic interpretability* and *uncertainty propagation*, ensuring that decision-making processes remain aware of evidence quality and gaps.

This principled transformation from evidence to opinion is pivotal to enable trust-aware actions across CASTOR’s distributed architecture.

4.3.2 Discounting and Indirect Evidence

In CASTOR, trust assessments are not limited to direct self-measurements (i.e., assessment made by an isolated trust assessment process within the router). When the orchestration layer obtains both A's trust score and the trust scores A reports for its neighbours, it should apply a discount operator to A's reported opinions, with the discount determined by how much the orchestration layer trusts A. This ensures that evidence from less reliable sources is appropriately weakened (typically by increasing the uncertainty component), while strong referrals retain more influence. Thus, trust discounting allows transitive trust to be computed robustly in a dynamic, distributed environment — mirroring the role of referrer credibility in multi agent systems. This is captured formally through *trust discounting*, a fundamental operator in Subjective Logic.

When a node A receives an opinion ω_B^X from a node B regarding proposition X, and holds an opinion ω_A^B on B's trustworthiness, a derived opinion ω_A^X is computed using a trust discounting operator, such as:

$$\omega_A^X = \omega_A^B \otimes \omega_B^X,$$

where $\otimes$ denotes a discounting operator appropriate for the edge type and context.

In CASTOR this operator will be used by the global TAF to discount opinions coming from the local TAFs. However, CASTOR also supports advanced configurations through the use of a recently introduced *Referral-Edge Path Discounting Operator* [114]. This operator addresses the limitations of prior models by enabling consistent discounting along chains composed entirely of referral (indirect) trust links, a frequent scenario in dynamic and federated environments. Moreover, this operator can be used to calculate trustworthiness of paths containing multiple referral edges within complex networks.

By integrating these operators, CASTOR is able to compute transitive trust robustly across arbitrary network topologies and maintain meaningful uncertainty estimates. This approach mirrors the structure of real-world multi-hop trust relationships, such as those found in vehicular systems, cooperative IoT, and collaborative learning contexts.

4.3.3 Fusion of Multi-Source Evidence

Subjective Logic also supports trust fusion, enabling CASTOR to combine multiple opinions about the same node or proposition into a single view. This is particularly important at the level of the Global TAF, which collects trust reports from various Local TAFs (and potentially other telemetry sources) and must resolve conflicting or partial views.

CASTOR leverages several fusion operators [83] from Subjective Logic, depending on the independence assumptions and semantic nature of the evidence:

- **Consensus Fusion** is used when the opinions are assumed to be independent. It reinforces strong agreement and penalizes conflicting beliefs. Given two independent opinions $\omega_1 = (b_1, d_1, u_1, a)$ and $\omega_2 = (b_2, d_2, u_2, a)$ with the same base rate a, the fused opinion $\omega_c = (b_c, d_c, u_c, a)$ is given by:

$$K = u_1 + u_2 - u_1 u_2, \quad b_c = \frac{b_1 u_2 + b_2 u_1}{K}, \quad d_c = \frac{d_1 u_2 + d_2 u_1}{K}, \quad u_c = \frac{u_1 u_2}{K}.$$

  This operator is useful for combining attestation opinions from distinct, independent Local TAFs [83].

- **Averaging Fusion** is suitable when the opinions may be dependent or correlated. It ensures proportional influence of each source. For n opinions $\omega_i = (b_i, d_i, u_i, a)$, the average fused opinion is:

$$b_a = \frac{1}{n}\sum_{i=1}^{n} b_i, \quad d_a = \frac{1}{n}\sum_{i=1}^{n} d_i, \quad u_a = \frac{1}{n}\sum_{i=1}^{n} u_i.$$

This operator is applied in CASTOR when integrating behavior-based metrics that may reflect common environmental factors or correlated measurement noise.

- **Cumulative Fusion** is applied when opinions represent sequential or additive evidence—ideal for time-series-based monitoring. For two opinions with belief masses r_1, r_2 and evidence strengths n_1, n_2, the cumulative opinion is:

$$b_{cum} = \frac{r_1 n_1 + r_2 n_2}{n_1 + n_2}, \qquad u_{cum} = \frac{K}{n_1 + n_2}, \quad \text{with } K \text{ being a prior weight (e.g., } K = 2).$$

Disbelief is derived by $d_{cum} = 1 - b_{cum} - u_{cum}$. This operator supports incremental trust tracking over time.

As emphasized in [83], the choice of fusion operator has a critical impact: consensus fusion generally reduces uncertainty with more sources; averaging fusion can smooth but also dilute strong opinions; and cumulative fusion reflects evidence accumulation over time. CASTOR selects the appropriate fusion model for each use case to ensure accurate, interpretable, and robust trust aggregation in dynamic environments.

For example, in the context of trust calculations at the orchestration layer, fusion is applied to combine opinions received from two in-router trust evaluations that characterize a common trust proposition such as the link security or communication reliability of the shared link. It is also important to note that CASTOR's architecture allows for the definition of custom fusion operators, should the need arise to address trust aggregation scenarios that are not adequately handled by existing Subjective Logic operators. For instance, such extensions may be necessary for basic fusion schemes where the aggregated trust opinion is defined as the minimum among individual opinions. In that case, this would require the introduction of an ordering relation over subjective opinions.

4.4 CASTOR TAF high-level description

The CASTOR architecture redefines trust in the network not as a binary property evaluated at onboarding, but as a continuous and quantifiable process. In this model, each network element is not evaluated as simply "trusted" or "untrusted," but along a scale that reflects the degree to which it can be trusted to participate in critical routing decisions, based on runtime behaviour and metrics. Detailed presentation of the CASTOR TAF blueprint is provided in Deliverable D4.1.

To achieve this, CASTOR adopts a federated trust model that distributes trust assessment responsibilities across the network, introducing two core components: the Local Trust Assessment Function (Local TAF) and the Global Trust Assessment Function (Global TAF).

- Local TAF agent: It is instantiated on each participating router or network function. Its role is twofold: (1) to assess the trustworthiness of its own device (via local attestation and runtime monitoring), and (2) to evaluate the behaviour and integrity of neighbouring devices through evidence exchange and lightweight peer-to-peer validation protocols.
- Global TAF: It operates as a logically centralized trust orchestrator and it is responsible for building and maintaining a dynamic trust topology of the network. It collects trust reports from Local TAF agents, covering both self-assessments and neighbour evaluations. It combines these with additional evidence about the links (connections) of the network in order to have a complete picture of the trustworthiness of nodes and links. This results in real-time trust scores of the whole topology that guide secure path selection in compliance with SSLA requirements.

4.4.1 The Architecture of the Trust Assessment Framework (TAF)

The Trust Assessment Framework (TAF) is a modular system architecture designed to enable evidence-based evaluation of trust in dynamic, distributed environments. This section introduces the overall architecture of the TAF, setting the foundation for the subsequent discussion of its two main variants: the Local TAF and the Global TAF.

The CASTOR TAF consists of five tightly integrated functional sub-components that enable trust to be calculated based on runtime and static evidence.

4.4.1.1 Trust Model Manager (TMM)

The Trust Model Manager is responsible for instantiating and managing the internal representation of trust relationships. These relationships are modelled as directed graphs of entities and propositions. An entity might be a component of the system. The source of the graph is called the agent and the target is always the proposition to assess. Each proposition represents a logical assertion about some system property—for instance, that a software image is authentic, or that runtime behaviour has not exhibited anomalies.

The TMM supports the composition of atomic and composite propositions. Atomic trust opinions represent the trustworthiness of a single proposition (i.e. variable X). This type of opinion deals with only one specific aspect of trust, and the opinion about this proposition is formed based on direct evidence observed by agent A. Ideally, an atomic proposition is a proposition that cannot be broken down to simpler terms and evidence can either support it or contradict it.

For example, we might combine the following propositions:

- Proposition 1: ”VRouter has started”
- Proposition 2: ”vRouter is operational”
- Proposition 3: ”vRouter has been detected with vulnerability x”
- Proposition 4: ”vRouter forwards messages in less than $1\mu s$”

Composite propositions express higher-level assertions that are evaluated as a function of several atomic propositions using logical composition rules.

In practice, the TMM loads these models from policy-driven templates. This allows the system to adapt its reasoning logic based on operational context or threat models. For example, in a high-assurance environment, the model may include detailed runtime behaviour checks and multi-source redundancy, whereas in a lightweight deployment, a simplified trust graph may suffice.

4.4.1.2 Trust Source Manager (TSM)

The Trust Source Manager serves as the integration layer between raw evidence generators and the trust reasoning engine. It registers and coordinates the trust sources for a given trust model. Trust sources include static measurement tools (e.g., secure boot logs), dynamic monitoring components (e.g., host intrusion detection), remote attestation protocols, or behavioural telemetry systems.

The TSM abstracts the variability of these sources by transforming their outputs into structured, interpretable trust claims. It validates the authenticity and freshness of the data, handles conflict resolution among redundant sources, and converts inputs into normalized trust opinions.

Additionally, the TSM manages the life cycle of evidence collection: triggering periodic checks, subscribing to real-time feeds, and managing failure fallbacks when sources become unavailable or compromised. It ensures that the trust graph remains populated with current and relevant evidence.

4.4.1.3 Trustworthiness Level Expression Engine (TLEE)

The Trustworthiness Level Expression Engine is the computational core of the TAF. It takes the normalized trust opinions generated by the TSM and evaluates them within the trust graph defined by the TMM. This involves recursively combining trust opinions across the graph structure to compute the trust value of higher-level propositions.

The engine operates on Subjective Logic algebra, enabling it to manage, fuse, discount, and propagate trust as a first-class property of trust. This is crucial in environments where partial or conflicting evidence is common. For example, if two attestation reports partially disagree, or if one source is delayed, the TLEE does not discard the result but incorporates it with appropriate weighting (i.e., depending on the operator used it might increase uncertainty or put more weight to the most certain one.).

The TLEE outputs Actual Trust Levels (ATLs) for each target proposition defined in the policy. Each ATL contains the computed trust of the proposition and the source traceability that led to the computed result. This enables transparent audit and explanation of trust decisions.

4.4.1.4 Trust Decision Engine (TDE)

Once ATLs are computed, the Trust Decision Engine evaluates them against the Required Trust Levels (RTLs). The comparison may end to a binary result (trust granted or denied) by means of a threshold or an ordinal result (high, medium, low).

The TDE supports configurable policies for dealing with borderline or uncertain cases. For example, in safety-critical systems, an ATL below threshold may trigger immediate mitigation, whereas in resilient systems, it might prompt redundancy or escalation.

The TDE outputs actionable results: whether a proposition (which might be about a system or component) is considered trustworthy or not. These results are consumed by the orchestrator layer for the computation of the trusted path.

4.4.2 The Local TAF agent

The Local Trust Assessment Framework (Local TAF) agent is an instance of the generic TAF architecture that operates at the level of a single node, i.e., a network device such as a router. Its primary purpose is to continuously assess and reason about the local trustworthiness of the device on which it operates, using both static properties (e.g., platform configuration) and dynamic evidence (e.g., behaviour at runtime).

The Local TAF agent runs within a Trusted Execution Environment (TEE) or otherwise isolated system context.

CASTOR leverages the Local TAF agent to provide trust scores for devices that contribute to forwarding paths. These scores are ultimately exported to the Global TAF and will influence routing decisions.

4.4.2.1 Trust Model and Propositions

The trust model instantiated in the Local TAF is centered on device integrity. It typically includes the following atomic trust propositions:

- TP_{BOOT}: The device booted from a verified, cryptographically signed image.
- TP_{CONFIG}: The platform configuration (e.g., kernel parameters, control-plane policies) matches a trusted reference.
- $TP_{RUNTIME}$: No runtime anomalies (e.g., memory access violations, unexpected privilege escalations) have been detected.
- TP_{ATTEST}: The most recent attestation report is valid and fresh.
- TP_{FSM}: No misbehaviour has been reported by the FSM source deployed in the network element.

These propositions may be composed into composite trust assertions, such as:

- $TP_{INTEGRITY} := TP_{BOOT} \wedge TP_{CONFIG} \wedge TP_{RUNTIME}$
- $TP_{HEALTHY} := TP_{INTEGRITY} \wedge (TP_{ATTEST} \vee TP_{FSM})$

4.4.2.2 Trust Sources and Evidence

The Local TAF interacts with several trust sources, which act as evidence providers for evaluating the above propositions:

- CASTOR Attestation Source: These provide support for measuring the platform state, generating attestation reports, and validating those reports using expected reference values.
- FSM Source: This component reports on observed misbehaviour within the node. It can detect and classify security-relevant anomalies such as protocol violations, forwarding inconsistencies, or abnormal interface usage.

An open question is how to quantify and compare the trust value of evidence coming from FSM versus attestation subsystems. In some threat models, the FSM may detect real-time violations that attestations miss (e.g., mid-run privilege escalation). In others, attestation may provide stronger guarantees for static properties.

One option is to treat the two as complementary trust sources, each associated with its own confidence level. Another approach would be to define a trust fusion policy where, depending on the risk class of the service, different weightings apply to FSM vs. attestation-derived opinions.

4.4.2.3 Dynamic Neighbors Assessment

A key innovation in CASTOR is that each Local TAF is not only responsible for assessing its own device but also for forming trust assessments of its directly connected peers.

Each router periodically exchanges information with its immediate neighbours that may include:

- Current ATL of the neighbour, as claimed by its own Local TAF.
- Last attestation timestamp and freshness proof.
- Summarized FSM violation events or behavioural scores.

Upon receiving this evidence, the Local TAF evaluates the neighbour's trustworthiness and constructs a Neighbors Trust Vector, which will be reported to the Global TAF capturing the trustworthiness of itself and its immediate neighbours.

4.4.3 The Global TAF

The Global Trust Assessment Framework (Global TAF) is a service at the CASTOR Orchestration layer that maintains a dynamic, end-to-end view of trust across the network. Unlike the Local TAF agent, which evaluates the trustworthiness of individual nodes from a local perspective, the Global TAF composes these local evaluations with telemetry and interconnection evidence to derive a holistic picture of trust at the network-wide level. This capability is essential in the CASTOR architecture, where services must be routed over paths that satisfy stringent, end-to-end SSLAs.

The primary responsibility of the Global TAF is to assess whether end-to-end paths satisfy a specified Trust Profile. These profiles correspond to policies that define acceptable levels of trust for both:

- Nodes, evaluated via Actual Trust Levels (ATLs) produced by the Local TAF,
- Links, evaluated through telemetry, behavioural evidence, and cross-device reports collected at runtime.

The Global TAF does not merely aggregate raw trust values: it composes trust evidence across topology graphs to support constrained path selection and colouring. This requires carefully defined composition semantics and efficient mapping into data structures such as Segment Routing (SR) Policies and Flex-Algo topologies.

4.4.3.1 Trust Composition Across Paths

A central challenge for the Global TAF is trust composition: given a path composed of multiple nodes and links, how can their individual ATLs be combined to assess the overall trust level of the path?

Different trust properties call for different composition rules:

- Minimum: Suitable for properties like integrity and confidentiality, where the weakest element dominates the outcome.
- Average: Appropriate when trust reflects a statistical or experiential score (e.g., long-term resilience).
- Weighted Average: Used when certain nodes (e.g., ingress or egress) or links (e.g., across domains) are more critical.

For this, one can also use existing fusion operators or create a new one. The choice of operator affects both the evaluative semantics and the search space for the path computation engine. For instance, with minimum composition, high-trust links followed by one low-trust hop disqualify the whole path. In contrast, average-based profiles tolerate outliers under certain conditions.

Composition must account for the fact that link trust is not purely local. The trustworthiness of a link may depend on the trust of both endpoints (e.g., if a T_L node is involved, the link cannot be fully trusted), shared evidence (e.g., consistency in attestation reports) and observed behaviour (e.g., abnormal packet drops, latency spikes, or forwarding inconsistencies). This introduces the need for cross-node reasoning and may blur the separation between link-centric and node-centric trust.

This creation of path-level trust is a crucial and significant open question within CASTOR's Trust Assessment Framework, necessitating the effective fusion of both node-level and link-level trust evaluations. Local TAF agents at the node level only evaluate integrity, forwarding their derived ATLs to the Global TAF. However, these ATLs must be discounted based on the Global TAF's own trust in the local TAF agent's evaluative capabilities, often requiring additional evidence concerning the local TAF's secure operation

and functionalities. For link-level trust, the Global TAF may directly collect trustworthiness evidence, such as information about bandwidth and/or availability. As highlighted above, a significant challenge lies in defining the composition rules appropriately given the context and/or trust property being assessed, to aggregate potentially diverse trust opinions into holistic, verifiable path-level trust values. The difficulty is further amplified by the dynamic, multi-agent and complex nature of the overall network topology.

Furthermore, it is important to consider the extent to which information is shared from the local TAF to the Global TAF in relation to the establishment of path-level trust. While Local TAF agents forward their calculated ATLs (with respect to device integrity) to the Global TAF, it is unclear as to whether or not *only* this information is sufficient, or if further raw trustworthiness evidence (such as that obtained from different trust sources) must also be shared. Whilst the Global TAF must discount incoming ATLs based on its own trust in the Local TAF agent's capabilities, this inherently requires some form of verifiable information, in the form of evidence, about the Local TAF agent's trustworthiness within the current context.

4.4.3.2 Colouring and Integration with Routing

In the CASTOR architecture, trust information becomes operationally relevant only when it can influence concrete routing decisions. The conceptual bridge between abstract trust evaluations and the actual selection of forwarding paths is built through the mechanism of colouring. Colouring, in this context, refers to the act of assigning symbolic labels—called "colours"—to portions of the network topology that satisfy the conditions of specific trust profiles. This transformation allows trust assessments to be encoded into the language of routing, specifically into Segment Routing (SR) policies and Flexible Algorithm (Flex-Algo) instances.

Each trust profile is mapped to a unique colour identifier. For example, a profile requiring high integrity and confidentiality may be bound to colour 100, while a profile with more relaxed trust constraints might be assigned colour 200. These colours serve as scoping primitives, allowing both control plane components (e.g., the PCE or orchestrator) and data plane elements (e.g., routers implementing SR policies) to selectively operate within trust-aligned subgraphs of the network.

This process begins with the Global TAF, which uses node-level ATLs (provided by Local TAFs) and link-level trust evaluations (derived from telemetry and federated behaviour analysis) to determine which elements of the network meet the requirements of a given profile. Importantly, this determination is not static. As trust levels fluctuate due to ongoing monitoring and emerging evidence, the membership of each coloured subgraph must be continuously reevaluated. A single drop in the trustworthiness of a node may invalidate its participation in multiple trust profiles, and by extension, compromise the viability of paths that depend on it.

However, colouring is not a simple binary operation. Trust assessments often involve nuanced conditions that cannot be easily translated into yes-or-no labels. For instance, consider a link that satisfies confidentiality requirements (e.g., it is encrypted), but connects to a node whose integrity is questionable. Should this link be admitted into the high-trust subgraph? The answer depends on the underlying trust composition logic: if the trust profile insists that all components, including both link and endpoints, meet a uniform standard, then the link must be excluded. If, on the other hand, the profile tolerates partial trust in some dimensions while enforcing strict guarantees on others, the decision becomes more complex.

Several open challenges arise in the colouring logic:

- If a node is classified as T_L, should all its adjacent links be disqualified from the trust-colored subgraph?
- Can a link be trusted if only one of its endpoints is high-trust?
- Should link ATLs be computed as functions of endpoint ATLs plus link evidence?

- What happens when trust is asymmetric (e.g., node A trusts B, but not vice versa)?

These questions suggest that colouring must operate not on raw evidence, but on composed, policy-aware interpretations of trust.

The trust composition logic directly shapes how the CASTOR Optimization Engine selects paths. If trust is cumulative (e.g., using average), the Optimization Engine must track a trust budget as it explores the graph. If trust is based on minimum, the Optimization Engine may prune all branches below the threshold early. If weighting applies, trust annotations must carry metadata such as criticality weights. The Optimization Engine must therefore operate not only on cost metrics (e.g., delay), but on multi-dimensional annotations (trust, risk, availability) that are the output of the Global TAF.

4.4.3.3 Topology Evolution and Recolouring

In a typical network topology, trust is a dynamic concept. The trustworthiness of an entity in a network does not remain static, and is continuously assessed and re-evaluated in real-time in response to the collection of new and updated evidence. CASTOR's dynamic approach moves away from more traditional processes that treat trust as binary and only to be assessed at the moment of onboarding.

This means that a node deemed initially trustworthy may later fail to meet SSLA standards due to a degradation of trust, for example due to poor quality evidence or a malicious attack. CASTOR improves on these traditional processes, accommodating granular changes in trustworthiness such that optimal, safe and relevant network paths can be updated and chosen in real-time. As a result, changes in network trust must be handled swiftly and with high priority.

As discussed, colours are assigned to segments of the network to indicate adherence to the requirements specified by trust profiles. However, as trust is dynamic, the process of recolouring must be incorporated. If the trustworthiness of an entity drops based on the evaluation of new evidence, that entity's membership in certain trust profiles may no longer be valid, requiring the entity to be assigned to a different trust profile (i.e. recoloured). This ensures that alternative paths can be selected in the event of trust degradation, keeping the chosen path in alignment with the relevant SSLA.

4.5 Open Questions Relating to Trust Characterisation of Routers, Links and Paths

Several complex architectural and computational challenges arise regarding the optimal implementation of dynamic, distributed and accurate trust assessment in the CASTOR framework. Addressing these issues requires specific trust modelling decisions forming the foundation for dynamic trust assessment. In the following section, we detail the core open questions critical to defining the relationship and interaction mechanisms between Local TAF agents and the Global TAF in order to characterise the trustworthiness of routers, links and paths.

4.5.1 Information Sharing and Trust Models

A fundamental architectural question relates to the level of abstraction at which trust information should be sent from the Local TAF to the Global TAF. Should a Local TAF agent be able to share its derived ATL, effectively sharing a complete high-level trust opinion, or should it just transmit trust propositions and related evidence? Calculating ATLs at the infrastructure level may introduce complexity and ambiguity; after all, an ATL represents an ultimate evaluation of a complex composite trust proposition and may therefore be better positioned at the orchestration layer, i.e. calculated by the Global TAF.

A potential approach is to restrict the Local TAF to focusing on atomic trust proposition derivation and evidence collection, sending its opinions to the Global TAF as configured by the network operator. This approach enforces a clear separation of concerns, allowing the Global TAF to perform all logical aggregation and composition steps based on what is received from the Local TAF agent, in order to derive composite trust propositions specific to the required path profiles as well as ultimately determine the ATL of an arbitrary trust proposition.

In addition, the IETF Trusted Path Routing draft proposes the exchange of stamped passports (attestation reports) between routers at the link-level. Should the Local TAFs involved in a link consume these stamped passports during the overall evaluation of link-level trust? The concepts of links and paths, as well as how they can be composed, are discussed in more detail in Section 4.5.4.

Furthermore, should the ATL of the link be an aggregation of both local trust and remote-next-hop trust? In doing so, the boundary between the Local TAF agent and Global TAF would be blurred, and could allow the Local TAF to have more of a view on adjacency-level trust relationships.

Other relevant open questions revolve around that of managing trust model variability. Should the trust model change across different types of device? Also, given that CASTOR typically operators within a multi-vendor environment, how can consistency be maintained across TAF instances? Further still, can trust models adapt and be updated dynamically (for example in response to new threat vectors or degradation of link and/or path-level trust), or are they fixed once deployed?

During the onboarding phase, devices must also establish initial trust anchors. Should such anchors be pre-provisioned (for example reference states provided by router vendors), retrieved dynamically from the orchestrator, or verified via out-of-band channels (such as a signed manifest exchange)?

As previously discussed, trust propositions are at the foundation of trust models within CASTOR, and directly influence the resulting ATL calculation in relation to a given trust property. It is therefore crucial to be able to determine the optimal set of evidence based on which a trust proposition can be constructed for a given property. Optimising evidence collection and interpretation will directly influence the accuracy of trust evaluation at the orchestration layer, helping to increase confidence in output ATLs, with the additional benefit of reducing the overhead on both the Trust Model Manager (TMM) (optimal trust models require less computational overhead and space) and the Trustworthiness Level Expression Engine (TLEE) (fewer and more space-efficient trust models will lessen overall computational overhead and result in faster and more reliable ATL computation).

4.5.2 Managing Computational Dependencies and Discounting

Subjective Logic (SL), a core component of the CASTOR trust assessment framework, performs optimally in directed, acyclic trust models. Introducing computational dependencies between trust opinions complicates the application of SL, and in particular, the process of discounting, discussed in more detail in Section 4.3.2. The Global TAF must perform a discounting step of opinions received from the Local TAF based on its own perceived trustworthiness of the relevant Local TAF agent. This opinion encapsulates the Local TAF agent's ability to accurately provide trust evaluations specific to the trust property currently being assessed, for example integrity.

The careful selection of discounting operator is vital in the calculation of an accurate ATL, and difficulties arise given the inherent complexity of managing interconnected dependencies and chains of trust. Opinions are often distributed across referral paths where credibility of a report depends on the trustworthiness of another entity, necessitating a discounting process that is consistent and fit for purpose across sequential operations, such that transitive trust can be accurately encapsulated. Combined with the fact that the Global TAF must also perform a discounting step on opinions received from Local TAF agents, a context-specific discounting operator must be chosen to maximise the accuracy of trust assessment. As previously highlighted, an example of such a discounting operator that CASTOR supports is the Referral-

Edge Path Discounting operator, allowing for meaningful discounting in scenarios where chains of trust are purely indirect.

Another related open question is whether or not the Global TAF's opinion on a Local TAF agent should incorporate details about the freshness of the evidence collected by the Local TAF (for example by verifying that affinity bit updates were reported recently) in order to improve the discounting process.

In addition, coupling the discounting mechanism directly with the specific evidence being reported introduces various computational dependencies which should ideally be avoided. It may be optimal to separate these functions entirely, allowing for a separate mechanism to generate a generalised trust assessment of the source, that the Global TAF can subsequently use as part of its discounting process. This would ultimately simplify the core SL calculation steps.

4.5.3 Modelling Uncertainty

Uncertainty is an inherent component of trust modelling, exacerbated by the heterogenous and dynamic nature of the network environments in which the CASTOR trust assessment framework operates. Perhaps most notably, the evaluation of trust must be performed even in cases where evidence is lacking, disjointed or incomplete. This inherent epistemic uncertainty introduces complexity in the trust modelling process. How can an accurate and meaningful trust evaluation, and its subsequent trust decision, be performed in the absence of a full set of evidence? CASTOR employs Subjective Logic (SL) as the foundational calculus for reasoning about trust-related information and computing trust metrics, allowing for the management of uncertainty.

The opinion (i.e. the primary abstraction in SL), extends classical probability through additional quantifiable parameters, namely belief, disbelief and uncertainty (in addition to an optional base rate). Trust propositions are evaluated into these forms of opinions, ensuring any uncertainty is quantifiable and explicit, before any kind of trust decision is made. This allows CASTOR to formalise trust despite incomplete or contradictory evidence. Various weights can be used depending on the context (e.g. the sparsity of evidence) to fine-tune this process.

An additional layer of uncertainty stems from the quantification and aggregation algorithms used within CASTOR. By using an evidence-to-opinion methodology, the TAF can select the most appropriate quantification model based on evidence type, for example differentiating between deterministic attestations and behavioural heuristics, introducing the implicit management of evidence quality. An appropriate fusion operator can also be selected in the event that the TAF receives multiple opinions on the same proposition, discussed Section 4.5.5.

Furthermore, evidence collected by a TAF can also differ in quality and format. As discussed in the previous section, CASTOR can mitigate the potential impact of a vast array of evidence through the use of discounting. If the reporting source is not perceived to be trustworthy, discounting allows for the weakening of the trust opinion, typically by increasing its uncertainty.

4.5.4 Challenges in the composition of trust propositions to achieve link and path-level trust

Before links and paths can be considered, CASTOR must first be able to aggregate atomic propositions into composite trust propositions that are capable of encapsulating more abstract concepts of trust (such as "High Integrity"). The derivation of composite trust propositions requires the logical combination of atomic trust propositions using logical operators. For example, a device having "High Integrity" may be defined as "Secure boot for Router 1 stands AND Router 1 runtime integrity checks have passed".

However, an open question is the choice of which set of logical operators to use in such a scenario (for example AND and/or OR). It may be the case that the chosen operators are entirely dependent on the context of a specific trust assessment calculation and SSLA, and therefore cannot be assumed as a constant. Aside from which logical operators to use, it is also important to consider the optimal set of trust propositions themselves to be aggregated into a composite trust proposition that fully encapsulates the target trust property.

After atomic trust propositions are received by the Global TAF and combined appropriately into composite trust propositions, the Global TAF must construct opinions relating to links and to entire paths in order to check SSLA compliance. The concept of links and paths introduces several architectural and design questions crucial to allowing for correct trust assessment of higher-level concepts such as path-level integrity. For example, can the algebra of agents be extended such that links and paths can be considered independent trust agents, or should they simply be treated as an evaluation of several atomic trust propositions with an associated decomposition function?

Treating links and paths as independent agents may introduce modelling complexity as well as the implication that a link or path would subsequently be a new independent trust agent capable of forming trust opinions and providing referrals itself. In reality, this is not the case, and it would, therefore, be more accurate to represent links and paths as aggregated logical expressions (again using logical operators).

As an example, a link may be represented through the combination of atomic trust propositions in relation to node-level properties (i.e. “Secure boot for Router X stands”) and link-specific properties (i.e. “The link between Router X and Router Y has high bandwidth”). This approach avoids the unnecessary complication of introducing the link as a stand-alone agent itself. Again, however, it is important to consider which logical operators to use to accurately encapsulate a specific trust property of any given link or path. Furthermore, the decomposition function required to derive the final ATL of a complex proposition must be weighted properly to ensure that trust characterisation is accurate.

4.5.5 Subjective Logic Fusion

It is crucial to implement SL fusion as a means for the Global TAF to derive a final ATL that accurately conveys its opinion of a trust proposition in the case where the Global TAF receives multiple opinions (that may be complementary or contradictory) in relation to the same trust proposition, and, if necessary, additionally incorporating the Global TAF’s own opinion on related supporting evidence. It is important to select the most appropriate fusion operator given the context and collected evidence, in order to mitigate the impact of low quality (and/or incomplete) evidence, or evidence that is submitted maliciously in an attempt to degrade the trustworthiness of a neighbouring node. In general, the fusion operator determines how agreement and disagreement among sources is handled, and various options include:

- Cumulative fusion: Typically preferred when evidence is sparse and complementary, reducing uncertainty by prioritising differing sources that cover differing aspects,
- Weighted fusion: Better suited to evidence that is heavily overlapping (and possibly incomplete),
- Consensus fusion: Draws attention to cases where an entity disagrees with an overwhelming majority, and
- Epistemic fusion: Useful when evidence is largely contradictory, placing priority on uncertainty and disagreement between contributing entities.

Fusion also directly plays a role when considering the composition of links and paths. For example, we must consider how a Global TAF fuses the opinions of each entity participating in a link and/or a path so that it can provide an accurate evaluation of link and path-related trust propositions.

It is crucial that the correct fusion operators and logical operators are chosen given the trust property being assessed, and it will be equally important to have a systematic process to enable this selection for every trust assessment calculation as it cannot be assumed to be a constant throughout the runtime phase.

Chapter 5

Multi-Path Control & Agility for Optimal {*Network, Trust*}-Aware E2E Path Construction

In continuation to CASTOR's capabilities in providing **failure isolation** (at the routing plane) and **explicit trust information management** for end-to-end secure communication, in what follows, we focus on the other core building block towards **flexible and agile route control via trusted, continuously updated paths**: That is the enrichment of the control plane with trust- and network-aware decision making features that allow **multi-path control** - a strong, influential property enabling both senders and receivers to control the end-to-end forwarding path establishment over nodes/segments (*routing subplanes*) that can exhibit the Required Trust Level (RTL) for enhanced resilience, availability; i.e., **data transmission path establishment and maintenance capturing all network and trust requirements from the incoming service intents (requests)**. This is based on the provision of an **optimization layer** for efficiently reacting to both incoming service requests but also to any changes in the network and trust characteristics of the underlying infrastructure and data plane. The goal is to identify the optimal network-management decisions over a set of pre-established path profiles (adhering to different network and trust characteristics) so as to be able to recommend the optimal set of *forwarding paths* featuring the required network agility over routing compute elements that can verifiably guarantee the required (from the SP) level of end-to-end assurance.

This, however, translates to the formulation of a complex {*network, trust*} optimization problem capturing both the **network representation specifics** (including topology and temporal semantics) as well as the **trust characterization** of the entire routing fabric (i.e., routing compute elements) as multiple objectives/constraints that need to be achieved/resolved. In contrast to traditional shortest-path problems (as is the current landscape of most prominent routing protocols including source and segment routing as described in Section 2.2.3), the goal is not to simply minimize a cost function, like distance or delay, but to also maximize the "*trust profile*" of the established path; i.e., forward packets over routes comprising elements equipped with trusted computing capabilities that can guarantee the integrity of the entire communication with specific attention to availability, resilience, privacy. Considering both network and trust properties, this inherently becomes a multi-objective optimization problem: for instance, *minimizing path length or latency while maximizing the minimum trust level along each peering link and, consequently, the end-to-end path*. This, in turn, can directly affect the set of trust relationships to be established between endpoints adding another dimension on the optimization complexity towards **trusted path construction over only those essential entities that need to be part of the path segment** - thus, minimizing also any possible impact on the already formulated path segment compound for the other service intents.

Historically, Border Gateway Protocols (BGP) and LDP were used for deploying Layer-3 and Layer-2 VPN services, between different administrative domains, whereas "IS-IS" or OSPF are used as Interior

Gateway Protocol (IGPs) for fast next-hop convergence as part of the (inter-domain) traffic engineering process. However, as aforementioned, **IGP Prefix-SIDs are used to send traffic on the shortest path to a node or prefix**. This shortest path metric is usually the accumulated IGP path link metric cost. Multiple shortest paths may be identified if the optimization was to be done considering additional network metrics: IGP cost, TE cost, or delay. In this context, Flexible Algorithm enhances such segment routing protocols allowing network operators to deploy multiple logical topologies (Flex-Algos) on top of the same physical network infrastructure. Each Flex-Algo can run its own Shortest-Path First (SPF) optimization objective and set of constraints. Prefix-SID segment lists, with each list belonging to a different Flex-Algo, can be used to steer traffic to the same egress PE loopback address.

It becomes evident that **accounting for a combinatorial consideration of the available trust states (of devices HW and SW) and the available network resources, that are candidate to be utilized for path selection, explodes the design space problem**; especially when this needs to be considered during runtime for unlocking scalable routing updates with high path freshness guaranteeing continuous SSLA compliance. While this is not a new challenge in the topic of trusted path routing in MANETs [132], elevating it into a hyper-heuristic problem optimization extending the set of constraints beyond traditional performance-driven objectives (scalar-represented objectives) to also include trust modelling as a first-class optimization dimension has not yet been formally defined. Considering that ATL semantics cannot be directly measured as scalar values but more as probabilistic encodings of an element's trust state (Chapter 4), incorporating the concepts of *uncertainty* and *disebelief* in the overarching optimization pipeline requires a careful choice of representation.

The remaining of this chapter focuses on fleshing out this set of specific questions that need to be answered for the best modelling of explicit trust representation as a constraint or a set of cost functions in the context of trusted path routing and route control. The goal is to start with the simple modelling of network- and trust-related metrics been considered as separate optimization problems, been solved sequentially (having scalability challenges), so as to then identify the optimal conjunction of the two design space representations. After breaking down the inner-workings of the simple constraint satisfaction problem, in an optimization layer (Section 5.1), we put forth the (3) core research avenues that will dictate CASTOR's optimization design principles summarized in Section 6.2.7 and detailed in D5.1 [41].

5.1 Explicit Path Identification

This section provides (from a global perspective) the **constraint satisfaction formulation for the NP-hard trust-path routing problem** studied in CASTOR: Let the network be modelled as a directed graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. The vertices in v represents the routers in the network topology. The edges in E represent the directional (peering) links connecting the routing elements in the network topology.

Figure 5.1 presents a network topology in which we consider the need to provide service connectivity from the left most router S_1 to the egress router D_1 (endpoints). Based on the key insights that are available by the network telemetry data and the CASTOR-enabled trust evaluations, the Traffic Policy Engine (Section 3.1) is possible to construct a rich topology graph with attributes that characterize different aspects of the routing plane. In their simplest form, such attributes may be associated with node and edge entities in the graph capturing the network and trust state of each node and link. However, based also on the **composite trust evaluations** that were identified in Chapter 4, it is also possible to extend this attribute design space to characterize entire paths or even network segments in a domain. This allows for a **more granular traffic engineering process where CASTOR-enabled trusted path construction and control is considered only at the level of ingress/egress interfaces (thus, allowing ISPs to still control paths internally which, however, might lead to link-level trust or network property violations) or at the next-hop level which leads to a higher level of security and failure isolation at**

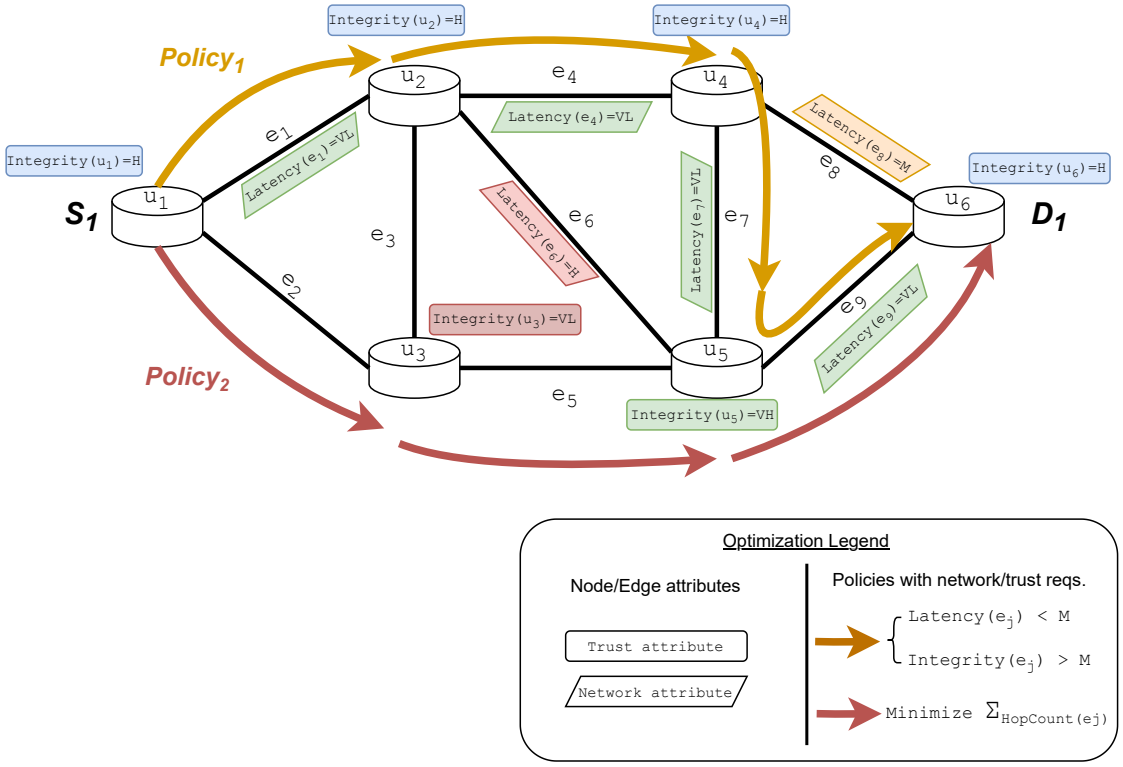

Figure 5.1: Routing topology with network and trust attributes and possible paths satisfying different policies

a link level. In the following example, we consider the latter case where the trust model comprises all relationships and links between physically connected routers. Eventually, this enriched knowledge of the underlying infrastructure layer culminates to the construction of TE policies (e.g., $Policy_1$ and $Policy_2$) that are able to offer fine-grained network and trust-related guarantees in the forwarding plane.

Building on top of the example topology of Figure 5.1, we consider that each node $v_i \in V$ has three trust attributes, namely {Integrity, Resilience, Confidentiality} obtainable individually by functions $Integrity(v_i)$, $Resilience(v_i)$, and $Confidentiality(v_i)$, respectively. Integrity, Resilience, Confidentiality are defined over the enum Very Low (VL), Low (L), Medium (M), High (H), and Very High (VH).

We have to highlight that improving one objective (e.g. shorter paths) may worsen the other (lower trust) so the goal is often to find a set of solutions, Pareto-optimal routes, or apply lexicographic/weighted optimization to balance both objectives.

$$Integrity(v_i), Resilience(v_i), Confidentiality(v_i) \in \{VL, L, M, H, VH\} \tag{5.1}$$

Each directional edge $e_j = (v_i, v_{i'}) \in E$ that connects node v_i with $v_{i'}$ has network attributes Latency, Bandwidth, and HopCount obtainable individually by functions $Latency(e_j)$, $Bandwidth(e_j)$, and $HopCount(e_j)$, respectively. Integrity, Resilience, Confidentiality are defined over the enum Very Low (VL), Low (L), Medium (M), High (H), and Very High (VH).

$$Latency(e_j), Bandwidth(e_j), HopCount(e_j) \in \{VL, L, M, H, VH\} \tag{5.2}$$

Let S and D be set of all ingress (source) and egress (destination) routers, respectively. Let P be set of all solution paths. Let P_{S_x,D_y} be the set of all paths between source $S_x \in V$ and destination $D_y \in V$. A Path $P_z = (v_1, v_2, ..., v_n)$ is in P_{S_x,D_y} if $(v_i, v_{i+1}) \in E \; \forall v_i \in P_k$ and $v_1 = S_x$ and $v_n = D_y$.

We have to highlight that improving one objective (e.g. shorter paths) may worsen the other (lower trust), so the goal is often to find a set of solutions, Pareto-optimal routes, or apply lexicographic/weighted

optimization to balance both objectives.

5.1.1 A Generic Example of Multi-Constraint Optimization Problem Definition

Inspired from the enriched topology graph of Figure 5.1, it is possible to formulate the problem of identifying the optimal paths that satisfy the requirements of a single policy as a constraint satisfaction problem.

As a domain operator, and given a set of ingress and egress nodes, I want to find a near-optimal set of paths that meets all of the following requirements:

- Bandwidth does not fall below Medium.
- Latency does not go above Low.
- Confidentiality is evaluated as Very High
- Hop count is minimal
- Integrity is maximized
- Integrity cannot fall below Medium

Inputs : S and D

Output : P

Objective: Find all $P_{S_x,D_y} \in P \quad \forall\, S_x \in S$ and $D_y \in D$ such that

$\forall P_z \in P\ \forall e_j \in P_z\ \forall (v_i, v_{i'}) \in e_j$

- $Bandwidth(e_j) \geq M$
- $Latency(e_j) \geq L$
- $Confidentiality(v_i) \geq VH$ and $Confidentiality(v_{i'}) \geq VH$
- Minimize $\sum HopCount(e_j)$
- Maximize $\sum Integrity(e_j)$
- $Integrity(v_i) \geq M$ and $Integrity(v_{i'}) \geq M$

5.2 Control Plane: Beaconing for Optimal Forwarding Path Identification

The problem of explicit identification of the optimal set of possible forwarding paths in CASTOR includes several challenges: *All revolving around the careful selection of representation semantics that can allow the convergence of a large attribute design space (capturing all network- and trust-aware corner cases) with high degree of scalability so as to efficiently manifest into deployable TE policies, providing meaningful and enforceable trust adaptive to topology changes and any routing updates needed for maintaining path control; i.e., minimizing future service migrations due to workload changes.*

Challenge: This leads to the first central question that needs to be examined: "***How can trust be represented and integrated into CASTOR's multi-objective routing optimization?***" This challenge

stems from the fact that while network-related metrics can be represented as scalar values, this is not the case with trust-related information that go beyond (binary-expressed) properties like resilience and/or availability. The node- or link-level trust characterization, based on such a model, results in probabilistic (projected) values trying to engrain the level of uncertainty and/or disbelief as part of the overall representation to which the monitored evidence can support the extract trust proposition - especially in the case of evidence-based trust assessment theory such as the Subjective Logic employed in CASTOR.

CASTOR Convention I: The first approach is to pass the (SL-based) expressed trust opinions generated by the Trust Assessment Framework directly into the optimization stage. This preserves all trust semantics—the triplet (b, d, u) representing belief, disbelief, and uncertainty—and, therefore, allows the (explicit) optimal routing configuration based on the direct interpretation of the monitored trust-related evidence. In other words, it embeds the uncertainty representation as part of the optimization layer and not as an add-on information given by the Trust Assessment Framework: Rather than forcing a trust assessment decision, in this case trust uncertainty, discounting and/or fusion are considered different dimensions of the optimization problem which evolve over time. This allows the blending of evidence - representing both belief and disbelief while leaving room on how link-level trust characterizations can affect the trust level of the overall path segment. This allows us to adjust the routing topology dynamically as more data becomes available. Such an approach aligns naturally with the evolving trust model encountered in network topologies and would, in principle, allow **CASTOR to select paths that account not only for the estimated trustworthiness of network elements but also for the reliability of the underlying measurements.** However, this expressiveness comes with substantial computational overhead. Full trust opinions are multi-dimensional, they require complex fusion or discounting operations when aggregated across a path, and they do not map efficiently onto the scalar objective terms needed by multi-objective routing solvers or QUBO/Ising-based formulations. *Given that the trusted routing problem is NP-hard and CASTOR aims to support real-time path computation, maintaining the full Subjective Logic structure inside the optimizer would significantly limit scalability.*

CASTOR Convention II: A second, more practical approach is to project each probabilistic trust opinion into a single scalar value (e.g., projected probability) before it reaches the optimization layer. This creates a compact numerical trust indicator that retains the essential meaning of the underlying trust assessment while being far easier to incorporate into multi-objective formulations. Such scalar values can be aggregated along candidate paths, combined naturally with latency or bandwidth metrics, and encoded efficiently in binary optimization models used by quantum or quantum-inspired solvers. By reducing the dimensionality of trust information, scalar projection improves computational efficiency while preserving the connection to the evidence-driven evaluations of Chapter 4. Nevertheless, the projection inevitably reduces expressiveness, eliminating explicit modelling of uncertainty or disbelief. Careful calibration is therefore necessary to ensure that the chosen projection method aligns with operator goals, risk posture, and service requirements. *Despite this simplification, this approach offers a strong balance between trust fidelity and runtime efficiency, making it particularly suitable for real-time or near-real-time optimization scenarios targeted by CASTOR.*

CASTOR Convention III: The third approach relies on the categorical path profiles, supported by a single or a combination of ASes (Section 3.1), capturing those domains where path segments with a uniform trust representation can be formed. This, essentially, translates to path segment formulation over routing compute elements that exhibit equivalent network and trust capabilities; i.e., achieve Medium (M) Integrity with High (H) Resilience over Low(L) latency communication channels. Here, trust is not optimized numerically but is instead used as a policy filter: only paths that satisfy the required trust thresholds—for example, Integrity at least Medium or Confidentiality at least High—are considered eligible before network-level optimization begins. This method integrates seamlessly with SSLA-driven service provisioning, reduces the size of the search space significantly, and supports rapid decision-making. **However, by treating trust categorically rather than numerically, this approach cannot capture fine-grained trust differences and may exclude paths that narrowly fall short of a threshold even when they would otherwise offer superior performance.** Consequently, while trust profiles ensure strong policy compliance

and excellent runtime behaviour, they provide limited flexibility for nuanced trust-performance trade-offs.

In summary, the three approaches offer different balances between trust expressiveness and computational feasibility. Using full Subjective Logic opinions preserves maximum detail and remains closest to the envisioned trust semantics (Chapter 4), but introduces significant complexity for multi-objective routing. Projecting opinions into scalar trust values provides a more compact form that can be integrated into the aforementioned optimization model instances (Section 5.1), while still maintaining continuity with the underlying evidence in TAF. Using categorical trust profiles offers a straightforward way to enforce SSLA requirements and reduce the search space, though with reduced granularity. Each approach therefore carries its own strengths and considerations, and their suitability must be examined in light of CASTOR's ambition to support trusted path computation under real-time or near real-time conditions. The current analysis does not prescribe a single solution; rather, it identifies the key trade-offs that must be explored further as CASTOR advances toward a fully trust-aware optimization framework.

Challenge: This brings us to the second major question: ***"What computational techniques and hardware (classical, quantum) are suitable for solving the trusted routing problem, and how mature are they for real-time use?"***

CASTOR Convention IV: This challenge is essentially directly related to the **efficiency, scalability and extensibility** requirements, described in Section 2.1 so as to overcome existing scaling issues in cases of network fluctuations, where routing protocol convergence can require minutes. When this is considered together with security and high availability, resilience properties, it comes with even a higher cost resulting in lower efficiency and further diminished scalability. High performance and scalability with high degree of trust are required for viability in the current economic environment. We, therefore, explicitly seek high efficiency in **multi-path generation**: The optimization layer is expected to produce multiple candidate path segments per path profile. Consequently, the number of solutions to be maintained, evaluated and validated increases significantly - especially considering also that this process might need to be triggered every time there is a new service workload to be deployed or change in the underlying network topology (e.g., addition on removal of a routing element). Therefore, the question remains on *how can CASTOR framework scale the optimization process to handle multiple valid paths per profile, while maintaining computational efficiency and minimizing redundant evaluations?*

A novel approach that CASTOR aims to explore is the employment of Quantum Annealing (QA) mechanisms (Section 2.2.9). QA is well suited to combinatorial optimization and aligns naturally with a QUBO or Ising formulation of the routing task. However, despite its conceptual attractiveness, current QA hardware remains limited in scale, connectivity, noise characteristics, and energy landscape control. As a result, existing annealers cannot yet support real-time or near real-time operation, particularly in the context of large, dynamic, and multi-domain network environments such as those targeted by CASTOR. For this reason, Quantum Annealing cannot form the primary execution path of CASTOR's routing optimization engine. Nevertheless, the conceptual framework of a QUBO/Ising formulation remains extremely valuable. Once CASTOR defines the trusted routing problem's cost function, combining both network and trust dimensions into a unified mathematical representation, this formulation can be solved not only by quantum hardware but also by a broad family of quantum-inspired and classical optimization algorithms. These approaches offer a far more mature and scalable technological base for real-time operation.

Quantum-inspired methods such as Simulated Bifurcation, Coherent Ising Machines, can process large QUBO instances efficiently using classical hardware while drawing on principles inspired by quantum physics or nonlinear dynamical systems. Such solvers are already capable of handling thousands or tens of thousands of variables with millisecond-scale iteration times, making them strong candidates for CASTOR's real-time optimization needs. Alongside these, classical heuristic or meta-heuristic approaches remain important baselines that provide robustness, transparency, and ease of integration into the overall architecture.

In this context, QA will still be investigated within CASTOR, but with a research-oriented role rather than an operational one. QA experiments will allow the project to benchmark current and emerging quantum

hardware against quantum-inspired and classical methods, evaluate scalability constraints, and identify potential future pathways as quantum technology matures. This comparative analysis will enrich CASTOR's understanding of the trade-offs between computation time, solution quality, problem size, and hardware requirements across different optimization paradigms. Overall, CASTOR will adopt a pragmatic and forward-looking approach: formalize the trusted routing problem as a QUBO instance, explore quantum-inspired and classical algorithms as viable real-time solvers, and evaluate Quantum Annealing as a research instrument for comparative assessment. This strategy ensures that CASTOR remains aligned with current technological realities while positioning the system to leverage future advances in quantum computing.

Challenge: Finally, it is important to consider the last fragile aspect in the overall traffic engineering process where this optimization pipeline needs to be engrained: ***"How the optimization process can support fine-grained path control without inhibiting ISPs from traffic engineering?"***

CASTOR Convention V: This challenge unveils a complex and multi-faceted problem in traffic engineering: On the one hand, many service intents (as will be the case of the envisioned Aerospace targetted use case detailed in Chapter 8) require link-level security and trust guarantees, during the SSLA conformance checks, which might limit the path control maintained by the ISPs as part of a path segment. This, in turn, might limit the applicability of the offered solution. On the other hand, keeping the optimization process targetted at recommending path solutions at the level of ingress/egress interfaces (path segment establishment) while minimizing the complexity of the overall process, it might offer limited application due to the inability of converting these path segments into enforceable TE policies. For instance, consider the example illustrated in Figure 5.1: A path segment traversing over nodes u_1, u_2, u_3, u_5, u_6 while violating some trust properties at a link level (i.e., between nodes u_3 and u_5), it still might capture the required network and trust properties at a path segment level - because the exhibited trust level between (for instance) nodes u_5 and u_6 is calculated as Very High, thus, discounting the Low integrity level of the previous link in the hop. However, even if this path level control allows transparency to the ISPs, it is not as obvious on how to translate it to enforceable routing policies especially considering the Segment Routing paradigm. **What is the optimal set of affinity link colours or SR-TE policy colours that capture such level of abstraction without affecting the solutions already converged across the other path profile segmentations?** Each path profile defines a unique optimization design space. However, these profiles are not independent. For example, path optimized under one profile may share resources with another optimized path. Therefore, the question remains, how can the CASTOR framework efficiently manage the interdependencies between multiple path profile specification without compromising on the solution quality or convergence time? This challenge is particularly relevant because it may hinder the parallelization of optimization search for multiple path profiles, severely affecting the scalability of the optimization engine.

This challenge is further aggravated considering the fact that when two or more path segments are computed on the same set of link state (both network- and trust-related information), it is possible that the resultant paths will compete for limited resources within the network. This might result in success for only the first path segment to be calculated (by the optimization layer), or it might be the case that no path can be established. Batch processing, back-off times, recommendation of alternate paths, and crankback can help mitigate this sort of problems but it is not straightforward as to how they can be integrated into the overall optimization process.

Together, these challenges define the scope of CASTOR's ongoing research in developing a scalable, adaptive, and trust-aware optimization framework for explicit path identification. CASTOR, by addressing them, would be the first frame work to produce path recommendations that remain both valid and trustworthy in highly dynamic, multi-domain networking environments.

Chapter 6

CASTOR Conceptual Architecture and Functional Components

6.1 CASTOR Conceptual Architecture

CASTOR's trust-aware approach to traffic engineering requires a comprehensive framework that spans the compute continuum, addressing both the routing and control planes, while encompassing lifecycle management of network elements and ensuring service assurance guarantees. To better describe all the different aspects and technical details of the CASTOR framework we, first, define four concrete phases that characterize all relevant network entities throughout their operational lifecycle (Figure 6.1): i) Preparedness, ii) Service Registration, iii) Proactive and iv) Reactive. As can be seen from this high-level figure, the four phases are not strictly sequential. While some aspects focus on the the service lifecycle, others address the configuration and behaviour of the network topology.

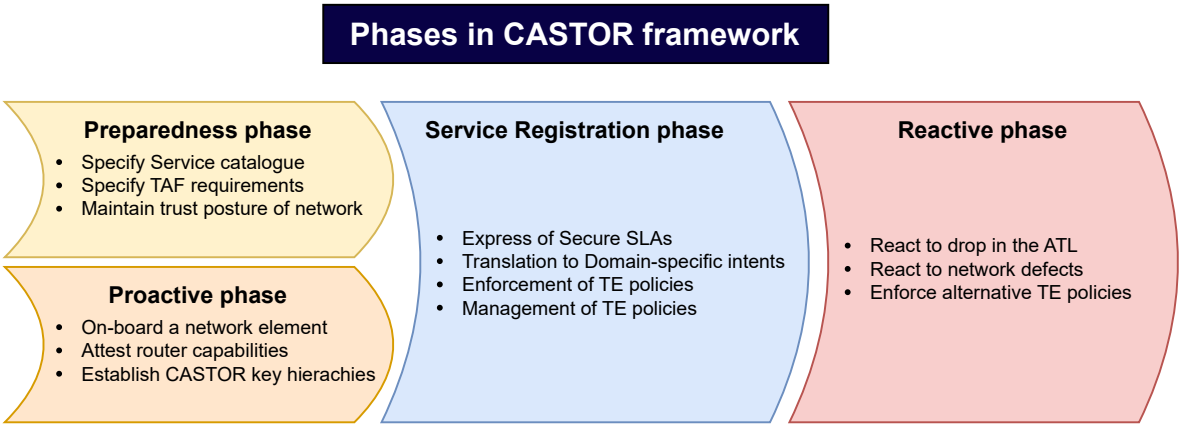

Figure 6.1: High-level phases in CASTOR framework

The **Preparedness phase** relates to all the configurations that allow an OSS environment to fulfil new service requests and to guarantee the required level of assurance. This phase covers aspects that span across the compute continuum, hence, part of it assumes that the network topology - i.e., the infrastructure layer - is already in place. Starting from the preparation at the orchestration layer, it is crucial for any domain operator to specify the envisioned service catalogue. In CASTOR, this involves the specification of the different path profiles that may satisfy varying levels of network and trust requirements. The preparedness phase expands, also, on the mechanisms that allow CASTOR to identify the optimal paths that accommodate the offered service catalogue, as well as the means to enforce the selected trust-aware routing policies in the infrastructure layer. Finally, to ensure the assurance of all registered services, the preparedness phase covers the provisioning of the probing mechanisms that need to be in

place in the network elements, allowing the continuous and dynamic evaluation of the (S)SLA compliance of an established service.

Once all elements of a domain vertical are prepared, the Service Orchestrator is able to serve incoming service requests. In this context, once the critical service-level objectives are negotiated and agreed between the relevant parties, the **Service Registration phase** describes how the CASTOR framework is able to realize the service placement when a new request of (network and trust) intents is expressed by a service provider. This phase expresses the flow of converting the high-level service into domain-specific requirements and the enforcement of the appropriate traffic engineering policies at the infrastructure layer. This allows the end-to-end interconnectivity that allows the service to user a network path that satisfies the network and trust related objectives established in the (S)SLAs.

The instantiation and configuration of the orchestration layer — in the context of the CASTOR preparedness phase — allows the Service Orchestrator to proactively onboard additional routers dynamically as well as adapt the already recommended paths accordingly. First, the **Proactive phase** captures the interactions between the newly incoming router and the Service orchestrator in order to realize the secure onboarding process through the secure transmission of the necessary guarantees pertaining to the correct state of critical router functionalities. At the same time, similar guarantees are exchanged in the data plane between network elements, thus allowing for the discovery of possible secure links across the topology. All these actions provide the necessary evidence to the orchestrator in order to successfully enrol the new node in the topology and include it in the envisioned paths to be established. Secondly, it is also possible that the current version of the infrastructure topology is unable to meet the expected requirements. For example, it is not possible to establish a path with the necessary security guarantees when all routers comprising the infrastructure layer are running with a specific firmware vulnerability. Consequently, the proactive phase ensures that the necessary security mechanisms are also enforced in the infrastructure layer in order to enable the realization of particular types of (S)SLAs.

Finally, the **Reactive phase** focuses on the aspects that the CASTOR framework offers once a service is deployed and served through the underlying infrastructure. Throughout the service lifecycle, there are numerous changes that may cause an established traffic engineering policy to no longer satisfy the specified (S)SLA terms. There are different types of reaction strategies to cope with these situations, depending on the specific use case. Regardless of the communication fabric on top of which a path is established, there are inherent mechanisms that the routing protocols provide to allow a network to self-heal upon a dramatic change in the topology (e.g., continuous re-calculation of the shortest path in a topology based on an IGP metric, candidate paths and on demand next-hop configurations in SR TE policies). Even though these mechanisms will be examined throughout the instantiation of the CASTOR architecture, the concept of the CASTOR reactive phase refers to the capability of the overarching framework to adapt to the latest changes and enforce new policies—if needed—to ensure the continuous adherence to the established SLAs. Overall, in the context of the CASTOR framework, we distinguish the possible runtime changes according to their root cause: either *network-driven* or *trust-driven*. As analyzed in the following subsections, the former category focuses on the reaction in the traffic engineering policy provisioning when a network element gets attached to or detached from the topology, or any network-related disruption occurs (e.g., traffic congestion in one the links of a path, physical damage in an Ethernet connection between two network elements). On the other hand, the latter category examines compromises in the agreed security guarantees due to a significant decrease in the trustworthiness level of a network element (on a link or even on an entire path). In this context, it is the CASTOR framework that enables the calculation of the possible strategies that allow the re-establishment of trust in the network topology, e.g., by enforcing new security policies over the existing paths or by enforcing completely new traffic engineering policies.

Figure 6.2 presents the high-level blueprint of the CASTOR architecture. This figure shows the core CASTOR technologies and their placement across the compute continuum, shaping the path towards trust-aware Traffic Engineering policy enforcement. In the following, we delve into the realization of each phase of the CASTOR framework, showcasing the flow of operations between the various CASTOR

artifacts deployed from the network elements up to orchestration plane.

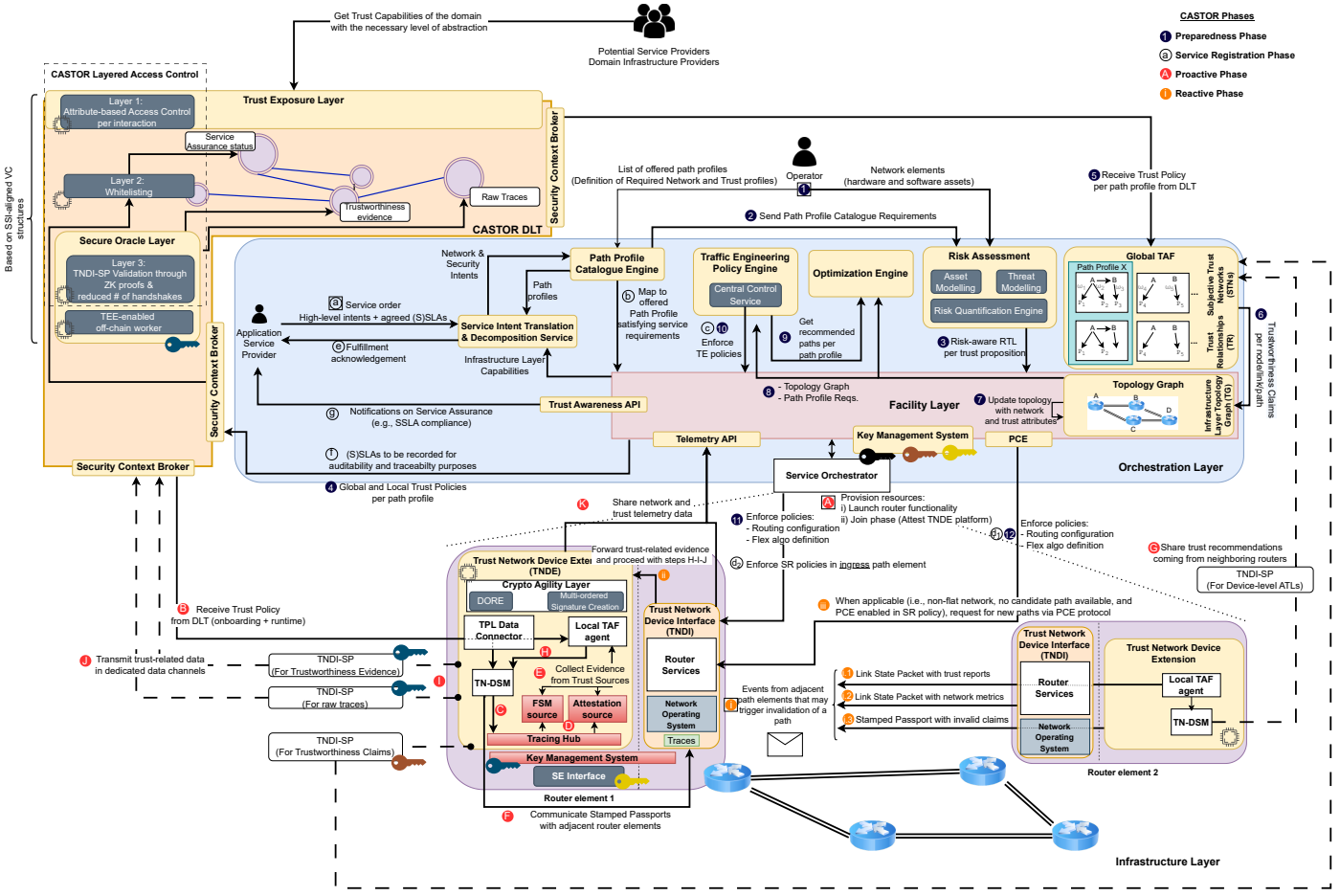

Figure 6.2: CASTOR high-level architecture

6.1.1 Preparedness phase

CASTOR's preparedness phase covers the provisioning of the control plane, enabling the specification of the service catalogue with its translation into a set of traffic engineering policies that satisfy both network- and trust-related requirements. In the context of CASTOR, the service catalogue of the resources managed by the Service orchestrator is equivalent to the types of traffic engineering policies that can be offered by the operator with respect to network and trust capabilities. Consequently, this allows the operator to define the path profile catalogue (Step 1, Figure 6.2) which captures the various network and trust capabilities that will be offered by the underlying infrastructure layer. This is also intrinsically linked to the telemetry data (i.e., network and trust attributes) that need to be measured during the operational phase of the topology. Along with the offered path profiles—capturing network and trust attributes—the operator specifies the Trust Policies that need to be enforced in the Local TAF agents (as well as the Global TAF) to ensure that the infrastructure layer is trustworthy with respect to the security guarantees that the infrastructure needs to attain when serving the security SLAs (SSLAs) upon a service request. The derivation of the appropriate Trust Policies that need to be assessed is dependent on the security posture of each network element to be considered in the topology (Step 1). Hence, based on the envisioned trust requirements captured in the path profile catalogue (Step 2), the Risk Assessment component is able to derive the minimum required trust level that needs to be satisfied by a router element for it to securely participate in any of the envisioned path profiles (Step 3). The derived Trust Policies are recorded on the CASTOR DLT (Step 4), allowing onboarding routers to pull the device-applicable Trust Policy in order to perform the required trust evaluations during the operational phase.

The aforementioned steps can take place as part of the design phase of the orchestration layer, in the sense that they do not require the infrastructure topology in place. However, the remaining steps of the preparedness phase require the router elements to be securely onboarded and configured so that telemetry data arrives at the orchestration layer. First, this refers to the establishment of network telemetry channels that provide fresh evidence from the underlying infrastructure layer to the Service Orchestrator. At the same time, through the enforcement of the Trust Policy in the overall CASTOR Trust Assessment Framework (Steps 5), it is possible for the Global TAF to calculate the trust propositions that characterize the security posture of the infrastructure layer (Step 6). All these pieces of information, namely the topology graph and network telemetry data from the Service Orchestrator and the trust telemetry data from the Global TAF, allow the construction of a mirrored Topology Graph maintained by the CASTOR Facility Layer (Step 7). The telemetry data is reflected in the Topology Graph in the form of network- and trust-related attributes that may characterize any element in the topology: a node, a link, or an entire path. This CASTOR-enriched topology information provides guidance on the elevation of runtime trust characterization into routing configuration that can be applied via the Service orchestrator. For example, through traffic engineering practices, labelling a network link can be associated with the capabilities that the participating routers exhibit. This enables the specification of routing policies that include instructions to avoid any link that is characterized by a specific label (e.g., signalling that the confidentiality trust property cannot be achieved). But most importantly, the Topology Graph provides all the necessary information that allows the identification of the optimal paths in the infrastructure layer that are able to accommodate the requirements of the offered path profiles. The trust-enriched insights in the Topology Graph allow central control services - such as the CASTOR Traffic Engineering Policy Engine to search for optimal paths that can satisfy the supported path profiles in the service catalogue. In order to do so, it needs to trigger the Optimization Engine (Step 8) which is responsible for receiving the latest topology posture - i.e., Topology Graph - and provide a set of recommended paths that are able to accommodate all path profiles in the catalogue (Step 9); so that the Service orchestrator can deploy them when a respective service order request arrives (the transition from a service order to a path profile is explained further below in the Proactive phase). The output of the Optimization Engine is leveraged by the Traffic Engineering Policy Engine (Step 10) which is able to provide the Facility Layer with all the traffic engineering policies that need to be enforced so as to accommodate the corresponding path profiles; either on demand or prior to any service order request. Depending on the nature of the prepared traffic engineering policies and the requirements of the operator, the Facility Layer may trigger the Service Orchestrator to enforce the policies (Step 11), e.g., by setting up BGP extended communities, defining a Flex Algo definition satisfying the characteristics of a path profile, and enabling a dynamic Segment Routing Traffic Engineering policy to contact the PCE for the path provisioning. Alternatively, it may leverage the CASTOR-extended PCE component deployed in the control plane in order to enforce the requested Segment Routing Segment ID list in a semi-automated manner (Step 12).

6.1.2 Service Registration phase

The Service Registration phase captures the processing of an incoming service request up to its fulfilment. As shown in Figure 6.2, the first concept refers to the translation of high-level intents coming from the service provider to concrete, domain-specific requirements that can be enforced in the infrastructure layer (Step a). Even though this intent-layer logic is not considered part of the CASTOR framework, it is captured in the overall architecture for completeness capturing also the knowledge that needs to be shared from the lower orchestration layers - e.g., in terms of the path profiles and the topology network and trust capabilities. These concrete requirements allow the association of the established (S)SLAs with a specific path profile that can be provisioned in the domain (Step b). Based on the work performed as part of the preparedness phase, the Service Orchestrator is able to spawn a new path based on the recommended policies that are maintained by the Traffic Engineering Policy Engine. The latter component has all the necessary information (from the Facility Layer) to make a decision (Step c) on the policies

that need to be enforced in the infrastructure layer. This involves the configuration of the appropriate network elements either via the available management interfaces of the Service Orchestrator (Step d_1) and the actual enforcement of the traffic engineering policies via a path computation element (PCE) interface (Step d_2). Of course, in the case that there is no available recommendation to accommodate a new service request, the Traffic Engineering Policy Engine may trigger the Optimization Engine (Steps 8-10) to re-calculate a fresh set of recommendations considering the latest network and trust profiles reflected in the Topology Graph. Once the path is provisioned, the Orchestration layer is able to provide an acknowledgement to the Service Provider notifying them on the availability of the requested path. After this step and throughout the lifecycle of the service, CASTOR allows the Service Orchestrator to continuously monitor the (S)SLA compliance when it comes to both the network- and trust-related service-level objectives. This status along with additional information pertaining to the service assurance (e.g., level of assurance or trust capabilities characterizing the service/) are offered to the Service Provider via the Trust Awareness API (Step f).

6.1.3 Proactive phase

As shown in Figure 6.2, in the context of preparing the enrolment process of a router element in the topology, the operator needs to specify the minimum requirements that a candidate router element needs to adhere to in order to successfully enrol the topology. These requirements mainly revolve around the security guarantees that a router (i.e., a Trust Network Device Interface as further explained in this chapter) need to demonstrate when enrolling in the topology. The derivation of these requirements is coupled with the risk posture of each router element, i.e., hardware and software characteristics—as an isolated asset but also as part of the envisioned network topology (Step 1 of the preparedness phase). An additional aspect that needs to take place prior to the enrolment process of a router element relates to the ability to evaluate the specified security requirements. In the context of CASTOR, the Trust Network Device Extension attached to a network element is able to process collected traces from the target router functionalities. The available evidence and the local trust evaluations enable the overarching Trust Assessment Framework to identify—with an adequate level of uncertainty—whether the specified requirements are satisfied. Consequently, the final step of this design phase requires the specification of all the necessary aspects—in the form of Trust Policies—that need to be enforced across the compute continuum so as to enable the derivation of the final trust decision that will dictate whether a candidate router element can enrol the topology.

Once all policies are in place, it is possible for the operator to serve any enrolment requests when a router element wants to join the topology. In the context of managed networks, this process can be realized by the orchestration layer. Specifically, as shown in Figure 6.2 (Step A), the Orchestration Layer interacts with the candidate router element to request the necessary guarantees (e.g., attestation evidence) that the router functionality has been securely launched including the CASTOR artifacts that are deployed in tandem with the router element. The successful execution of this process grants access to the router to proceed to the next phases of the enrolment process. On the one hand, this unlocks the router to access and enforce the corresponding Trust Policy to the Local TAF agent from the CASTOR DLT (Step B). This provisions the collection of the relevant evidence from the underlying Tracing Hub (Step C) and the processing of the collected traces by the relevant Trust Sources (Step D). Based on this evidence (Step E), the Local TAF agent is able to perform the trust computations on a local, in-router level and share the perceived trust reports with the control plane. Specifically, data that are required for the derivation of the final ATL values by the Global TAF are transmitted through secure and confidential TNDI-SP channels (Steps H-I-J), while the final local Trust Decisions are shared via extended telemetry probes to the Telemetry API (and the Service Orchestrator) for visualization and reporting purposes (Step K). In addition, depending on the Trust Policy, the candidate router is able to share trustworthiness evidence with the neighbouring routers (Step F) to allow the establishment of provisional links across the topology. Based on the concepts of IETF's Trusted Path Routing concepts, each router is able to share attestation

quotes with their network adjacency (i.e., neighbouring routers) attesting to the secure boot of the router. On this basis, CASTOR aims to enrich these capabilities by extending the attestation functionalities to cover runtime characteristics as well, enabling both the establishment but also the continuous evaluation of trust during the operational phase of the network. All available evidence is forwarded to the Local TAF agent of the receiving routers, and all trust reports are securely shared to the Global TAF deployed in the orchestration layer through secure and confidential TNDI-SP channels (Step G); allowing the orchestration to complete the final trust evaluations and decide whether the candidate router can successfully enrol the topology. Overall, it is worth noting that a key aspect in the preparedness phase relates to the provisioning of the necessary cryptographic keys at the in-router TNDE side, so as to enable different communication interfaces outside of the router element. For instance, as illustrated in Figure 6.2, there are different cryptographic material for establishing and managing the confidential TNDI-SP channels (the blue key icon for establishing different TNDI-SP data channels between the TNDE and the CASTOR DLT, whereas a different brown key is provisioned for the TNDI-SP channel between the TNDE and the Global TAF for the transmission of trustworthiness claims).

Finally, as part of the proactive phase, we take into consideration any actions that allow the infrastructure layer to serve one of the offered path profiles. Specifically, it is possible that the existing onboarded topology is not able to serve one of the intended path profiles, e.g., due to insufficient assurances on the integrity or due to a cascading attack that may leverage a vulnerable ingress router to affect its neighbouring ones. Consequently, as part of the proactive phase, we include the enforcement of security policies that will help reduce the risk posture of the underlying topology. Such security policies may refer to the application of a security control that mitigates a critical vulnerability or to the extension of the target environment that is continuously monitored by the CASTOR Trust Network Device Extension in order to collect more evidence that can provide stronger guarantees that the router is behaving as expected. Consequently, even though the problem of seamless security control enforcement is considered orthogonal to the objectives of the project, CASTOR provides all the core enablers that allow the orchestration layer to achieve a high-level zero-touch service management.

6.1.4 Reactive phase

The objectives of the CASTOR framework go beyond the establishment and continuous evaluation of trust-aware traffic engineering policies. In fact, one of the core objectives of the CASTOR framework is the re-establishment of trust and adaptability of the instantiated path recommendations when a change is detected in the underlying topology. Updates to established traffic engineering policies are typically triggered by two main categories of events, based on their source: network-driven changes and trust-driven changes. Each category has different characteristics requiring specific reaction strategies by the CASTOR framework to ensure a seamless transition to alternative traffic engineering paths without service disruption. That is, CASTOR's goal is to preserve the (S)SLAs of the actively deployed application services.

There are two lines of defence that are envisioned in CASTOR. First, CASTOR engrains trust characterization into traffic engineering policy provisioning, as it allows to attribute network links with traffic engineering metrics that are related not only with network properties (e.g., low-latency, network cost) but also trust properties (e.g., level of assurance capturing integrity aspects of the target router environment). Consequently, dynamic traffic engineering policies can accommodate network objectives while also respecting trust-related constraints (e.g., run shortest path first algorithm while excluding any link with no integrity and confidentiality guarantees). In addition to the attribution of links (or nodes) depending on the network and trust characteristics collected in the CASTOR Facility Layer - forming the Topology Graph - the operator can enhance reaction strategies as part of the enforced traffic engineering policies. For instance, in the context of Segment Routing Traffic Engineering, there can be policies with multiple candidate paths as a restoration mechanism, allowing one (e.g., either the operator or a controller entity such as a PCE) to trigger a path switchover, leading to the headend router to divert its traffic through an

alternative path; or even enable local, fast re-routing capabilities so that the intermediate path elements can decide on their own a viable solution to minimize the time it takes for the notification to reach the ingress router. In this first line of reaction strategies, the CASTOR mechanisms in the orchestration layer provide all the mechanisms that allow operators to define the traffic engineering policies that can best accommodate the network and trust guarantees reflected in the corresponding path profile.

The second line of reaction strategies refers to the actions that the CASTOR orchestration layer can enforce directly through the PCE component. Regardless of the source of change in the infrastructure layer, the Topology Graph is able to capture all attributes of the network (through the Telemetry API of the Network Orchestrator) and trust (through the Global TAF evaluations). As already discussed in the proactive phase, the CASTOR TNDE is able to capture any change pertaining to the router element that it is deployed. In addition, such events may flow across the data plane (Step i). This can be either in the form of the Stamped Passports (Step i.3) or in the form of Link State Packets. Regarding the latter category, the Link State Packets may include notifications about a network-driven change (Step i.2) in the topology, e.g., a link has been disconnected. In addition, CASTOR examines how to leverage different Link State Packet extension mechanisms to encapsulate trust-related information flowing across the infrastructure layer (e.g., IETF 8665 on OSPF Extensions for Segment Routing, or IETF 9086 on Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing BGP Egress Peer Engineering). The availability trustworthiness evidence (e.g., Stamped Passports) and trust reports (e.g., through Link State Packet extensions) in a router element (i.e., TNDI) allow the corresponding Local TAF agent to form trust relationships - and hence opinions - about its neighbouring routers (Step ii). This sets the scene for the construction of a trust plane, whereby Local TAF agents share enriched information with the Global TAF, focusing on both their trustworthiness and that of their adjacent entities (Steps H–I–J). A change in the mirrored Topology Graph triggers the Optimization Engine to revisit its existing recommendations and revise them to accommodate the envisioned network and trust capabilities in each path profile based on the latest view of the infrastructure layer. Depending on the output of the optimization engine, the CASTOR orchestration layer may enforce new traffic engineering policies through the management interfaces to the corresponding router elements (control-plane optimization) or download (Step iii) new set of paths in the existing policies via the PCE (data-plane optimization).

6.1.4.1 Network-driven

This category constitutes the legacy events that may impact a network topology and, consequently, the well-established types of SLAs. Existing routing protocols are capable of detecting such changes, e.g., a network interface in a router's interface becomes inactive, or even reacting to such events with automated steering and fast re-routing capabilities. Such events may arrive to the control plane either via the data plane updates, e.g., through link state updates that are reflected in a controller entity, or via control plane interfaces, e.g., through the provisioning of network telemetry data from the infrastructure layer. In the CASTOR architecture, all these metrics are mirrored in the Topology Graph that is maintained in the Facility Layer. As any of the network-driven events is reflected in the Facility Layer, no matter if occurring in the nodes and links comprising the network topology or in any of the network attributes characterizing the existing topology, the Optimization Engine can receive the latest view of the underlying topology and adjust its recommended paths accordingly.

6.1.4.2 Trust-driven

CASTOR provisions the security guarantees that each router needs to exhibit throughout its lifecycle, in order to participate in the network topology and serve application workflows. Expressing, also, the objectives of the established SSLAs, these guarantees are encoded in (domain-specific) Trust Policies that dictate i) the raw traces that need to be collected, ii) the security properties that need to be evalu-

ated, iii) and eventually the trust propositions that need to be assessed in order for the overarching trust assessment framework to derive a trust decision on the assurance of the deployed services.

During the runtime behaviour of a router element, the CASTOR Trust Network Device Extension allows the continuous monitoring of the target security properties that need to be assessed as per the enforced Trust Policy. When a violation is detected, e.g., by the Attestation Source or the Finite State Machine source, this allows the Local TAF agent to form trust opinions on atomic trust propositions that are tailored to the evidence that are collected. Depending on the enforced Trust Policy, the notification of a violation detected by the CASTOR TNDE enablers may require the recording of the relevant evidence to the CASTOR DLT for auditability and post-processing purposes. Through the provision of the necessary TNDI-SP channels, the raw traces - pertaining to an observed violation detected by the (in-router) TNDE - are recorded in a confidential and privacy-preserving manner so that only authorized entities (e.g., hardware vendor of a router, operator of the infrastructure layer) can access and process them.

Once the local trust evaluations have been performed, both the computed trust opinions and the necessary evidence are shared with the Global TAF in the orchestration layer in a secure and confidential manner through the CASTOR TNDI-SP. This information allows the Global TAF component to re-evaluate the trust propositions that help form a trust decision pertaining to the fact that the deployed application workload is served over a path that adheres to the trust-related defined in the offered path profile that serves the established SSLA. Upon a detected violation, the Global TAF may evaluate that a link (or a path) is unable to continue offering the trust requirements of a particular path profile. This trust (re-) evaluation is reflected in the routing policies and also in the updated Topology Graph to be used by the Optimization Engine. This will allow the Optimization Engine to offer a new set of recommended paths for the particular path profile, excluding the nodes/links that are related to the detected violation. These recommendations are converted into policies through the Traffic Engineering Policy Engine. Depending on the operator's requirements, the resulted policy can be enforced via the network orchestrator through legacy management interfaces (e.g., RESTCONF session) or in a semi-automated manner via the PCE.

6.2 CASTOR Functional Components

Name	Main purpose
Orchestration Layer (and above)	
Service Intent Translation & Decomposition Service	Given an incoming service request, it translates the received high-level set of Intents to Service Level Agreements covering network (typical SLAs) and security/trust (Security SLAs; SSLAs) requirements.
Path Profile Catalogue Engine	Maps service requests (and SSLAs) to one of the offered path profiles.
Facility Layer	Manage/Serve (CASTOR- and application-related) requests from/to the different orchestration layers.
Service Orchestrator	Manage resources (e.g., Spawn new VM for a vRouter function).
Telemetry API	Interface for collecting trust- and network-related telemetry data from the operational environment (i.e., network).
Topology Graph	Maintains a view of the trust- and network-profiles associated with the underlying network topology.
Trust Awareness API	Exposes interfaces to provide meaningful information with respect to the Service Assurance to the corresponding Service Provider.
Path Computation Element	Implements the Path Computation Element Protocol and enforces the SR policies realized by the CASTOR framework.
Risk Assessment Engine	Calculates the risk graph of the observable network segments taking into consideration the threats that can affect the router elements (threats related to hardware/os/-software stack); and contributes to the derivation of the risk-aware RTL logic to be enforced in the Global and Local TAF agents based on the offered path profile catalogue.

Global TAF	Assesses the trust profile of all the observable network segments based on the available evidence (i.e., Local TAF reports and additional trustworthiness evidence from each router).
Optimization Engine	Given the offered path profile catalogue and a snapshot of the (network and trust) characterization of the network segments, it recommends a set of optimal (primary and backup) paths that can satisfy the (network and trust) objectives associated with each path profile.
Traffic Engineering Policy Engine	Maps the set of recommended paths for each path profile into enforceable: i) Flex Algo Definition (FAD) to be advertised across the network segment, and ii) SR policy to be instantiated into the ingress node of the requested workload.
Secure Oracle	Part of the CASTOR DLT; checks the veracity of the data before being stored on the DLT.
Security Context Broker	Part of the CASTOR DLT; provides interfaces for authorized entities to interact with (i.e., process/consume) the on-chain data.
Trust Exposure Layer	Part of the CASTOR DLT; exposes interfaces for external stakeholders to access trust capabilities that are provided by a domain, while maintaining the necessary level of abstraction.
Infrastructure Layer (Routing plane)	
Trust Network Device Extensions (TNDE)	All the CASTOR components that are deployed in a network device (i.e., router element) to enable trusted TNDI onboarding, runtime monitoring, and trust assessment. The TNDE forms the main device-side TCB of CASTOR (together with the Trace Units operated by the TNDE's Tracing Hub).
Trust Network Device Interface (TNDI)	(Managed by the TN-DSM) Unit of a network device that joins a CASTOR network domain for trusted path routing (e.g., a vRouter). The TNDE securely collects runtime evidence and performs local trust assessments for each TNDI.
Trust Network Device Security Monitor (TN-DSM)	(Part of the TNDE) Enables a CASTOR orchestrator to attest the correct state of the TNDE and securely onboard one or multiple TNDIs of a network device to the CASTOR network domain. The TN-DSM implements the TNDI-SP to expose control and data channels towards CASTOR's upper layer components for configuration and trace/evidence/ATL sharing. The TN-DSM manages and reports the secure collection of trustworthiness evidence and local ATLs for each TNDI via the Tracing Hub, supported Trust Sources (i.e., Attestation and FSM sources), and local TAF.
Trust Policy Language (TPL) Data Connector	(Part of the TNDE) Retrieves the Trust Policy (i.e., Trust Model, RTL Logic, Target Trust propositions) of the operator for a TNDI and enables it in the Local TAF agent. The TPL Data Connector is configured by the orchestrator via the TN-DSM using the TNDI-SP control channel.
Tracing Hub	(Part of the TNDE) The Tracing Hub is configured by the TN-DSM based on the trust policy to monitor runtime configuration and behavior traces of a TNDI for the evidence generation by the attestation and FSM sources. The Tracing Hub can operate multiple different Trace Units to collect TNDI traces. The Tracing Layer constitutes an overarching term that incorporates the tracing capabilities in CASTOR. This includes the Tracing Hub as well as any Trace Unit that may be connected to it in order to provide meaningful trace information to the CASTOR framework.
Trace Unit	The Trace Units are the TNDE-external tracing mechanisms operated by the Tracing Hub to collect configurational and/or behavioural traces from the TNDIs. CASTOR considers the exploration of different types of Trace Units, some located outside the TNDIs, some inside the TNDIs. The Trace Units are part of CASTOR's device-side TCB in the sense that the trustworthiness of the raw traces of that particular unit depends on the unit's security.
Attestation Source	(Part of the TNDE) Serves attestation requests and provides fresh attestation claims based on the traces collected of a TNDI (e.g., about a router's OS).
FSM Source	(Part of the TNDE) Receives runtime behavior traces of a TNDI via the Tracing Hub. The agent leverages pre-learned FSMs to detect anomalies from a TNDI's "normal" (benign) behavior and generates respective evidence for the TAFs.
Local TAF Agent	(Part of the TNDE) Performs the dynamic trust assessment within the network device based on the evidence received from the Attestation and FSM sources. It reports the ATLs to the relying parties (Global TAF, Orchestrator, DLT, and neighbouring routers) via the TN-DSM, e.g., through TNDI-SP data channels.

Table 6.1: CASTOR Artifacts - Naming convention

6.2.1 SLA Translation & Decomposition

The Service Intent Translation & Decomposition Service (SITDS) is responsible for taking the high-level "service ask" from the Application Service Provider formalized and encoding the required trust and network properties straight from the Service Level Agreement (SLA) and Secure SLA (SSLA) into Default and Fallback Intents. This is the crucial semantic step: it binds business commitments (availability, integrity, confidentiality, latency, etc.) to technical requirements that the rest of the system can reason about and later enforce.

Before initiating the process for this component, negotiations with the Service Provider must take place. During these discussions, the required SLAs and SSLAs are defined, agreed upon, and subsequently submitted to SITDS. Then, SITDS consults the Path Profiles Catalogue and current Topology & Path Profiles. As soon as the Facility Layer responds, SITDS maps and translates each SLA to Intents that are capable of satisfying the desired trust and performance specifications, and then forwards them for further processing to the Facility Layer.

The encoded intents are forwarded to the Path Profile Catalogue Engine, which is responsible for gathering and matching them against the available path profiles. Once the mapping is completed, the resulting profiles return to the Facility Layer.

At this stage, the Facility Layer publishes the current global and local trust policies for each path profile to the blockchain, managed by the CASTOR DTL component.

To enforce the generated path profiles on network devices, the Facility Layer supplies the Optimization Engine with the target network scope, trust profiles, and topology. Once candidate paths are computed, they are forwarded to the Traffic Engineering Policy Engine, which compiles them into device-ready, enforceable configurations. After the Facility Layer receives these configurations, it delivers them to the Path Computation Element (PCE) for activation and network-wide deployment.

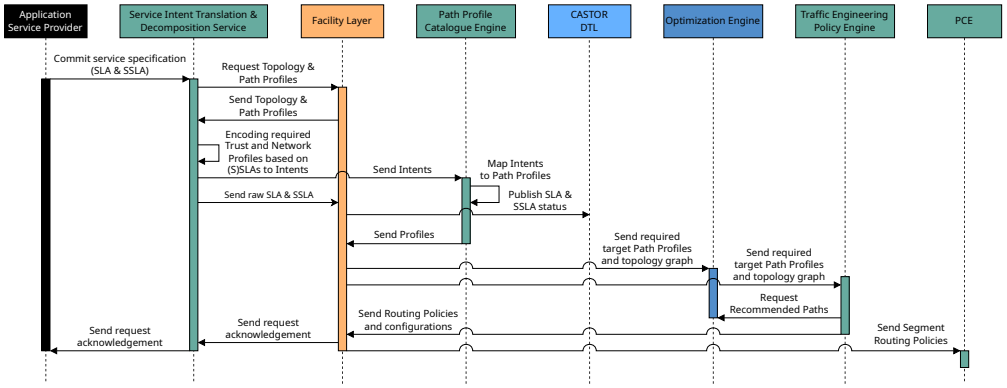

Figure 6.3: Translation of intents to enforceable policies

In CASTOR, inter-domain service provisioning is pretended to be transparent to the service provider/requester, delivering seamless end-to-end (E2E) services across multiple administrative domains. This model assumes pre-existing peering agreements that establish a trusted basis for cooperation. When a service request arrives, coordinated control-plane interaction among the participating domains is required—either via a higher-level E2E Orchestration entity or through direct PCE-to-PCE communication—to negotiate and instantiate the inter-domain connectivity. Such coordination may involve exchanging BGP routes and deploying inter-domain TE policies, ensuring consistent policy enforcement and performance guarantees across domain boundaries. Detailed implementation choices and operational procedures for these cross-domain mechanisms will be specified in Deliverable D5.1.

6.2.1.1 Intent and SLA Standards

In order to standardize the terms used in this document, this section will define the concepts used:

- **Intent**: Intent is the formal specification of all expectations including requirements, goals, and constraints given to a technical system. (TMForum - Intent in Autonomous Networks v1.3.0 (IG1253))
- **Service Level Agreement (SLA)**: A SLA is an element of a formal, negotiated commercial contract between two Organizations, i.e. one with a Service Provider (SP) Role and one a Customer Role. It documents the common understanding of all aspects of the Product and the roles and responsibilities of both Organizations from product ordering to termination. SLAs can include many aspects of a Product, such as performance objectives, customer care procedures, billing arrangements, service provisioning requirements, etc. (TMForum - GB917 SLA Management Handbook R3.1)
- **Secure Service Level Agreement (SSLA)**: The SSLA (Secure Service Level Agreement) functions as the first intent within Intent Based Networking Management (IBNM) systems, serving to define Security and Service Level Objectives (SSLOs). This initial intent is structured to conform to the requirements provided by stakeholders and end users, particularly regarding trust. The scope of the SSLA's objectives is essential security requirements, including confidentiality, privacy, authentication, and protection against specific attacks. Establishing the SSLA is the initial step in a workflow that ultimately moves to enforcement, leading to the creation of specific policies, rules, and profiles for security-related properties and service provisioning.
- **SLA Negotiation**: SLA negotiation refers to the dynamic process through which Service Provider and CASTOR establish and agree on quality of service, security, and reliability parameters that govern their interactions, ensuring that data exchange and resource access meet predefined expectations without undermining trust. Rather than being static, this negotiation often involves adaptive mechanisms that consider context, capabilities, and risk tolerance, allowing systems to align performance guarantees — such as latency, throughput, and availability — with security assurances, including confidentiality, integrity, and compliance. This negotiation is beyond the scope of CASTOR, so it is assumed that the negotiation will occur before our service is reached.

6.2.2 CASTOR Orchestration

Figure 6.4 presents the CASTOR orchestration message sequence chart when a service provider initiates an intent-based service order. The diagram outlines the steps that follow the triggering of the CASTOR Facility Layer. Network- and trust-related telemetry data are continuously collected and sent to the Facility Layer, originating from the local TAF and the telemetry API, which receives data from the edge layer. This enables the Facility Layer to construct a fresh Topology Graph with all the network and trust-related attributes that characterize the infrastructure layer. This information is crucial both when negotiating a new SSLA (see Section 6.2.1), but also when the Traffic Engineering Policy Engine needs to provide new TE policies based on fresh recommendations from the Optimization Engine. Based on the available Topology Graph and the selected path profile requirements, the Traffic Engineering Policy Engine decides whether to request fresh recommendations from the Optimization Engine. If so, and upon computation of the recommended paths, the Optimization Engine sends them back to the Traffic Engineering Policy Engine, which generates the corresponding routing policies. Eventually, this allows the Traffic Engineering Policy Engine to trigger - via the Facility Layer - the Service Orchestrator to configure the underlying infrastructure resources (e.g., deploy the necessary pods, including source and destination pods) and/or also activate the PCE to enforce the required routing paths.

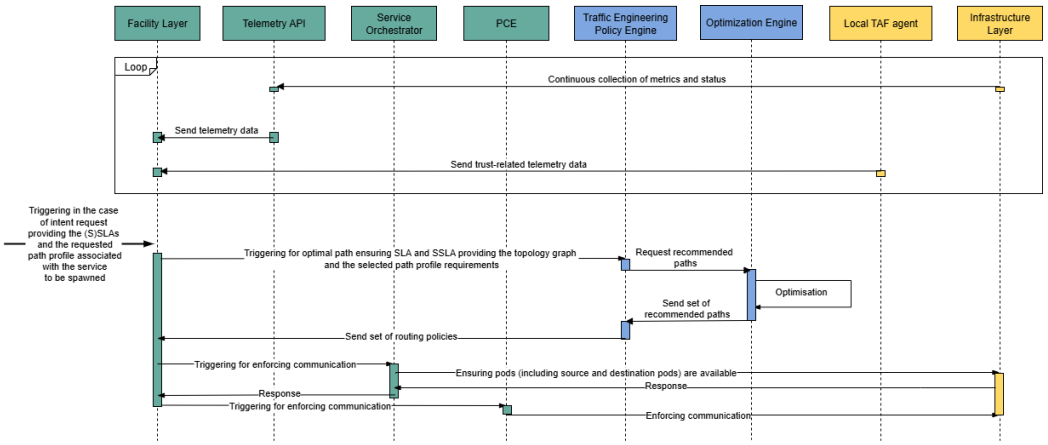

Figure 6.4: Orchestration flow of actions when there is an intent-based service order

Figure 6.5 illustrates the orchestration flow when a new node joins the network. In this scenario, the CASTOR orchestrator initiates the triggering of the Facility Layer. The Facility Layer updates the topology graph and sends the updated graph along with relevant telemetry data to the traffic engineering police engine, which requests from the optimization engine to recalculate the recommended paths. These are passed back to the traffic engineering policy engine, which returns the new routing policies to the Facility Layer. As before, the Facility Layer initiates the service orchestrator to deploy the necessary pods and activates the PCE to manage the communication setup.

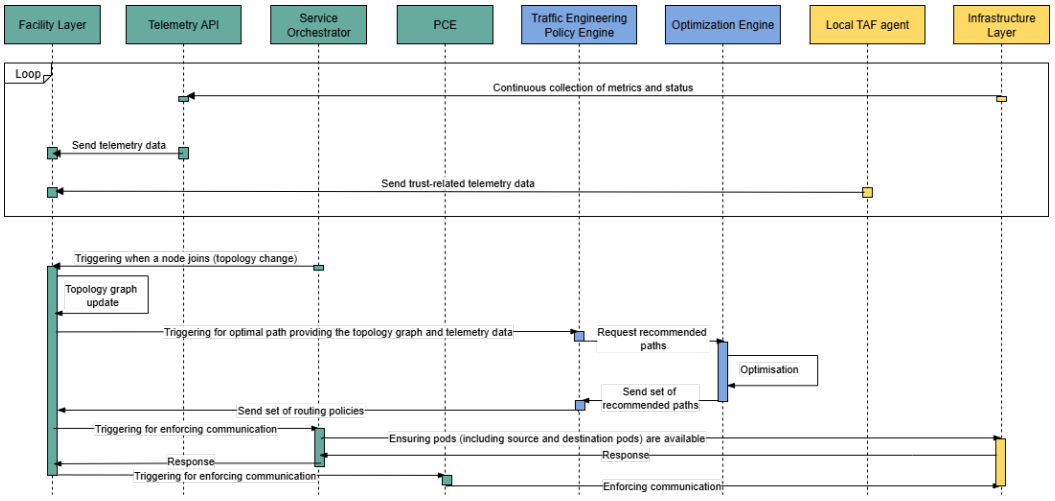

Figure 6.5: Orchestration flow of actions when a node joins the network

Figure 6.6 depicts the orchestration flow in response to topology changes, such as a node drop or departure. These events are detected by the telemetry API, which triggers the Facility Layer to update the topology graph. The subsequent steps, triggering the traffic engineering policy engine, optimization engine, Facility Layer, service orchestrator and PCE, follow the same process as described above.

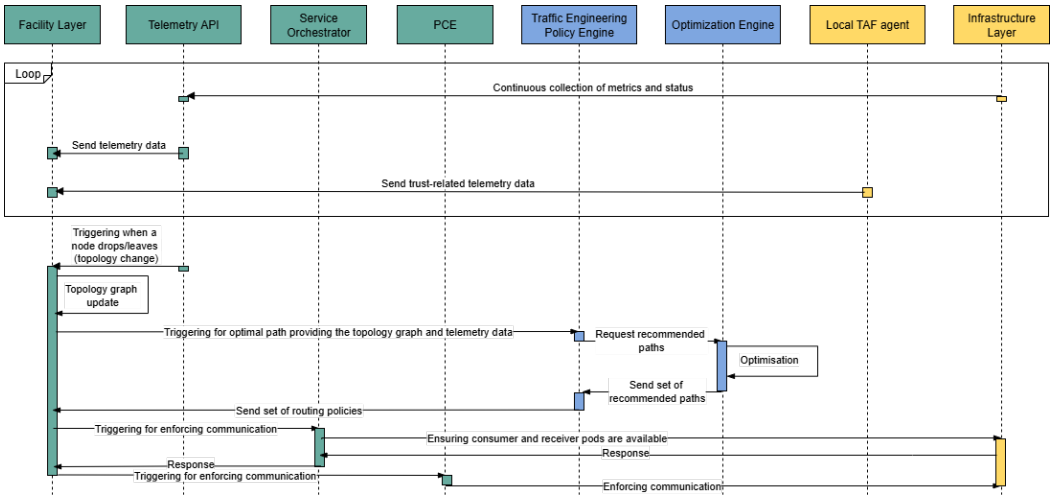

Figure 6.6: Orchestration flow of actions when a node leaves/drops the network

As an example of its main actions, the service orchestrator relies on a Kubernetes-native deployment model to realize intent-based services. It manages the instantiation, configuration, and lifecycle of services represented as containerized pods, which are stored in an image registry and instantiated on-demand across a cluster of worker nodes. The orchestration pipeline ensures the correct placement and interconnection of service pods (e.g., endpoints in a slice), infrastructure pods (such as vRouters, load balancers, and telemetry collectors), and control-plane components (such as the optimization engine or PCE), all of which may run as microservices within the cluster. The Kubernetes control plane handles resource scheduling, pod health, node status, and network policies through CNIs. Virtual links between vrouters (Kubernetes pods) is created by assigning each pod its own unique IP address, forming a shared network that enables IP-based communication between pods managed by CNI plugins.

The orchestration flows described in Figure 6.4 to Figure 6.6 can also be understood through the four discrete phases identified in Section 6.1.

In the preparedness phase, the operator defines the service catalogue (e.g., the path profile catalogue) within a Kubernetes-managed environment. While the service itself is represented by a set of pods stored in an image registry, the services offered by CASTOR are also implemented as pods, managed through Kubernetes. The path profile catalogue, defined and maintained by CASTOR, will be stored in a dedicated database. At this stage, service intents must also be translated into domain-specific policies, enabling their fulfilment through the orchestration framework.

In the service registration phase, once the orchestration environment is configured, service requests can be registered through an interface exposed to the user. Service deployment involves instantiating the required pods on the appropriate worker nodes and configuring the corresponding vRouters. The same interface used to register a request also provides feedback to the client, notifying them when the requested service has been successfully deployed.

The proactive phase addresses the modification of network resources, including both the addition and removal of nodes/resources. For example, the orchestrator may deploy new vRouter pods and configure them as required. Alternatively, a new physical node may initiate its own onboarding process, reaching out to the orchestrator once it joins the network. Secure and authenticated communication may involve the exchange of keys managed as part of the pod's security context configuration. SLA compliance is continuously monitored through telemetry collection mechanisms (e.g., Prometheus), which extract performance metrics from pods and potentially report link-state information (via BGP-LS, IS-IS LSPs, or other methods). Alerts are generated when SLA violations are detected, enabling proactive responses before service quality degrades.

In the reactive phase, when SLA assurance is compromised, CASTOR's self-healing capabilities come into play. Events such as node failures or link degradation trigger telemetry updates that propagate

through the orchestration chain (Facility Layer, optimization engine, routing construction engine, orchestrator, PCE). The orchestrator may respond with corrective actions, such as migrating pods, spawning replacements, updating container images, or applying patches. Alerts raised by monitoring systems (e.g., Prometheus AlertManager) can be delivered via messaging platforms like Kafka, where CASTOR services listen for events and execute mitigation strategies.

While the current orchestration framework demonstrates dynamic, intent-based control of 5G services using Kubernetes, several open challenges must be investigated. One such challenge concerns the co-existence of router services and security controls on Kubernetes worker nodes. Specifically, it remains unclear how the virtualization of vRouters affect the trust characterization that CASTOR aims to support. A key question is whether secure isolation can be maintained when router functions and sensitive workloads are co-located on the same physical infrastructure. Another important challenge involves the latency between physical routers and virtualized network interfaces. The integration of physical and virtual network paths may introduce delays, particularly through mechanisms such as container-based interfaces or SRv6 encapsulation, potentially impacting end-to-end latency and SLA compliance. These questions will be addressed in subsequent CASTOR deliverables through targeted evaluations, prototype deployments, and performance assessments.

6.2.2.1 CASTOR Path Computation Element

A Path Computation Element (PCE) is a network component, defined by the IETF, that is responsible for determining efficient end-to-end paths across a network, particularly in complex, multi-domain, or multi-layer environments. It uses advanced algorithms and knowledge of the network topology, constraints (such as bandwidth, latency, or policy), and traffic engineering requirements to compute optimal routes that may not be easily determined by distributed routing protocols alone. The PCE can operate centrally, and network devices (like routers) interact with it through the Path Computation Element Communication Protocol (PCEP), enabling dynamic and flexible path setup—critical for applications such as MPLS-TE, GMPLS.

Within this context, a PCE contributes to end-to-end service realization by computing paths for the intra-domain segment and steering configuration at ingress and border routers in line with local traffic-engineering constraints and SLA/trust objectives. Continuity beyond the domain boundary relies on pre-established peering and inter-domain TE policies, while coordination—via higher-level orchestration or PCE-to-PCE communication—stitches the per-domain segments into a consistent E2E service and preserves policy alignment across domains.

A novel extension currently under research enhances the traditional PCE architecture by introducing an intermediate functional block, the Enforcement Extension, between the PCE and the Path Computation Clients (PCCs) on the Infrastructure Layer, aimed at bridging the gap between path computation and operational enforcement. Unlike the default PCE–PCC model, where the PCE merely provides path recommendations or delegates path setup, this new extension assumes responsibility for enforcing configuration directly on the PCCs, ensuring that computed paths are not only selected but also consistently applied across the network.

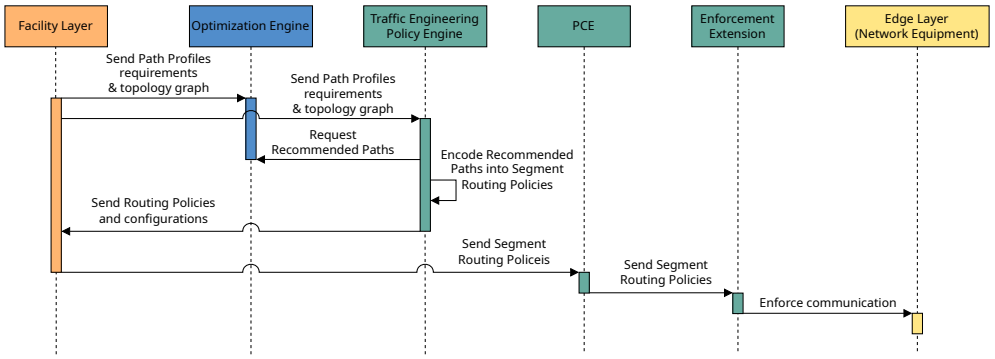

Figure 6.7: PCE Extension

6.2.3 Distributed Ledger Technologies

CASTOR adopts and extends the usage of the Distributed Ledger Technology (DLT) as **a dedicated trust enablement layer that supports the management, monitoring and storage of all data related to the trust lifecycle of compute continuum elements.** In contrast to generic blockchain-based infrastructures, in CASTOR the DLT is utilised specifically **to ensure that trust-related semantics are consistently maintained across administrative domains, supporting secure decision-making during path establishment and traffic engineering processes.**

The integration of DLT in CASTOR facilitates the end-to-end lifecycle management of trust-related data, ensuring that the data stored, exchanged and processed complies with varying confidentiality requirements and access control restrictions. Such trust-related data include attestation claims and raw traces collected from the infrastructure layer, SSLA paramters that characterize the requirements that dictate the service fulfillment and assurance, as well as the Trust Policies that encapsulate the Required Trust Levels (RTLs) and Trust Models that need to be enforced by the different CASTOR TAF actors. Since different types of trust data (such as attestation claims, trust models, SLA/SSLA thresholds) expose different levels of sensitivity, CASTOR enforces granularity in access control policies. This enables CASTOR to maintain the required balance between auditability of all trust data transactions and operational efficiency when leveraging this information during runtime, without unnecessarily exposing sensitive or infrastructure-specific details.

From a trust assurance perspective, CASTOR leverages the DLT to enable complete auditability of trust data processing and verifiability of trust evidence ingestion. Upon their collection from the network edge via the CASTOR TNDEs, attestation claims are transmitted via secure communication channels and validated on-chain by means of secure oracles elevating Trusted Execution Environments (TEEs), such as the Phala network or Phala-like confidential blockchain network envisioned in CASTOR.

Trust policies stored within the CASTOR DLT incorporates the envisioned trust models which specify behavioural expectations and compliance requirements for network operation and characterising path profiles (e.g., high availability, high integrity), together with the associated RTLs, and the relevant trustworthiness evidence needed for deriving the Actual Trust Level during runtime. Similarly, SSLAs are anchored on-chain during service onboarding to establish the agreed trust and performance guarantees between the Service Provider and the network operator. These are subsequently used as reference data for automated detection and notification of any violation during runtime.

The specific mechanisms governing identity management and access control in relation to DLT-interfacing components – including the three CASTOR access control categories and the enforcement of zero-trust segmentation at system and per-property level – are described below.

In the following subsections, the main high-level workflows of the CASTOR DLT are presented, demonstrating how different types of trust-related data are handled across the compute lifecycle. These work-

flows focus on: (1) the management and monitoring of SLAs/SSLAs and the notification of their property violations, (2) the definition and use of trust policies alongside runtime transmission of trustworthiness claims, and (3) the controlled exposure of abstracted trust capabilities to authorized external entities.

6.2.3.1 CASTOR Multi-layered access control framework

In CASTOR, access to trust-related data managed via the Distributed Ledger Technology (DLT) infrastructure must adhere to different policies, depending on the sensitivity, origin and intended use of this information. Since these data include trustworthiness claims, trust policies, runtime attestation submissions and SLA/SSLA agreements, their management must respect auditability requirements while ensuring operational efficiency for time-critical processes such as trust assessment and traffic engineering.

To enable selective access over highly granular trust-related content, CASTOR enforces three distinct access control models, each mapped to a specific category of system actors. Following a top to bottom approach, and considering also the overall CC-wide CASTOR architecture in Figure 6.2, we distinguish the following access control layers:

- **Layer 1: Layer 1: Attribute-based Access Control per interaction** External actors such as service providers, certification authorities or cross-domain orchestrators are granted access on a per-request basis and must present proof of attribute ownership using Verifiable Credentials (VCs) aligned with CASTOR-defined roles and policies. Requests are received either via CASTOR APIs (for internal services) or through blockchain-based querying mechanisms for inter-domain interactions. When access is requested, policy evaluation is conducted by the Trust Exposure Layer based on pre-defined rules and access policies. If approved, the appropriate abstraction function gets invoked so that the necessary obfuscation mechanisms are applied and the appropriate trust-related insights are returned (e.g. minimum trust level, SLA/SSLA compliance flag), preserving confidentiality.
- **Layer 2: Whitelisted domains**. As already mentioned in the introduction of this section, the CASTOR-enabled managed domains are able to interact with the CASTOR DLT for auditability and trust-related data exchange with the necessary level of abstraction. In this context, the main control services - such as the Traffic Engineering Policy Engine - that are running with the same locality as the Service Orchestrator are considered trusted components. In addition, even the rest of the CASTOR components at the orchestration layer that can exhibit acceptable level of isolation guarantees do not require updatable access policies. Hence, the concept of whitelisting in this layer to encompass the static policies that are associated with the overarching CASTOR framework at the orchestration layer. As shown in the following sequence diagrams, these components interact with DLT-stored data using on-chain whitelisted blockchain addresses, eliminating the need for continuous authentication or presentation of Verifiable Credentials (VCs). This approach supports low-latency access to time-sensitive data (e.g. SLA violation events) required for seamless multi-path control and overall lifecycle management of the underlying infrastructure layer. If required, additional hardware-level attestation may be enforced at system bootstrap, but runtime validation is not re-enforced.
- **Layer 3: TNDI-SP Validation**. CASTOR deployed elements contributing runtime trust evidence, such as the Local TAF and other operational CASTOR components of the TNDE, are not inherently trusted. They must authenticate when establishing communication session(s) with CASTOR infrastructure. CASTOR uses session-based authentication, where identity validation occurs once per session — typically during secure onboarding or session renewal. Authentication leverages CASTOR Verifiable Credentials where applicable, and secure channel establishment uses Diffie–Hellman-like authenticated key exchange mechanisms directly between the TNDE enclave

and the Secure Oracle running within a Trusted Execution Environment (TEE). This reduces handshake frequency and minimizes protocol overhead. In case of session updates, cryptographic key rotation is used instead of full re-authentication. The TN-DSM supports key generation, management, and refresh procedures. Details on the overarching concepts around the internal TNDE architecture and the requirements that guide the designs on the confidential TNDI-SP data channels is provided in Deliverable D3.1.

Overall, CASTOR enforces Zero Trust segmentation at workflow, component and trust property level. Each service is mapped to a path profile, which in turn dictates specific access control and authentication requirements applicable to different trust-related data segments. As a result, information derived from multiple tenants or domains is processed within isolated Secure Oracle applications to prevent cross-tenant leakage even if a key or component is compromised.

6.2.3.2 Storage of SLAs and SSLAs

In the Orchestration Layer of CASTOR system during the preparedness phase (step f in Figure 6.2), the SLAs and SSLAs of a router that is prepared to onboard in the network - as defined by the Application Service Provider in the form of service intents and translated accordingly in secure requirements - have to be stored in the CASTOR DLT (Blockchain Infrastructure). As shown in Figure 6.8, the Facility Layer makes the related request to the Security Context Broker (SCB) of the CASTOR DLT, which acts as the sole trusted intermediary between the external CASTOR entities and the DLT. The SCB, leveraging its Attribute-Based Access Control (ABAC) capabilities, checks the attributes that are originated from the Verifiable Credential of the domain operator and along with the deployed smart contract logic they evaluate whether the invoker's address (e.g., Externally Owned Account address) belongs to the registered whitelist. Upon authorization, the transaction gets executed on-chain and the SSLA is stored in the Private Ledger of the CASTOR DLT and can be further updated during the operational lifecycle of the deployed services. Further details on the smart contract breakdown and the interfacing between the relevant components is presented in Deliverable D5.1.

As shown in Figure 6.9, the Facility Layer submits the SLA/SSLA storage request to the Security Context Broker (SCB), which operates as the trusted intermediary between CASTOR components and the CASTOR Ledger. In this context, the Facility Layer is able to use the Layer 2 authorization flow presented above in order to record SSLA data on-chain.

Once agreed, in high level, SLA/SSLA data is forwarded to the Secure Oracle or fetched by it, which validates the integrity and correctness of the data before it is anchored on-chain in the CASTOR Private Ledger. Storing these parameters on-chain enables both auditability (ensuring that historical agreements can be validated transparently) and support for SLA/SSLA violation detection by authorized and registered stakeholders (through the Trust Exposure Layer as presented in the following subsection).

Beyond initial storage, SLA/SSLA parameters are continuously monitored by the Orchestration Layer. If a violation occurs (e.g., exceeding latency bounds or falling below a required trust threshold), the orchestrator issues a violation event. This event is submitted to the SCB, which maps it to the corresponding SLA/SSLA record stored on-chain, creating a traceable audit record and triggering remedial action (e.g., policy adjustment, path reconfiguration or agreement renegotiation).

Furthermore, storing the SLA/SSLA on-chain provides the capability for authorized external stakeholders to register for violation notifications. These notifications may be delivered via CASTOR APIs (for internal entities) or through the Trust Exposure Layer, depending on integration requirements. Access to such notifications is governed by CASTOR multi-layred approach presented at the beginning of this Section.

This process establishes the basis for real-time compliance monitoring and reactive trust-based network management within CASTOR, ensuring that the guarantees agreed through the SLA/SSLA are not only enforceable but also continuously verifiable.

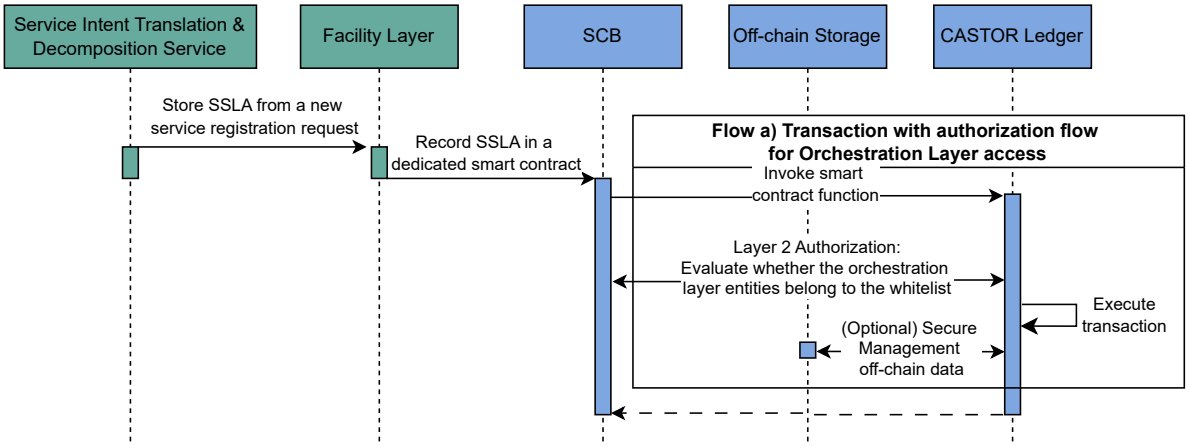

Figure 6.8: CASTOR DLT flow of actions - Storage of SLAs and SSLAs

6.2.3.3 Storage of Trust Policies

In CASTOR, Trust Policies define the operational trust conditions of compute continuum network elements. They specify the guarantees, behavioural expectations and compliance criteria that must be evaluated during runtime. These policies indicate which raw traces must be collected and what trust propositions must be assessed by the trust framework. Trust Policies form part of the CASTOR trust model and include the Required Trust Levels (RTLs) associated with predefined path profiles (e.g., high integrity, high availability), serving as reference criteria to determine whether nodes and links are eligible to participate in a specific routing path.

As shown in Figure 6.9, during the proactive phase, the CASTOR Facility Layer submits the initial Trust Policies to the DLT via the Security Context Broker (SCB), following ABAC-based authorization. This process is a result of the risk analysis that is presented in Section 4.4 and culminates in the realization of the RTL and the overall Trust Policy. Once the submission request is recorded on-chain, the Secure Oracle detects the event and either retrieves the referenced policy data or receives it via an event listener mechanism. The Secure Oracle performs authenticity and integrity validation inside a Trusted Execution Environment (TEE) before the policy is definitively anchored on the ledger. The Secure Oracle operates by continuously monitoring on-chain transactions and responding to relevant events. Once validation is successful, subscribed TAF instances are notified via the SCB.

During runtime, trust-related data is collected from the routers using the Trust Network Device Extensions (TNDEs), via dedicated TNDI-SP data channels. CASTOR introduces a key innovation by establishing an authenticated and encrypted enclave-to-enclave communication channel between the TNDE within the routing device and the Secure Oracle running as an isolated application inside a TEE (see Section 6.2.8.3). This is achieved using a Diffie–Hellman-like authenticated key establishment process performed only once at the beginning of each session. To minimise communication overhead, session updates trigger only key rotation rather than full re-authentication. Key lifecycle management, including key generation and update, is handled by the TNDE Security Manager (TNDE-SM). CASTOR identity management using Verifiable Credentials (VCs) supports authentication where required, although inherently trusted components (e.g. orchestrator) are exempt from per-session VC presentation. Details of this are discussed in the access control section. Runtime trust evidence may be processed using one of the following approaches:

- TNDE $\rightarrow$ Local TAF $\rightarrow$ SCB $\rightarrow$ Secure Oracle $\rightarrow$ DLT (submission event recorded). The Secure Oracle detects this on-chain event, fetches or receives the associated evidence, validates it inside the TEE and, if successful, authorises final anchoring. This approach assumes trust in the Global TAF for pre-processing.

- TNDE → SCB → Secure Oracle → DLT (submission event recorded without TAF pre-processing). The Secure Oracle monitors the event, retrieves or receives the evidence via the event listener, validates it inside the TEE and approves anchoring. The Global TAF may later retrieve the evidence directly from the ledger if required.

Figure 6.9 presents the former approach, focusing on how the Local TAF agent is able to carry out its trust engineering process. Nevertheless, the recording of data coming from other TNDE artifacts is almost identical, as the provisioning of the TNDI-SP channels is handled by the TN-DSM element depicted in the sequence diagram.

The selection of the preferred processing approach will be finalised in future deliverables, based on trust assumptions and performance requirements. This mechanism allows CASTOR to securely and efficiently ingest runtime trust evidence from edge routing elements and validate it before anchoring on the DLT, ensuring confidentiality, integrity and low operational overhead while supporting continuous runtime trust assessment.

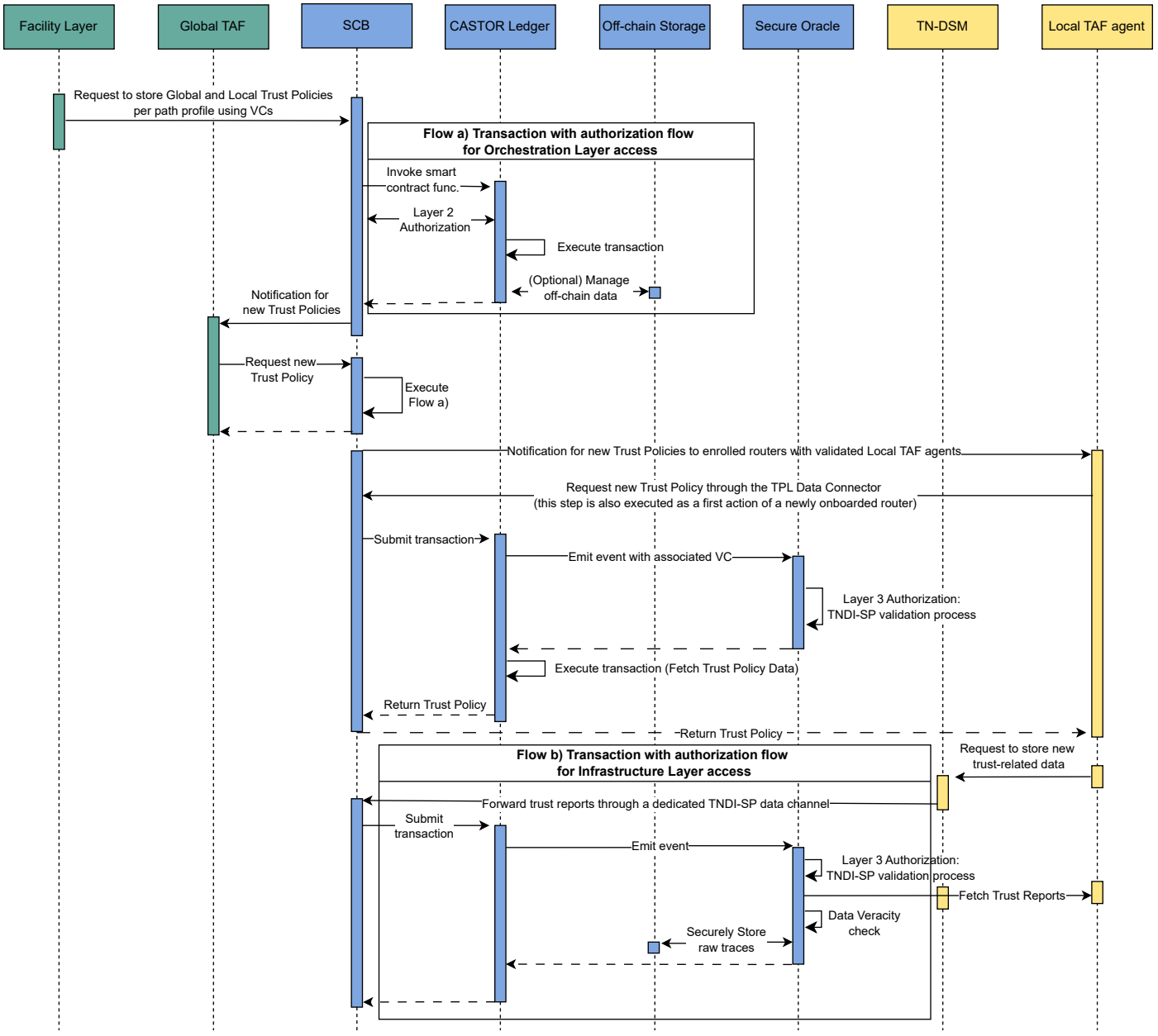

Figure 6.9: CASTOR DLT flow of actions - Recording of Trust Policies

6.2.3.4 Retrieval of abstracted trust capabilities

CASTOR introduces the capability to provide external stakeholders with controlled access to trust-related information, enabling them to assess the trustworthiness of network infrastructure without exposing sensitive internal data. Unlike static abstraction mechanisms, CASTOR does not store abstracted representations of trust data directly on-chain. Instead, abstraction is applied dynamically at query time, allowing the Trust Exposure Layer to expose only the minimum required information based on the requesting entity's attributes and authorization level.

Such trust insights may include aggregated metrics (e.g. overall infrastructure trust level), compliance evaluation results or notification indicators linked to SLA/SSLA violations. These capabilities are made available to authorized stakeholders such as service providers, cross-domain orchestrators or certification authorities.

As shown in Figure 6.10, the Trust Exposure Layer builds upon the CASTOR Blockchain Infrastructure by extending the existing network exposure functionality with additional trust-related information. For internal CASTOR entities (e.g. service orchestration platforms), access to this information is facilitated via CASTOR APIs (e.g., Trust Awareness API).

Only entities presenting the appropriate attributes (whose evaluation model is defined in the access control section) may access this information. Upon an authorized query event, the Trust Exposure Layer retrieves the relevant data from the DLT, processes it internally to enforce abstraction and privacy constraints, and returns only the permitted trust-related insights. This approach ensures that internal network details are not disclosed while still enabling cross-domain collaboration and compliance verification.

This mechanism complements SLA/SSLA monitoring and runtime trust evidence flows by enabling external entities to retrieve policy-relevant trust conditions and react in cases where violations or non-conformant trust states have been detected. It ensures that trust information can be leveraged in wider traffic engineering and routing decision procedures without compromising confidentiality.

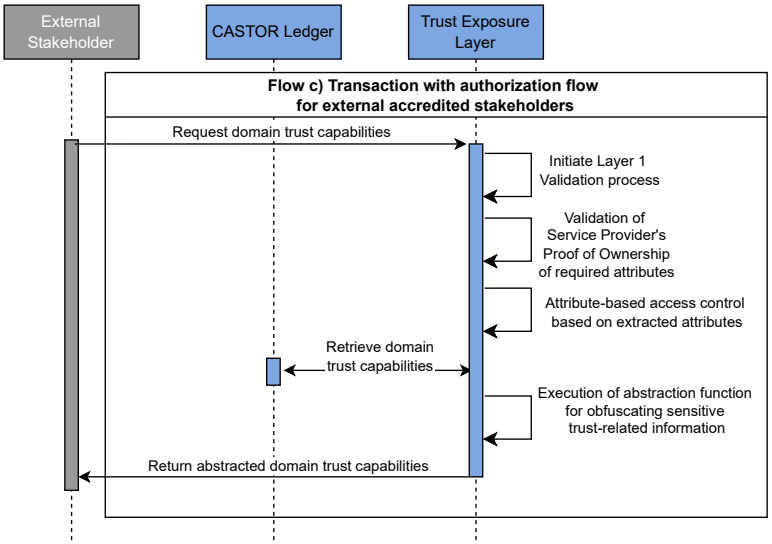

Figure 6.10: CASTOR DLT flow of actions - Abstraction of Trust Capabilities

6.2.4 CASTOR Risk Assessment Engine

6.2.4.1 Router type onboarding & Individual RTL Calculation

Phase 1, as shown in Figure 6.11, establishes the baseline trust requirements for router types before deployment. This phase focuses on individual router evaluation independent of network topology context,

providing the foundation for trustworthy network operations through evidence-based trust quantification.

The router onboarding process begins when the Network Operator's Security Administrator provides a list of specifications for each router type to be deployed to the CASTOR Risk Assessment Engine. As the designated security role within the Network Operator's domain, the Security Administrator is responsible for defining security policies and managing trust requirements across the operator's infrastructure. The above specification list includes detailed hardware characteristics, such as CPU architecture, memory capacity, and TPM capabilities, along with information on the software stack, including operating system versions and firmware details. The Security Administrator also defines the intended operational role for each type of router, whether it will function as an edge device, aggregation point, or core network element.

Following asset definition, the Security Administrator uploads cyber threat intelligence (CTI) data that combine information from multiple sources. This data incorporates automated feeds from public and private vulnerability databases, including CVE entries and CTI platforms, along with vendor-specific threat intelligence from router and hardware manufacturers who provide detailed security advisories, firmware vulnerability assessments, and hardware-specific attack vectors for their equipment. Data may also include manual input of organization-specific threats and operational security concerns. This hybrid approach ensures comprehensive threat coverage while enabling customization based on the organization's unique operational experience and risk profile. By integrating threat intelligence from router vendors alongside external sources, this comprehensive data collection allows the Network Operator to obtain an enhanced and holistic view of the risk posture across its entire infrastructure topology.

The CASTOR Risk Assessment Engine performs an internal analysis of the router specifications provided against the current threat landscape. This analysis involves five critical steps: First, threats are systematically analyzed per router type, considering both known vulnerabilities and potential attack surfaces based on the device's capabilities and deployment characteristics. Second, specific security controls are identified for each threat, mapping protective mechanisms such as secure boot, memory protection, and cryptographic validation to their corresponding attack vectors. Third, evidence types are mapped to each security control based on vendor capabilities and attestation infrastructure. These evidence mappings come pre-loaded in the Risk Assessment Engine from router vendors who specify which types of evidence their devices can provide to verify correct enforcement of security controls. For example, secure boot controls are mapped to boot measurement evidence from TPM PCR registers and signature verification status. Fourth, risk levels are calculated for each threat by quantifying their potential impact and likelihood of occurrence. Fifth, structured risk scenarios are generated by grouping threats according to their required security controls, maintaining clear traceability from threats to controls to measurable evidence types. It should be noted that this phase focuses on per-router-type risk assessment, analyzing each router type independently. The consideration of attack paths and cascading threats across interconnected routers is addressed in subsequent phases.

These structured risk scenarios, along with the calculated risk levels per threat and evidence type mappings, are transmitted to the Global TAF. The RTL calculation is performed collaboratively between the Risk Assessment Engine and the Global TAF. The Risk Assessment Engine provides the foundational risk analysis, including threat identification, risk levels per threat (combining impact and likelihood), security control mappings, and evidence type mappings pre-loaded from vendors. The Global TAF receives this data and calculates the onboarding phase RTL thresholds per router type configuration for each type of evidence. The calculation applies risk equations that consider the risk assessments from the Risk Assessment Engine, the effectiveness of security controls, and organizational policies (such as compliance requirements and risk tolerance). This collaborative approach ensures that the RTL values are both risk-appropriate and technically achievable based on what vendors can actually provide.

Critically, the RTL values are calculated per type of evidence, rather than per abstract security property, which is in agreement with the approach to the atomic trust proposition described in Chapter 4. The Trust Assessment Framework operates on atomic trust propositions, the most granular and measurable trust statements possible, which are directly derived from specific evidence types. For example, a boot-

time malware threat with high impact and medium likelihood may result in an RTL requirement of 0.85 for boot measurement evidence and 0.77 for signature verification evidence. This evidence-centric approach ensures that trust requirements remain tied to measurable, collectible data that can be verified through attestation during runtime operations. By expressing requirements at the atomic proposition level rather than abstract security properties, the system establishes clear, verifiable trust baselines for each router type. As the system progresses to Phase 2 with more complex path profile requirements, these atomic propositions can be composed into higher-level trust assessments, though the methodology for deriving RTL values for such composite propositions remains an area for future exploration.

Following RTL calculation, the Facility Layer encodes the Trust Policies by merging information from multiple sources: the RTL thresholds calculated by the Global TAF, the evidence types to be collected during runtime as specified by the Security Administrator and Risk Assessment Engine, and the Trust Models that guide TAF calculations. The Facility Layer constructs comprehensive Trust Policy documents for each router type, specifying RTL thresholds per evidence type, required attestation frequencies, and operational parameters. Trust Policies for the Global TAF are configured directly by the Facility Layer, while Trust Policies for Local TAF Agents are recorded on the CASTOR DLT infrastructure, creating an immutable and auditable record of onboarding requirements. The distributed ledger prevents unauthorized modifications and provides a tamper-proof audit trail for regulatory compliance and multi-party verification.

As part of the onboarding process described in Section 6.2.8, the Trust Policies are then distributed through the CASTOR DLT to configure TNDEs within the Infrastructure Layer and Network Operators, providing them with actionable specifications for deployment. These policies provide them with actionable specifications for deployment, including evidence types that must be collected from routers during runtime (such as boot measurements, signature verification status, or memory protection logs), quality thresholds that must be maintained for each evidence type (expressed as confidence levels), and collection frequencies for continuous attestation. This transforms abstract security requirements into concrete and measurable operational procedures that will be enforced when routers are deployed and begin operational trust assessment.

Although the sequence appears linear, the Risk Assessment process exhibits dynamic characteristics that enable continuous adaptation to changing conditions. Assessment can be triggered by various events beyond initial deployment, including firmware updates to existing router types, discovery of zero-day vulnerabilities affecting deployed systems, application of additional security controls, changes in organizational risk policies, or fundamental shifts in the threat landscape. This dynamic capability ensures that RTL calculations remain current and relevant throughout the entire router lifecycle, supporting the zero-trust principle of continuous verification rather than static trust assumptions.

Phase 1 establishes the essential foundation for reliable network operations by creating standardized, evidence-based trust baselines that are independent of specific considerations of the network topology. These baselines serve as the building blocks for more complex topology-aware analysis in subsequent phases, ensuring that individual device trust levels are clearly defined before analyzing how devices interact within network topologies.

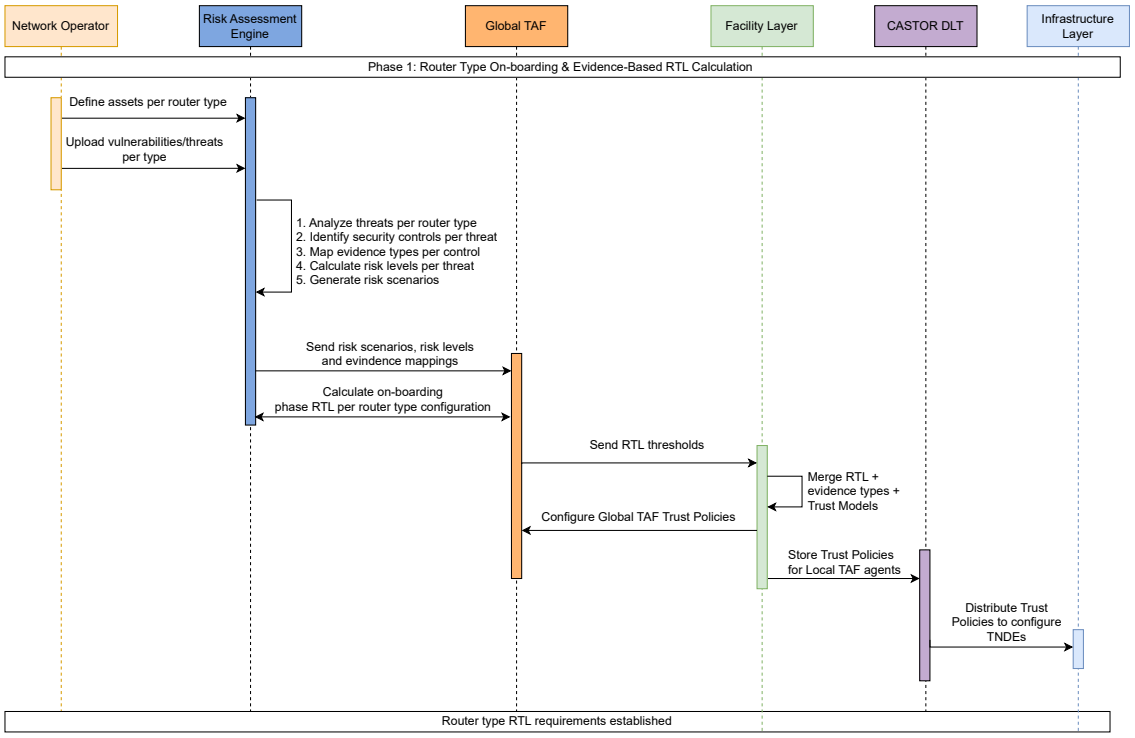

Figure 6.11: Phase 1: CASTOR router type onboarding & RTL Calculation

6.2.4.2 Topology Integration & Cascading Attack Analysis

Phase 2, as depicted in Figure 6.12, extends the individual router risk assessments from Phase 1 to a comprehensive topology-aware risk analysis that considers cascading attack scenarios and inter-router dependencies. This phase transforms static device-level risk calculations into dynamic, context-aware risk assessments that reflect the real-world operational environment where routers function as interconnected components of a larger network infrastructure.

Phase 2 assumes that service requirements have been translated into technical parameters and associated with appropriate path profiles from the Path Profile Catalogue, as described in the preparedness phase (Section 6.1.1). These path profiles define the trust and network constraints that must be met for different service levels.

The Network Operator provides intended topology information (via Security Admin), while the Facility Layer provides actual topology data that describe the operational network infrastructure where services will be deployed. This topology data encompasses the complete network hierarchy, including core routers in primary data centers, aggregation routers at distribution points, and edge routers at customer access locations. The description of the topology includes information about physical connectivity, bandwidth capacities, redundancy configurations, and geographic distribution of network elements.

The CASTOR Risk Assessment Engine performs sophisticated cascading attack analysis using the provided topology information to identify potential failure propagation paths and attack amplification scenarios (this step is omitted in the high-level description in Figure 6.2 for ease of readability). This analysis goes beyond individual device vulnerabilities to examine how compromise or failure of one network element could impact other connected devices and, ultimately, the services they support. The engine models various attack scenarios including BGP hijacking attacks that could redirect traffic through compromised nodes, physical infrastructure attacks that could isolate critical network segments, and coordinated attacks targeting multiple network elements simultaneously.

Through this cascading analysis, the Risk Assessment Engine generates updated risk assessments that

account for each router's position within the network topology and its potential impact on overall network security. Routers in critical positions, such as those providing backup connectivity to hospitals or serving as aggregation points for financial services, receive higher risk scores that reflect their elevated importance to network operations and the cascading effects of their potential compromise. In contrast, routers serving only residential traffic with multiple alternative paths may receive lower impact assessments due to their reduced criticality and limited cascading potential. Importantly, this topology-aware risk assessment may result in the derivation of different Trust Policies for routers with identical hardware specifications and configurations, solely based on their position within the network topology. Two routers of the same type may, therefore have different RTL requirements depending on whether they serve critical infrastructure or less sensitive traffic flows.

The Global TAF receives these topology-enhanced risk assessments, which include adjusted impact scores based on each router's network position, and recalculates Required Trust Level thresholds for each router position within the service path. The baseline RTL values established in Phase 1 are adjusted upward for routers with higher impact scores due to their criticality within the network topology. Routers in critical network positions where compromise could cause cascading failures may therefore require significantly higher RTL thresholds compared to their Phase 1 baselines, even though the router type and capabilities remain identical.

Following RTL recalculation by the Global TAF, the topology-adjusted RTL thresholds are transmitted to the Facility Layer for encoding into Trust Policies. It should be noted that the actual selection of network paths per service level involves additional considerations beyond trust requirements, including optimization problems that balance trust constraints with network performance objectives, which are addressed separately from the risk assessment process.

Following the transmission of topology-adjusted RTL thresholds to the Facility Layer, the flow from this point onwards follows the same process as Phase 1. The Facility Layer encodes the corresponding Trust Policies by merging information from multiple sources: RTL thresholds from Risk Assessment Engine and Global TAF, evidence types to be collected during runtime as specified by the Security Administrator and Risk Assessment Engine, and Trust Models that guide TAF calculations. Trust Policies for the Global TAF are configured directly by the Facility Layer, while Trust Policies for Local TAF Agents are recorded on the CASTOR DLT infrastructure. These Trust Policies specify the topology-adjusted RTL thresholds per evidence type, required attestation frequencies, and continuous compliance monitoring procedures that must be implemented on each router participating in the service delivery.

As described in the proactive phase (Section 6.1.3), when a new router is included in the topology, it accesses and enforces its corresponding Trust Policy from the CASTOR DLT. The topology-aware Trust Policies derived in Phase 2 ensure that each newly enrolled router receives RTL requirements that reflect not only its hardware capabilities (as in Phase 1) but also its specific position within the network topology and its potential impact on cascading attack scenarios. This integration between Phase 2's topology-aware risk assessment and the proactive phase's router onboarding ensures that trust requirements remain contextually appropriate throughout the network's operational lifecycle.

The topology analysis reveals how individual router compromise can have far-reaching consequences beyond the immediately affected device. For instance, compromise of a core router may impact hundreds of edge devices and thousands of customers, while failure of an edge router in a medical district could affect life-critical communication systems. This understanding of cascading impacts demonstrates why Phase 2's topology-aware approach is essential for realistic security planning, directly influencing the business and operational decisions that follow.

Like Phase 1, the topology-aware analysis exhibits dynamic characteristics that enable adaptation to changing network conditions. Topology modifications, service demand changes, and evolving threat landscapes can trigger reassessment of cascading attack scenarios and corresponding updates to service availability and RTL requirements. This continuous adaptation ensures that risk assessments remain accurate and relevant as the network infrastructure evolves to meet changing operational demands.

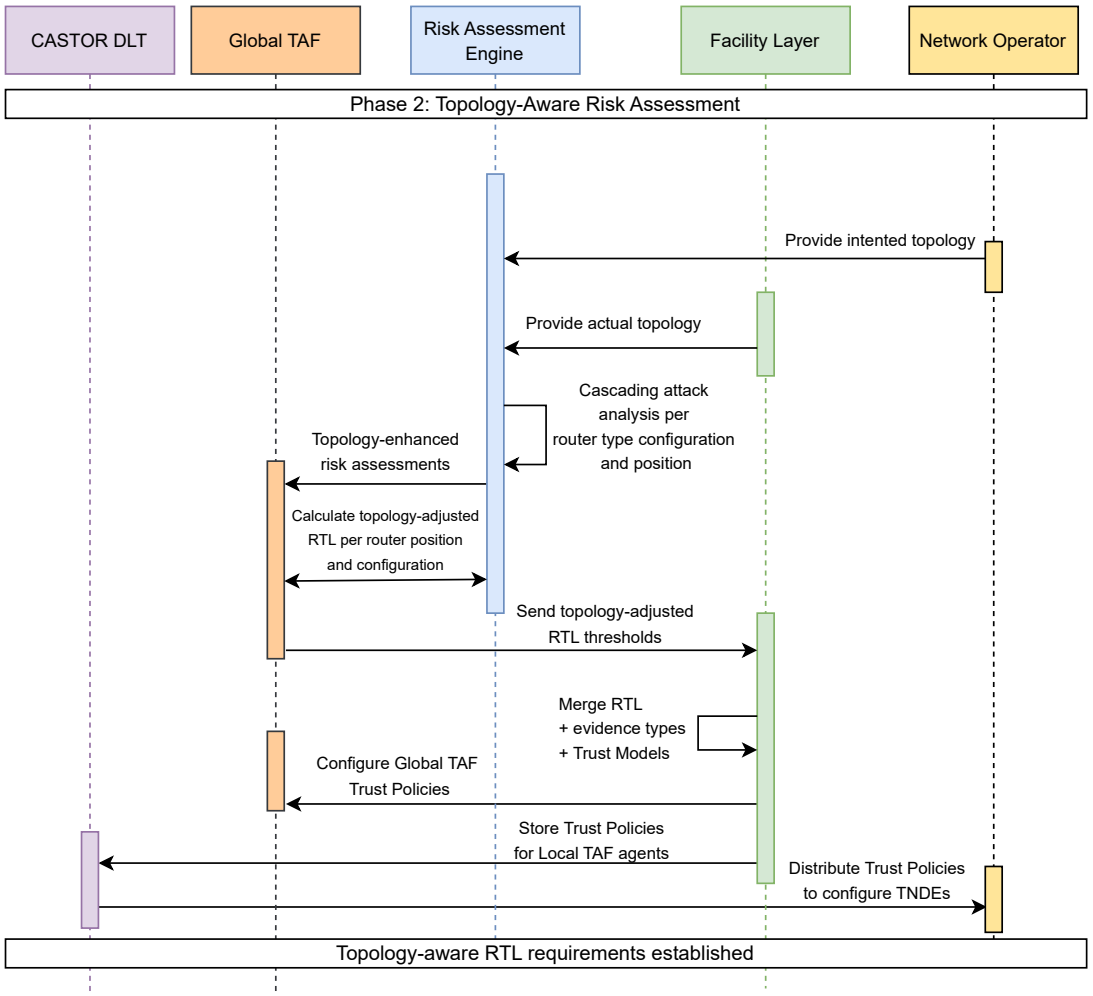

Figure 6.12: Phase 2: Topology Integration & Cascading Attack Analysis

6.2.5 Global and Local Trust Assessment

Overall, the Trust Assessment Framework (TAF) evaluates the trustworthiness across the network topology. It is located on the Orchestration layer (i.e., Global TAF) and on each edge network elements (i.g., Local TAF Agent). The overall TAF concepts span across the lifecycle of a router, instantiated either as a virtualized function on top of a commodity server (i.e., vRouter) or as a typical router hardware equipment. Based on the identified phases in the overall CASTOR framework (see 6.1), the TAF-related phases can be characterized as the following:

- **TAF Design phase**: As part of the preparedness phase, the network operator identifies the path profiles that will be offered by the underlying infrastructure. This phase includes the specification of all network- and trust-related characteristics that each path profile is offering. Trust characteristics can be expressed in the form of requirements that the underlying router elements must adhere to both during its onboarding phase but also throughout its operational phase.
- **Router onboarding phase**: This phase captures the process that a router must follow in order to enrol in the overall topology. This onboarding process involves the execution of an initial trust assessment task that can evaluate whether the router meets the minimum (integrity) guarantees posed by the network operator of this infrastructure. Part of the onboarding process is also the

establishment of the interactions with the neighbouring routers to ensure the continuous monitoring and assessing of trustworthiness across the topology.

- **Runtime phase**: This phase characterizes the trust assessment operations that need to be running when serving user workloads on top of the routers. With these policies, the control plane (i.e., via the CASTOR Facility Layer) is able to have an up-to-date view of the network and trust properties of all the routers in the underlying network segments.

6.2.5.1 TAF Design Phase

The definition of the trust-related guarantees of a path profile is expressed in the form of constraints defined over one or more trust propositions. These constraints - expressed in the form of Required Trust Levels (RTL) - may characterize the routers, the links and/or the overall path that will be selected once a path profile is selected to be deployed.

Depending on the intrinsic characteristics and capabilities of each router and its identified risk level, it is possible that there may be different RTL constraints that different types of routers may satisfy in order to serve a common trust guarantee for a trust property. This is intrinsically linked with the risk posture of each type of router to be employed in the underlying infrastructure as well as the (residual) risk that a network operator is willing to accept in order for the entire infrastructure to operate. Consequently, a network operator may require different number and type of evidence from routers with different characteristics.

That being said, part of the specification of the offered path profile catalogue involves the identification of all the trust-related information that will enable the overall Trust Assessment Framework to establish and maintain the trust characterization of the entire infrastructure layer.

The enforcement of a Trust Policy in a TAF agent allows the configuration of the continuous monitoring of the trust guarantees that the underlying infrastructure satisfies during the different phases of the CASTOR framework. There are different Trust Policies per type of router and per phase (e.g., onboarding of a router, runtime phase where a router servers one or more path profiles).

Once the Trust Policies are specified expressing both the minimum requirements for a router to enrol the topology and the guarantees that each path profile shall offer, it is possible to instantiate the Global TAF running in the control plane (i.e., in the orchestration layer). This involves the enforcement of all the Trust Policies designed for the Global TAF.

In the Global TAF, we consider target trust propositions on routers, links, and paths. This allows the identification of the trust profile that each trust object has during runtime (i.e., each entity in the topology is attributed with a network and trust profile; they are used for the optimization process). The set of trust propositions may evolve over time as new routers are enrolled in the topology (or are detached from the topology). Similarly, the decomposition of target trust propositions may be different over time as the number of routers (and thus associated trust propositions) fluctuates.

In our case, we examine the Subjective Logic Trust Network as a trust model representation to capture all dynamic trust relationships and their trust opinions during runtime. When a router is enrolled in the topology, the instantiated trust models are empty. Nevertheless, as more routers are onboarded, the corresponding trust relationships are introduced in the trust model and the respective trust opinions are formed and maintained throughout the router lifecycle.

As part of the design phase, before the provision of the infrastructure layer with the CASTOR enablers, the Operator is responsible for specifying the requirements for candidate network elements - e.g., routers. This involves not only the secure onboarding requirements, but also the runtime guarantees that the network elements need to exhibit in order to be considered as part of the envisioned path profiles to be offered. Figure 6.13 captures the process that is followed in order to to derive the necessary policies to be enforced by the candidate network elements. As shown in the figure, the Operator provisions the

requirements that the routers need to demonstrate in a secure and verifiable manner in order to either enrol the overall infrastructure topology, or satisfy the characteristics of a path profile. Different types of network elements may be associated with different inherent risks. This is intrinsically linked with the Required Trust Level (RTL) values that need to be attained by network elements in order to satisfy the specified requirements. The derived RTL values, along with the trust model representation and the types of evidence that need to be provided by the candidate network elements comprise the Trust Policies that need to be reflected in the CASTOR DLT. Apart from auditability purposes, this allows authorized network elements to listen for new updates in specific Trust Policies in order to enforce them locally and provide the necessary information to the orchestrator (see Section 6.2.5.3). At the same time, this process allows also the instantiation of the necessary trust models in the Global TAF.

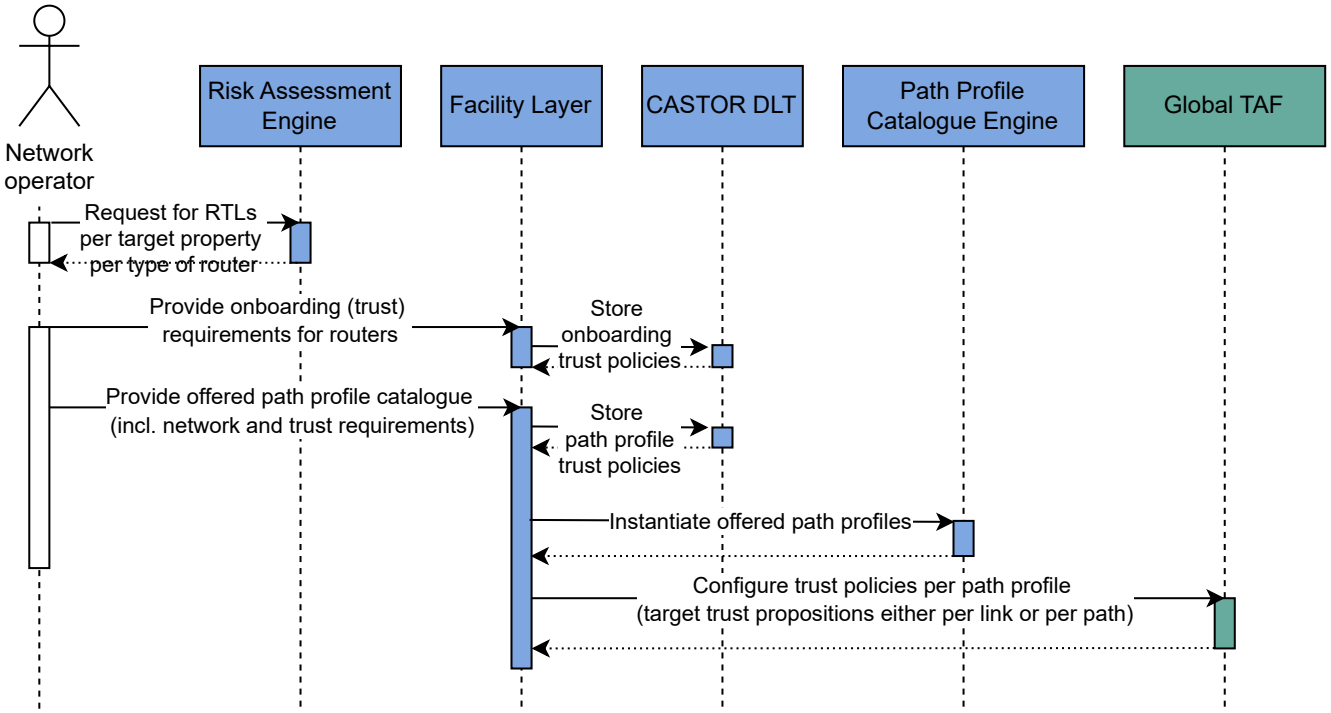

Figure 6.13: CASTOR TAF design phase

6.2.5.2 Deployment Phase

As a first step we consider the phase where a new router introduced in the managed infrastructure layer. This new element can be either spawned as a virtualized function by the network orchestrator. Alternatively, we also consider the enrolment process of actual router hardware. In either case, we envision that all the CASTOR artifacts are available as processes running in tandem with the router functionality.

The first time a router tries to enrol an existing network segment, a Setup Request is sent by the Orchestrator to the Local TAF Agent (Figure 6.14). The goal of the Setup Request is to set up the necessary items for the TAF to start the trustworthiness assessment for the application. The Setup Request contains the trust model ID, which allows the TAF to either locate a pre-stored static trust model or to send a Trust Model Template Subscription Request to the CASTOR DLT, where the path profile-specific templates are stored. A query to get the latest trust model template is automatically sent along with the Subscription Request. As a result of the query, a Trust Model Template is sent by the CASTOR DLT to the Local TAF agent. The CASTOR DLT has also registered the subscription request and will inform the router of any changes to the template during the Runtime Phase. Finally, the Local TAF agent sends an Trust Policy Subscription Request along with a query for the latest Trust Policy to the CASTOR DLT. Similarly, to the Trust Model Template Request, as a result of the query, the current Trust Policy is sent to the TAF agent.

The CASTOR DLT also registers the Trust Policy Subscription Request and will notify the router during the Runtime Phase if the Trust Policy has changed. The Trust Policy contains the Required Trustworthiness Level (RTL) for a specific path profile, which will be used during runtime to compare against the Actual Trustworthiness Level (ATL).

When the Local TAF Agent of a router is launched as part of the onboarding phase of a router, it connects to the CASTOR DLT to collect a Trust Policy to carry out its initial trust assessment process. Specifically, this refers to the onboarding Trust Policy that is associated with the specific characteristics of that particular router (e.g., type of hardware/firmware, software stack, applied security controls).

As part of the onboarding Trust Policy, we want to evaluate that the router is able to provide the necessary trustworthiness guarantees with respect to its integrity. This is our only target trust proposition that each router should satisfy in order to enrol the topology. Of course, as part of the enforced Trust Policy, the Local TAF agent receives the decomposition of the target trust proposition into atomic trust propositions. These atomic trust propositions can be measured by the Local TAF agent through the available Trust Sources that are supported by the router. Hence, the resulted trust opinions characterize functional trust relationships between the Local TAF agent and the atomic trust propositions. **In principle, the specification of the atomic trust propositions is intrinsically linked to the available evidence that we can collect and the threat model that we take into consideration**.

The Local TAF agent uses its own Trust Sources and form trust opinions about its atomic trust propositions. The in-device architecture is considered to be a singe-agent system model. This simplifies the trust model that needs to be maintained in the Local TAF agent. Specifically, this means that the trust model of the Local TAF agent is a simple key-value database that keeps track of the quantified trust opinion over the trust relationships between the analyst node (i.e., the Local TAF agent) and the atomic trust propositions. Ideally, each trust opinion is mapped to a single type of evidence collected from a single Trust Source (i.e., we should avoid duplication in the available evidence for a specific trust proposition). The quantification function that maps the available evidence to a trust opinion is provided in the Trust Policy. Nevertheless, it may be the case that multiple evidence refer to the same atomic trust proposition (e.g., the runtime integrity of the software stack of a router may stem from evidence coming from the Attestation and FSM sources). In such cases, the Trust Policy should provide the quantification function to aggregate the evidence and derive the trust opinion for that particular atomic trust proposition.

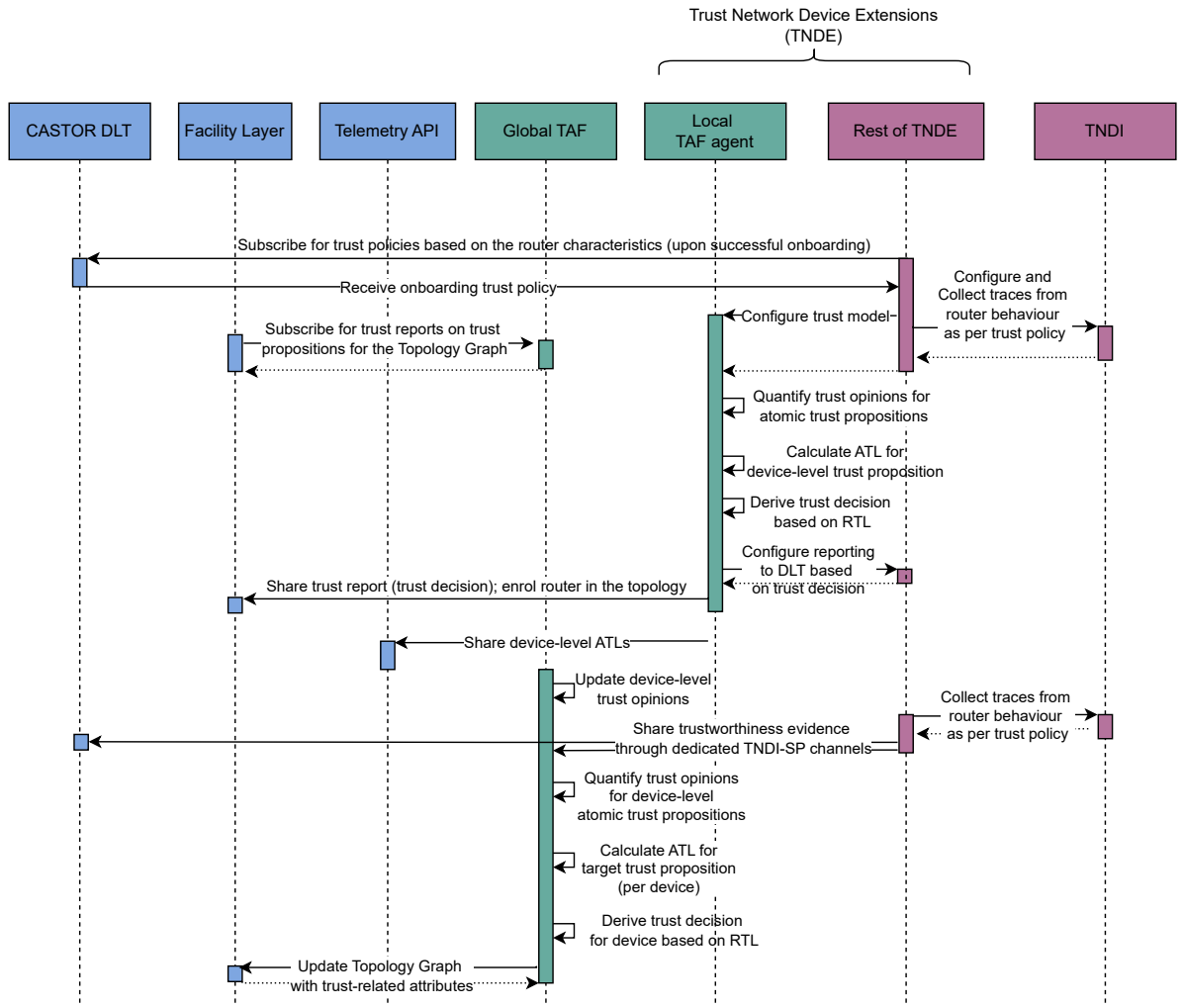

Figure 6.14: CASTOR TAF router deployment phase

6.2.5.3 Runtime Phase

From a Local TAF agent's perspective, there is no major difference in the sequence of actions between the onboarding and the runtime phase; only the Trust Policy changes to provide sufficient trust guarantees (i.e., beyond integrity) for each one of the path profiles. This is also reflected in Figure 6.15, where the first half of the diagram is capturing similar operations on the Local TAF agent in Figure 6.14.

As mentioned in the introduction, the main purpose of the Global TAF is to continuously assess the trustworthiness of the entire infrastructure layer and, thus, enable the selection of optimal paths for the offered path profiles that satisfy the requested application workloads. From the perspective of the global Trust Policy specification, this gets mainly reflected to the identification of the target trust propositions that map to the offered trust guarantees per path profile. As already highlighted, this guarantees may be node-, link- or path- centric. Hence, the Global TAF should be capable of characterizing the trustworthiness of the underlying topology graph for each one of the offered trust properties. This attribution of nodes, links and paths in the topology unlocks the optimal selection of paths that need to be established to serve the application workloads with the agreed network (SLA) and trust (SSLA) guarantees.

Overall, there are several critical processes that are involved in the trustworthiness assessment during the router's runtime phase. The established Trust Policy contains parameters which indicate if the ATL should only be sent once (synchronised TAR), sent periodically (periodic TAR), or sent whenever the ATL changes (event-based TAR). A periodic TAR could be helpful in situations where the trustworthiness of the trust object changes frequently because new evidence is constantly arriving.

Based on the trust model (template), the TAF agent decides which evidence should be collected from the router environment. The Trustworthiness Claims provide guarantees on specific trust propositions - e.g., that the hardware and software of the router are in a correct configuration state. In fact, the TAF can also request evidence about the software stack from the TNDE, which then provides the Attestation Evidence to the TAF agent. Finally, the TAF can also request results from the FSM source which may detect abnormal behaviour in one of the device-level state diagrams that capture the different router functionalities as dictated by the trust policy.

Once the TAF collects all the evidence needed, it can perform a trustworthiness assessment on the relevant data and produce a set of ATLs for the target trust propositions. The ATLs are then either forwarded to the orchestrator that sent the TAR or compared to the RTLs to produce a set of Trust Decisions to be sent to the CASTOR DLT or its neighbouring routers.

In general the Global TAF should be able to cope with a multitude of trust propositions referring to various trust properties. First and foremost, as shown in the second half of Figure 6.15, the Global TAF receives and quantifies an ATL value for atomic trust propositions. In a second stage, it may use logical SL operators to derive the target (composite) trust propositions that best characterize the requirements for the offered path profiles.

A trust model template and trust policies update process can also occur if the TAF is notified by the CASTOR DLT that there have been changes. In this case, the TAF queries the CASTOR DLT for the updated version of the template and the policies, which then forwards the updates to the TAF.

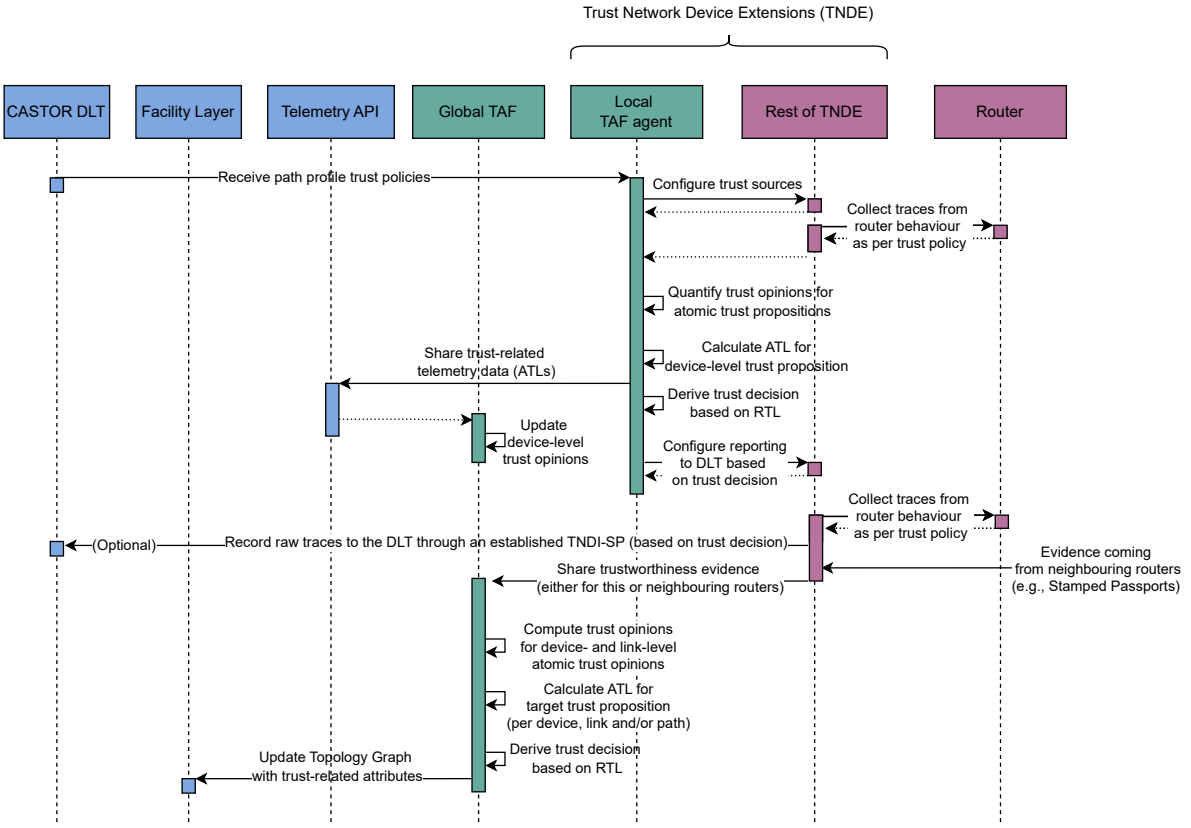

Figure 6.15: CASTOR TAF runtime phase

6.2.6 On-board Finite State Machine Analyser

Evaluating and assessing the trust associated to a target is a complex multidimensional problem that can be facilitated the more sources of trust evidence are available to be analysed to enable the most accurate trust evaluation possible. For example, the Attestation Source is capable of assessing the static

information of the router by analysing information related to its current configuration and memory usage, but has limited visibility and struggles in identifying and assessing the runtime dynamic aspects of the router during its operational phase. Furthermore, if we extend the scenario by including the requirements provided by the current threat model of interest for the router, the dynamic information of the router gain even more importance, requiring specific models aimed to analyse its behaviour and identifying the potential threats currently affecting its functionalities. It is also important to consider the computational restrictions tied to the router and to limit the impact that the security functionalities introduced on the platform have on the normal operational services of the router itself, which is why we opted towards a runtime lightweight model to attest the dynamic behaviour of the router represented by a finite state machine (FSM). The FSM analyser - or FSM source to denote its functionality of a Trust Source in the overall CASTOR Trust Assessment Framework - is capable of generating Trust Sources for the trust evaluation process, analysing runtime behavioural information of the router to be shared to the Local TAF Agent enabling it to perform a more complete and holistic evaluation of the router. As described in [72], the FSM will learn an FSM based on the traces collected at runtime and minimises the model to its minimum number of states without losing accuracy on the information which represents. Clearly, the process is highly dependent on the data fed to the learning process, which vary depending on the threat model landscape considered for the analysis and the nominal behaviour, or behaviours, of the router under study that are required to be represented by the FSM model. A preliminary study of the nominal router behaviours will be performed to identify which set of information are required to be collected at runtime to best represent them through the FSM model.

By relying on specific runtime information collected by the tracer, the FSM analyser performs a runtime evaluation of its ad-hoc router model and provides its evaluation in the form of behavioural evidence to the TAF agent. To configure the FSM analyser and the tracer, the TN-DSM will share a trust policy containing, but not limited to, the (i) information related to what kind of tracing data the tracer needs to collect to enable the the FSM analyser to perform its runtime evaluation, and (ii) the latest router behavioural model to be used by the FSM analyser for the evaluation and assessment of the collected runtime traces, effectively enabling the runtime evaluation and generation of its related Trust Source. At this point, both the tracer and the FSM analyser are assumed to be correctly configured and operational.

While the router is operating, the tracer collects and shares data that are relevant to the FSM analyser, which then performs its runtime behavioural evaluation of the router functionalities, identifying if the behaviour of the router is consistent with the expected model or if there are any discrepancies. In the latter case, the FSM analyser will generate the trust evidence related to the incident, raising an alert to the Local TAF Agent that contains information about the type of misbehaviour detected and related details about the runtime actions leading to the alert. Depending on the customer preferences, any kind of evidence can be stored on the CASTOR DLT through the TNDI-SP for further future inspection.

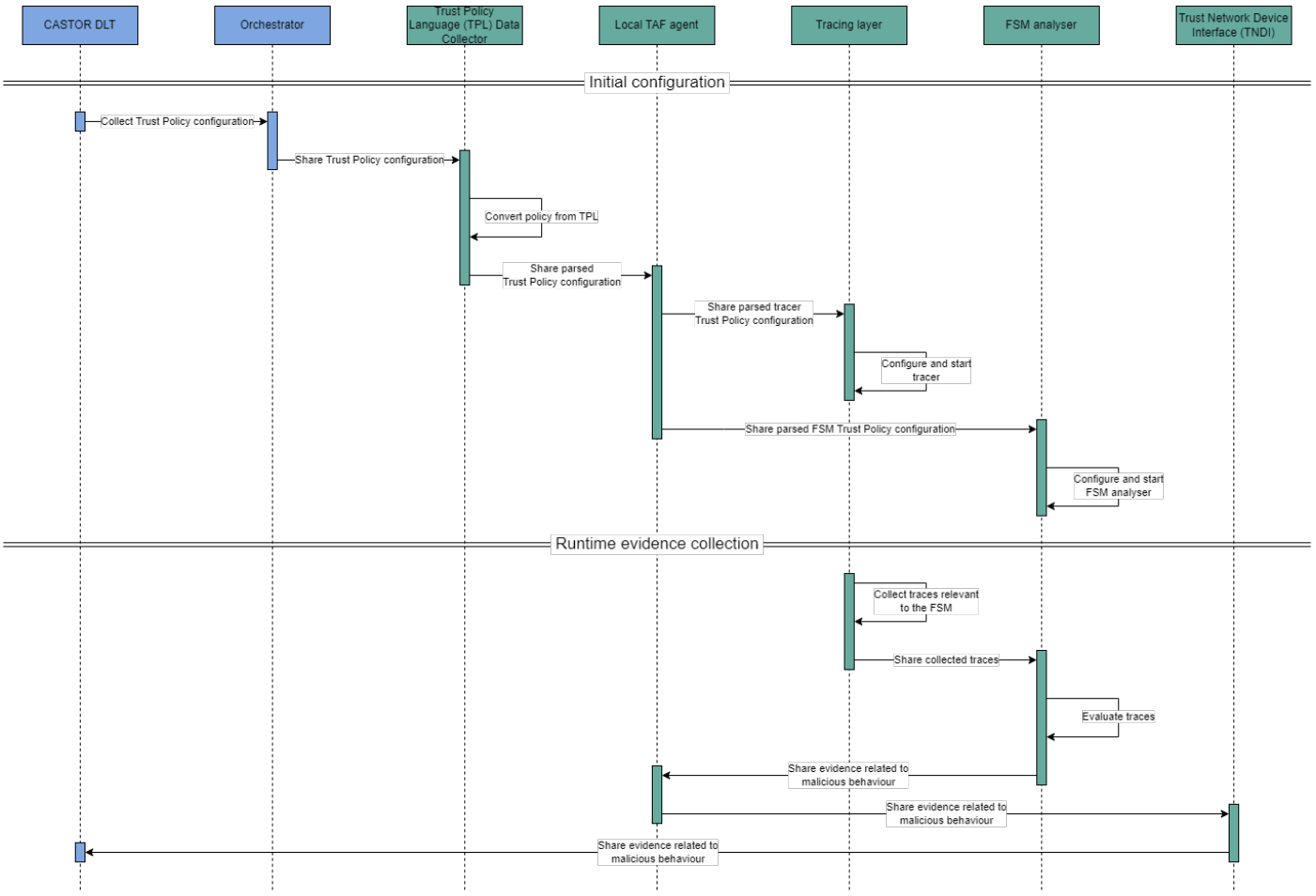

Figure 6.16: Onboard Finite State Machine analyser flow of actions

6.2.7 Optimization Engine

The Optimization Engine, as shown in Figure 6.17, in CASTOR is responsible for identifying and recommending optimal network paths between ingress and egress nodes, considering both intra- and inter-domain topology. At the core of the Optimization Engine is a multi-objective Design Space Exploration (DSE) algorithm operating on network-level (e.g., latency, bandwidth) and trust-level (e.g., integrity, confidentiality) constraints defined by user-defined SSLAs. The Path Profile Catalogue Engine facilitates the mapping of all SSLA objectives to path profiles. This mapping articulate domain-specific network and trust-related requirements. The Facility Layer serves as the interface that exposes this information to the Optimization Engine. The Topology Graph, - maintained within the Facility Layer - provides the Optimization Engine with an up-to-date representation of the network topology.

The Optimization utilizes Quantum Annealing to efficiently navigate the large and complex design space of potential paths. The Optimization Engine reformulates the multi-objective multi-constraint trust path routing problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem. Binary variables within the QUBO represent inclusion of nodes and links in the solution path, optimizing the objective function encoded as trust and performance metrics. A Quantum Annealer within the Optimization Engine then solves the formulated QUBO problem.

6.2.7.1 Quantum-Anneal-Based Trust Path Selection

The QUBO model incorporates hard constraints, such as meeting the minimum Required Trust Levels defined by the Global TAF, along with soft optimization goals like minimizing total latency by avoiding

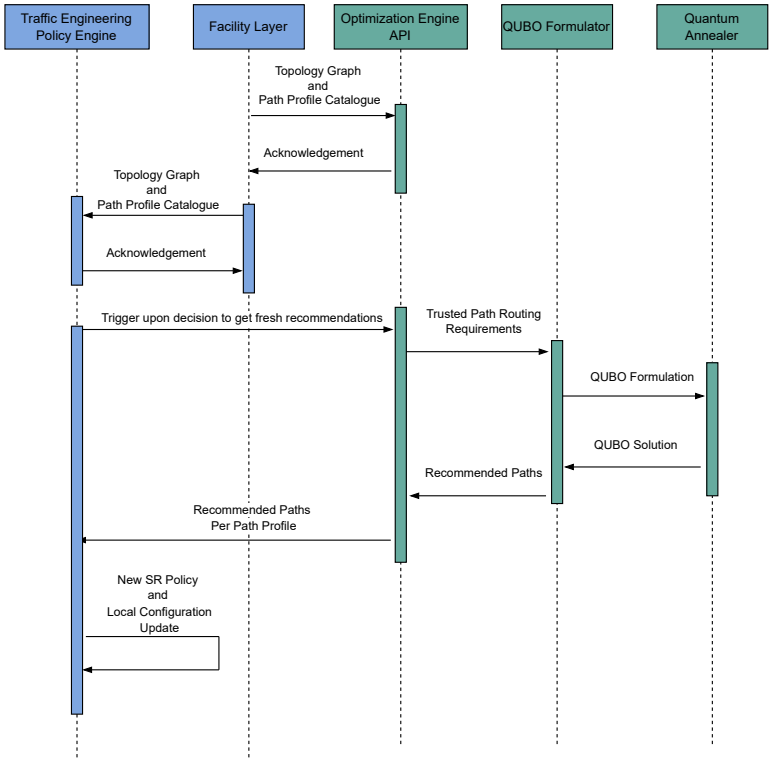

Figure 6.17: Optimization Engine action flow

congested or degraded links. The QUBO formulator translates each constraint and objective into a term in the QUBO formulation. The constraints (and objectives) come from the path profile requirements. The attribution in the actual topology comes from the topology Graph maintained by the Facility Layer. The trust-related attribution is coming from the Global TAF, while the network performance is coming through the Telemetry API. Finally, a penalty function discourages the inclusion of untrusted or degraded links. Subjective logic determines the quality of a path by analysing the nodes and links that form the path.

The Optimization Engine then submits the problem to a quantum annealer, which probabilistically explores the design space to identify the optimal (or near-optimal) paths. The use of Quantum Annealing reduces the likelihood of convergence to local optima (over classical solvers), especially in large, dynamic, constraint-heavy network environments targeted within CASTOR.

6.2.7.2 Integration with Segment Routing

The Optimization Engine assigns a unique colour for each tier in the path profile catalogue. The Optimization Engine passes its optimal path recommendations to the Traffic Engineering Policy Engine. The Traffic Engineering Policy Engine then converts the Optimization Engine output into Segmented Routing Policies. The Path Computation Element. as part of the orchestration layer, enforces these Traffic Engineering policies (e.g., Segment Routing configuration) at the local level for a scalable and distributed solution.

The optimal path defined by the Segment Routing policy may encompass one, or multiple candidate paths for all entries in the Path Profile Catalogue that satisfy the trust-based selection criteria using subjective logic. The paths thereby adhere to both network and trust requirements, as envisioned in CASTOR. Candidate paths can also include fallback or backup segments that are pre-computed by the Optimization Engine to support fast re-routing in case of a runtime violation. The Optimization thereby supports programmable path validations and a fallback mechanism that aligns with CASTOR's zero-touch automation goals.

6.2.7.3 Integration with Facility Layer and TAF

The Optimization Engine continuously consumes the latest insights that are collected at the Facility Layer so as to provide fresh and timely recommendations based on the latest topology state. Specifically, the Topology Graph information consists of network telemetry data via the Telemetry API, and trust metrics via the Global TAF. Whenever a new service intent is received or when a topology change occurs (e.g., due to a trust violation or a node joining/leaving), the Traffic Engineering Policy Engine triggers the Optimization Engine to recalculate the optimal paths. The Optimization engine then provides new paths incorporating both the latest network conditions and the current trust assessments.

6.2.8 Trustworthy Platform Attestation and TNDI Onboarding/Runtime

Before a Trust Network Device Interface (TNDI), representing, e.g., a vRouter instance or a physical router, can participate in CASTOR's trusted path routing, the underlying platform must first join the CASTOR domain and then the TNDI must be securely onboarded into the network. This includes the CASTOR orchestrator attesting the Trusted Network Device Extensions (TNDE) of the device platform and the subsequent attestation and trustworthy configuration of the selected TNDI. In addition, the TNDI onboarding includes the establishment of secure communication channels from the TNDI to CASTOR's upper layer components (e.g., Global TAF) and to the neighboring TNDIs in the network domain in order to share attestation-related information.

In the following, we will present this platform/TNDE join phase and TNDI onboarding phase, as well as the transition into the subsequent runtime phase. We will describe for each phase the high-level flow of actions between the CASTOR upper layer components, the device-side components (especially the TNDE), and the neighboring TNDIs.

6.2.8.1 Platform/TNDE Join Phase

Before TNDIs of a network device can be onboarded, the underlying platform must be added to the CASTOR domain, i.e., join the domain. As shown in the upper part of Figure 6.18, the CASTOR Facility Layer starts by establishing a connection to the TNDE of the network device and remotely authenticating and attesting the platform. The TNDE forms the device-side TCB (together with the TNDE-operated Trace Units) and executes in a HW-based TEE with platform-bound identity and attestation keys rooted in secure hardware. The TN-DSM serves as the TNDE communication endpoint for the Facility Layer and manages the TNDIs of a network device. After TNDE authentication and attestation, the Facility Layer and TNDE can securely exchange metadata and configurations required to complete the join procedure.

6.2.8.2 TNDI Onboarding Phase

After the platform TNDE has joined the CASTOR domain, the orchestrator can select one of potentially multiple TNDIs (e.g., vRouter instances) managed by the TNDE and start the onboarding process. As shown in Figure 6.18, in the TNDI onboarding phase, the CASTOR Facility Layer establishes trust in the TNDI via the TNDE and configures the TNDI for the runtime trust assessment required for CASTOR's trusted path routing. First, the Facility Layer establishes a TNDI-SP control channel via the TN-DSM for that TNDI, which is bound to the attested session between Facility Layer and TN-DSM. The Facility Layer then queries relevant TNDI metadata (e.g., vendor information) and measurements for the TNDI attestation, exchanges cryptographic keys (e.g., attestation keys for runtime evidence), and configures the TNDI with information on the CASTOR upper layers (e.g., communication endpoints), enforced by the TN-DSM. The TN-DSM then configures the TPL Data Connector of the TNDE accordingly to retrieve the CASTOR trust policy for the TNDI from the CASTOR DLT. The TPL Data Connector then shares the

trust policy with the TN-DSM and enforces it at the Local TAF Agent. On trust policy initialization, the Local TAF Agent configures its two Trust Sources (Attestation and FSM Source) for the runtime evidence generation based on traces collected for the TNDI. The TN-DSM configures the Tracing Hub accordingly to prepare the collection of configurational and behavioral runtime traces for the properties specified in the trust policy.

After the tracing and trust assessment components have been configured, the Facility Layer asks the TN-DSM (via the TNDI-SP control channel) to bootstrap the TNDI-SP data channels required for sharing runtime traces, evidence, and ATLs with the CASTOR upper layer components: the Global TAF and DLT. The TN-DSM establishes the requested data channels with the Global TAF and DLT, bound to the TNDI via the TNDI-SP control session. Finally, the TN-DSM might push network configurations to the TNDI to prepare it for the participation in the CASTOR topology and trusted path routing. At this step, the TNDI might already start establishing secure network links with neighbouring TNDIs by exchanging their attestation information (e.g., boot measurements). As the exact time this takes place might be flexible and as it might repeat periodically, potentially enhanced with runtime evidence, we will cover this further as part of the runtime phase. After that step, the configuration of the TNDE components and the TNDI is complete and the TNDI is ready to enter the runtime phase.

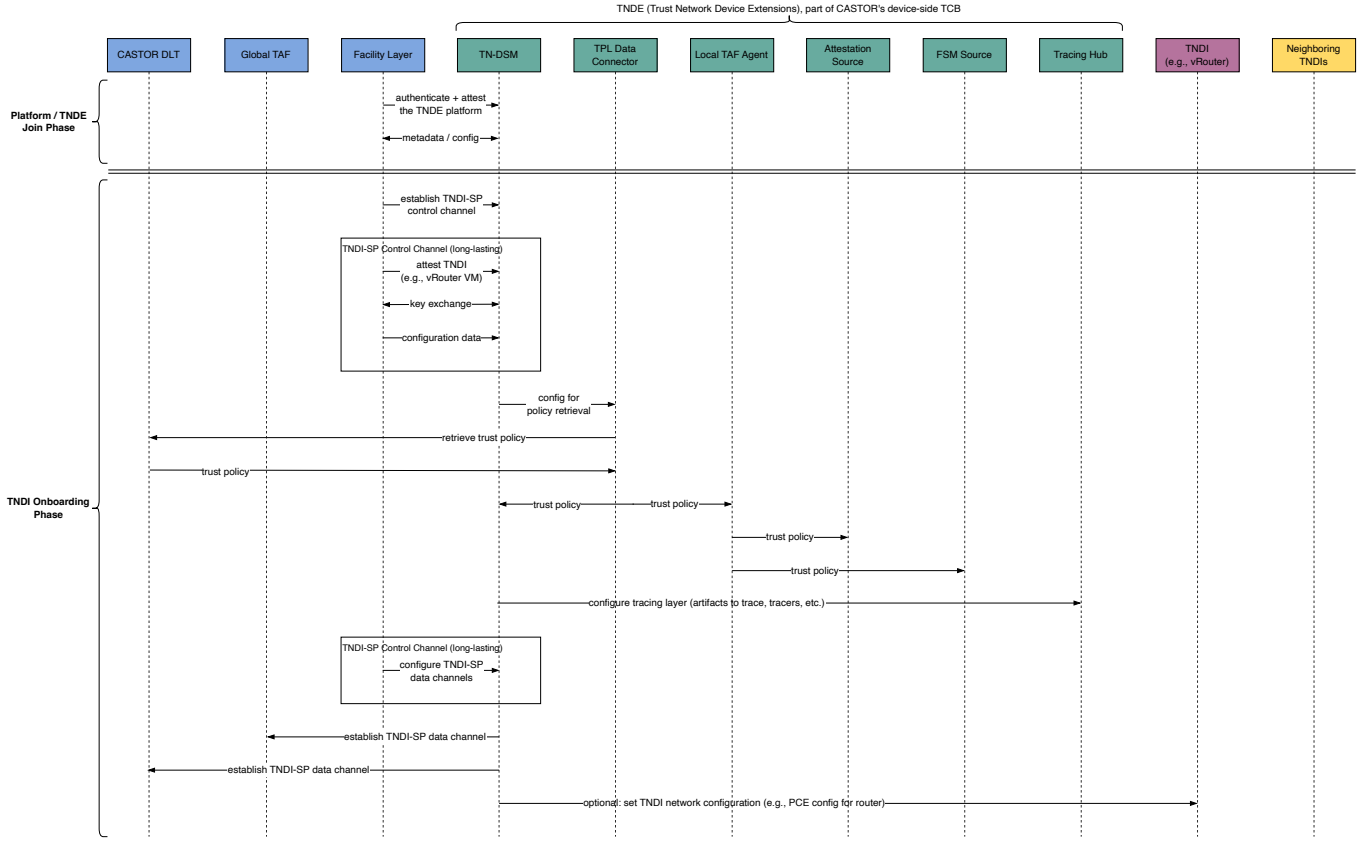

Figure 6.18: TNDE and TNDI flows during join and onboarding phases (attestation and configuration)

6.2.8.3 TNDI Runtime Phase

After a TNDI has been successfully onboarded to the CASTOR domain, the TNDI can enter the runtime phase in which the TNDI participates in CASTOR's trusted path routing. If not already done as part of the final onboarding steps (cf. previous section), first, the TNDI establishes protected links with trustworthy neighbouring TNDIs (of that CASTOR domain) in order to securely exchange network traffic. For the secure link establishment, CASTOR builds on top of the IETF trusted path routing (TPR) proposal [27], exchanging evidence-containing stamped passports between TNDIs. The TNDI interfaces with the TN-DSM to craft the stamped passport, including attestation information on the TNDI platform (e.g., including

boot measurements of the TNDI-associated vRouter). The TNDI then exchanges the passports with neighboring TNDIs using broadcast-based communication and bootstraps secure links with successfully verified neighbors (e.g., based on MACsec). The TNDIs can then securely exchange routing information and forward packets via the secure links. The TNDIs might be configured to periodically re-exchange passports, potentially incorporating additional runtime attestation information generated as part of the local trust assessment process.

During the runtime phase, the tracing and trust assessment for the TNDI is active, with the TN-DSM sharing traces, evidence, and ATLs via the TNDI-SP data channels with the CASTOR upper layer components. That way, CASTOR can identify and enforce trusted paths based on the trust levels of the TNDIs and their interconnecting links. To establish an ordering of trust-related information across all TNDIs, CASTOR will explore different strategies. For instance, CASTOR could build on top of hardware-rooted monotonic counters, or the CASTOR Facility Layer could periodically (depending on the setup) push new time markers to the TN-DSM, e.g., in the form of CASTOR-managed IETF epoch markers [26]. When the Tracing Hub collects configurational and behavioral traces of the TNDI, it distributes them to the Attestation and FSM Trust Sources for evidence generation (see subsection 6.2.9). The Attestation and FSM Sources will forward the traces and generated evidence to the TN-DSM and to the Local TAF Agent. The Local TAF Agent can then perform the ATL calculations and trust report constructions and share the resulting ATLs and reports with the TN-DSM. The TN-DSM shares the calculated local ATLs and evidence via the pre-established TNDI-SP data channel with the Global TAF for the global trust assessments. In addition, the TN-DSM can publish (a subset of) the traces, evidence, and trust reports for audit purposes at the CASTOR DLT through the respective TNDI-SP data channel. Finally, the TNDI periodically shares network telemetry data with the CASTOR telemetry service (e.g., latency and bandwidth statistics), and the TN-DSM can optionally share the trust reports as additional, trust-related telemetry data.

This concludes the overview of the TNDI-related flows of actions. Note that the described operations of trustworthiness data generation and sharing, freshness renewal (time markers), and exchange of stamped passports happens periodically or triggered by external events (e.g., a new neighboring TNDI getting onboarded), and might slightly vary in their order of occurrence.

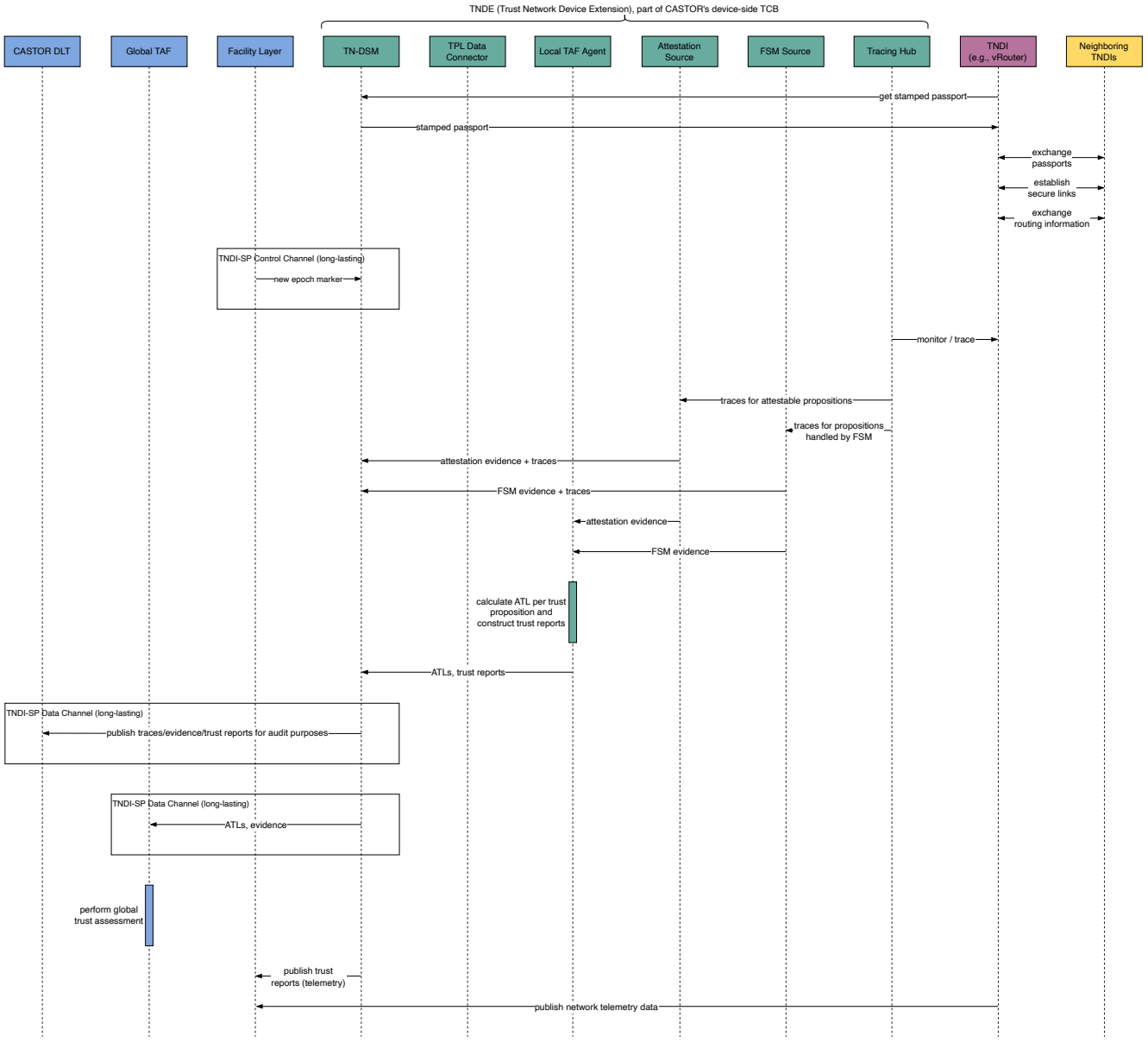

Figure 6.19: TNDE and TNDI flows during the runtime phase (trust assessment and data sharing)

6.2.9 CASTOR Tracing Capabilities

The Tracing Hub on each network device is crucial to provide runtime traces for the trust assessment of each TNDI. The Tracing Hub is part of CASTOR's TNDE and responsible for collecting configurational and behavioral runtime traces for each TNDI, e.g., capturing activities of the routing stack. The collected traces form the basis for the trustworthiness evidence generation and are therefore forwarded to the Attestation and FSM Trust Sources. The Trust Sources generate runtime evidence based on the traces and forward the evidence to the Local TAF Agent for the ATL calculations of each TNDI. As was shown in Figure 6.19, the traces, evidence, and ATLs are additionally forwarded to the upper layer CASTOR components by the TN-DSM for the global trust assessment (Global TAF) and for auditing purposes (DLT). The way the runtime traces are collected and shared with the TNDE and global CASTOR components has a direct impact on the completeness and precision of the evidence and ATLs, the required network bandwidth, and the latency of an incident detection, e.g., the time to react on the decrease of a TNDI's ATL.

In the following, we provide an overview of the tracing architecture and the tradeoffs of different modalities for exchanging the traces and/or evidence across the CASTOR components.

6.2.9.1 Multi-level tracing architecture

CASTOR envisions a flexible multi-level tracing architecture that allows for multiple Trace Units (i.e., tracers) and dynamic configuration by the CASTOR operator to achieve different tracing tradeoffs (e.g., performance vs. granularity). The TNDE is responsible for configuring the Tracing Hub on behalf of the CASTOR operator based on the trust policy and configuration information provided as part of the TNDI onboarding process (cf. subsection 6.2.8). Depending on the propositions for which traces and evidence need to be generated, the TNDE can ask the Tracing Hub to operate an appropriate subset of Trace Units that can collect suitable configurational and/or behavioral runtime traces from the TNDI. The Trace Units are the TNDE-external trace mechanisms interfaced by the Tracing Hub to perform the trace collection. The evidence-generating Trust Sources of the Local TAF Agent, i.e., the Attestation Source and FSM Source, can then subscribe to the required traces that they need to process. As the TNDIs might be based on different software and/or hardware platforms, e.g., physical or virtual routers of different vendors, the way the different Trace Units are implemented and controlled by the Tracing Hub at runtime might vary, and the evidence-generating Trust Sources might need to adapt to TNDI-specific traces accordingly (e.g., loading vendor-specific FSMs). However, CASTOR uses a common data format for encoding the traces to provide support across different Trust Sources.

For CASTOR's multi-level tracing architecture, we plan to explore two types of Trace Units operated by the Tracing Hub: (1) a memory-based introspection one that is isolated from the TNDIs, and (2) an OS-level one integrated into a TNDI's software stack (e.g., router NOS), as shown in Figure 6.20. The two Trace Unit types provide different tradeoffs w.r.t. their tracing primitives, security, performance, and platform integration, as summarized in Table 6.2. By considering both, we can explore which tradeoffs are best suited to collect runtime traces required for different trustworthiness evidence of a TNDI, backed by practical experiments.

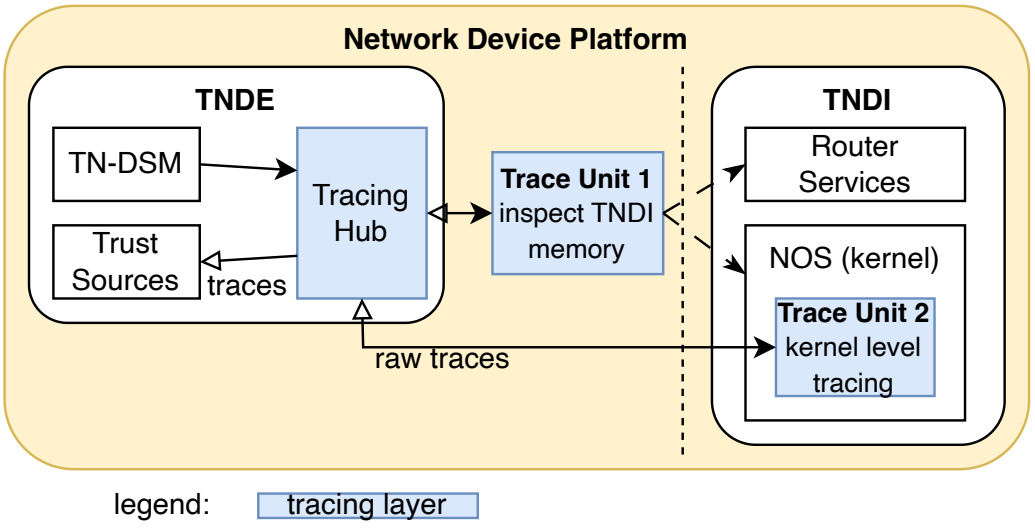

Figure 6.20: CASTOR's multi-level tracing architecture proposal

Trace Unit 1: Out-of-TNDI Memory Inspection The first Trace Unit that we consider is based on memory inspection and is located outside the monitored TNDI. That is, the tracer is isolated from the TNDI, e.g., co-located with the TNDE in a TEE of a physical router or isolated from a vRouter by a hypervisor, providing strong protection against a compromised TNDI. The Trace Unit reads from the memory address space of a TNDI to inspect its runtime state, monitor it for changes, and extract relevant configurational and/or behavioral information, similar to existing forensics and VM introspection approaches [96, 130]. That way, the tracer collects traces based on the live memory state and changes of a TNDI. The Trace Unit's ouf-of-TNDI design provides strong isolation guarantees and is non-intrusive in that it does not

Table 6.2: Comparison of the Trace Units considered for CASTOR.

Property	Trace Unit 1	Trace Unit 2
Location	outside TNDI (e.g., in TEE)	in TNDI (e.g., in network OS)
In-TNDI Changes	no	yes
Primitives	memory reads	(vetted) memory reads, inline tracepoints
Arbitrary Memory Access	yes	depends (e.g., eBPF: no)
Difficulty of Data Access	mid	low
Performance	low–high	(mid–)high
Security	high	low–mid
Portability	mid–high	low–mid

require changes within the TNDI software stack. Furthermore, by relying solely on reading and parsing a TNDI's memory, the tracer is easily portable to other network device platforms, TEEs, or other forms of isolated execution environments (e.g., DMA-capable peripherals). On the downside, memory inspection cannot perform inline behavior tracing without polling-based approaches [111] or expensive trap mechanisms, and it needs to deal with the semantic gap challenge, i.e., needs to manually locate and parse data structures as it executes outside the TNDI address space.

Trace Unit 2: In-TNDI OS-level Tracer The second Trace Unit that we consider is an OS-level tracer that uses mechanisms integrated into the TNDI software stack. Many existing network devices have a Linux-based OS [129] and therefore support Linux tracing mechanisms (e.g., eBPF) with static (or dynamic) inline tracepoints in the kernel. This Trace Unit provides no isolation against a full kernel-compromise of the TNDI (i.e., arbitrary code execution), however, it complements the previous Trace Unit with fast inline tracing capabilities for the collection of more fine-grained behavioral traces (e.g., on system calls). In addition, it can directly access tracer-exposed kernel data structures as it operates in the TNDI memory address space, in contrast to the previous tracer which requires additional parsing and translation effort. However, the Trace Unit might have access permissions only to a subset of the available kernel structures for security/stability reasons (e.g., eBPF). In addition, as this tracer requires direct integration into the TNDI software stack, this tracer provides weaker security guarantees for the traces, introduces new components into the TNDI (intrusive), and is less portable to other platforms. The tracing mechanisms need to be enabled for the specific vendor TNDI platform (e.g., router NOS), even if it is based on Linux.

By focussing on the secure out-of-TNDI tracer as the primary secure tracing source (unit) and enabling the more fine-grained but less secure in-TNDI Trace Unit on demand, CASTOR's multi-level tracing architecture provides the operator with strong flexibility for effective runtime tracing and trust assessments.

6.2.9.2 Sharing of Traces and Evidence

The traces collected by the Tracing Hub are shared with the Trust Sources to generate evidence for the trust assessment and can be shared (together with evidence/ATLs) with upper-layer CASTOR components through the TNDI-SP data channels, as previously shown in Figure 6.19. The Trust Sources can subscribe to the type of traces they require to generate the evidence expected by the Local TAF Agent. What types of traces or evidence the TN-DSM shares with the CASTOR upper layer components, e.g., the Global TAF and CASTOR DLT, is configured by the CASTOR orchestrator via the TNDI-SP control channel as part of the TNDI onboarding process (cf. subsection 6.2.8). While not the main focus, note that the TN-DSM can optionally share trust-related telemetry information with the upper layer services via CASTOR's Telemetry API (e.g., telemetry of the Local TAF Agent).

Traces can be passed to the Trust Sources and upper layer components in different ways, resulting in different tradeoffs. We therefore plan to experimentally explore some of the following strategies and asses which are best suited in the context of CASTOR's trusted path routing, considering, e.g., the resulting detection latency (of ATL changes), memory buffering, and network overhead.

Continuous vs. Operation-based The traces can be continuously passed to the Trust Sources, resulting in a minimal latency for the evidence generation and no need for buffering of traces. However, this implies that the traces might not capture a full operation, e.g., only a subset of the behavior shown during a routing table update, requiring a stateful operation of the Trust Sources (e.g., the FSMs). Alternatively, the Tracing Hub could be aware of the start and end of evidence-relevant activities and share only batches of traces that capture a full operation. That way, the Trust Sources could operate in a stateless way, receiving sets of full traces based on which evidence is generated. However, this strategy assumes that the Tracing Hub can guarantee a clear separation of traces for different evidence-relevant operations. In addition, this strategy might increase the detection latency as the delivery of some traces is delayed.

Sharing on ATL Change Another sharing modality decides if all generated traces and evidence should be shared, or only those that indicate a change in the TNDI's trust level. Such a policy could be implemented either at the Tracing Hub, Trust Source, or Local TAF Agent level, depending on when it is possible to identify a potential change in the ATL. For more complex behavioral traces/evidence, this might be difficult to achieve without an expressive state model as provided by the FSM Source. For more static, configurational traces/evidence, it might be possible to keep a minimal state at the Tracing Hub or Attestation Source level to detect changes that require sharing with the Local TAF Agent and CASTOR upper layer components. The advantage of such a change-based sharing strategy would be a decrease in network bandwidth overhead for remote sharing and a potential decrease in processing overhead on the local devices.

Remote: Push vs. Pull For the remote sharing with the upper layer CASTOR components via the TNDI-SP data channels (cf. subsection 6.2.8), CASTOR can follow a push- or pull-based approach. The push-based approach will enable a lower detection latency as new traces and evidence are shared immediately. However, the network will also face constant bandwidth overhead and the CASTOR upper layers might get overloaded when the network includes a high number of TNDIs, requiring buffering of traces and evidence at the upper layer components. A pull-based approach sacrifices lower detection latency for more traffic control by the upper layer. However, it implies the need for buffering of traces and evidence at the device-sides, potentially requiring a limited time window for which traces/evidence can be provided on resource-constrained devices.

Remote: Sharing Traces vs. Evidence For the remote sharing with the upper layer CASTOR components, sharing the (raw) traces generated by the tracing layer might introduce a non-negligible network bandwidth overhead. The size of traces generated by fine-grained behavioral tracing can be high, such that for some propositions it might be better to share only the evidence generated by the Trust Sources.

6.2.10 Composite Attestation

In CASTOR project, trust establishment is achieved through a composable attestation mechanism. Within a given network segment, each router possesses the capability to autonomously generate a cryptographically signed claim reflecting its current trust level and integrity status. These individual claims, originating from distinct routers along a potential path, are systematically collected and assembled to form a composite proof representing a trustworthy path. The orchestrator, acting as a centralized trust authority,

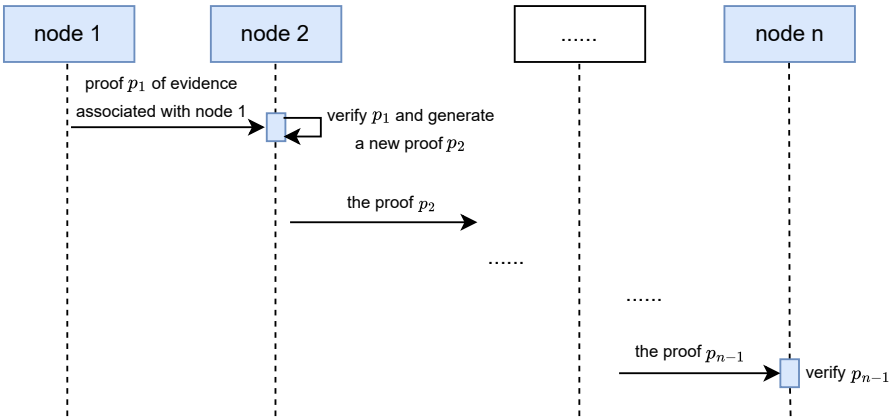

Figure 6.21: The workflow of composite attestation

can then perform a collective attestation of the entire segment of routers. This process yields verifiable evidence attesting to the overall integrity of the constructed path. Furthermore, the trust model extends beyond a single segment; the attested claims are inherently designed to be verifiable by and linked to those from neighbouring segments. Consequently, routers at segment boundaries may selectively verify different subsets of a combined claim set, a process governed by dynamically defined attestation policies that dictate the required scope and depth of verification.

From a cryptographic perspective, the primary objective is to engineer a secure system for trusted path routing across a collection of trustworthy network devices, such as virtual routers. The conceptual model, illustrated in Figure 6.21, assigns each router the role of a signer. Each signer generates a digital signature over its trust claim, serving as unforgeable proof of its trust level. A critical feature of the system is the ability to aggregate these individual signatures into a compact composite proof. This aggregation allows any verifying entity—be it another router or the orchestrator—to efficiently verify the entire set of claims or, importantly, any arbitrary subset thereof, depending on the verification context.

The process of establishing a trusted path routing can be formalized as follows. We assume a network comprising n nodes, each capable of acting as a prover/verifier. The protocol proceeds sequentially along the path: an initiating prover node 1 sends its proof p_1 to the next node 2. Upon receipt, node 2 first verifies the validity of p_1. If and only if the verification is successful, node 2 generates a new, cumulative proof p_2. This proof p_2 is computed based on his/her own trust evidence and the previously verified proof p_1, effectively chaining the trust assertions. This iterative step—verification followed by proof generation—is repeated hop-by-hop until the path is established at the final node p_n. Analyzing this interactive protocol reveals several fundamental cryptographic requirements that must be satisfied to ensure its security and practicality as follows:

1. There are multiple provers/verifiers.
2. Multiple proofs can be aggregated/combined one by one in sequence.
3. Any verifier can verify part of the aggregated/combined proofs.
4. Revocation of rogue devices is needed.

We call this kind of proof and verification process composite attestation. To achieve composite attestation, we need a basic digital signature and a method to achieve aggregation. Here, we give generic definitions of a basic digital signature and an aggregatable signature:

Definition of a basic digital signature. A basic digital signature is denoted as **SIG** $= (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, in which details of all algorithms are described as follows:

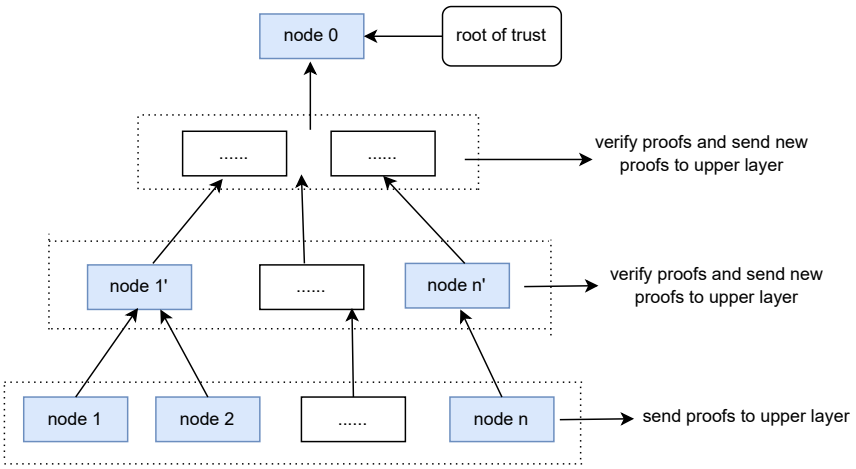

Figure 6.22: The structure of layered attestation

- KeyGen: This key generation algorithm takes the security parameter λ as input and outputs a pair of secret/public key, i.e., (sk, pk).
- Sign. Given a secret key $sk = x$ and a message to be signed $M \in \{0,1\}^*$, this signing algorithm generates a signature, σ.
- Verify: Given a public key pk, a message $M \in \{0,1\}^*$, and a signature σ, this verification algorithm verifies whether the signature σ is valid or not. If yes, output '1' for accepting this signature; otherwise, output '0' for rejecting this signature.

Based on this basic signature definition, as mentioned in subsection 2.2.7, to achieve composite attestation, we can consider about ordered multi-signature/layered signature attestation. The structure of multi-signature is more or less the same as that shown in 6.21. While the layered attestation is different. As shown in 6.22, there are multiple layers. Every lower layer can generate aggregated proofs, which will be sent to upper layer. The upper layer will verify the proofs and generate new aggregated proofs. This process will be repeated until the top layer. The top node meets root of trust. In summary, the aggregated signature (ordered multi-signature/layered attestation) **aSIG** $= (\text{aKeyGen}, \text{Sign}, \text{aSign}, \text{aVerify})$ is described as follows:

- aKeyGen: each signer i makes use of the KeyGen algorithm to generate a secret/public key pair, i.e., (sk_i, pk_i).
- Sign: Given a secret key sk_i and a message to be signed $m_i \in \{0,1\}^*$, this aSign algorithm generates a signature σ_i for the signer i.
- aSign: This algorithm is used to aggregate multiple signatures by one entity. Given multiple signatures σ_i associated with multiple distinct messages m_i, $i \in [n-1]$, an entity can generate an aggregated signature $\sigma = \prod_{i=1}^{n-1} \sigma_i$ by aggregating all signatures σ_i, $i \in [n-1]$.
- aVerify: Given all public keys pk_i, messages $m_i \in \{0,1\}^*$, $i \in [n]$, and an aggregated signature σ, this aVerify verifies whether the aggregated signature σ is valid or not. If yes, output '1' for accepting this signature; otherwise, output '0' for rejecting this signature.

Same as the basic digital signature, the aggregated signature is captured by *correctness* and *unforgeability*. Correctness means a valid signature generated by a legitimate secret key can pass the verification algorithm Verify and a valid aggregated signature can pass the aggregated verification algorithm aVerify;

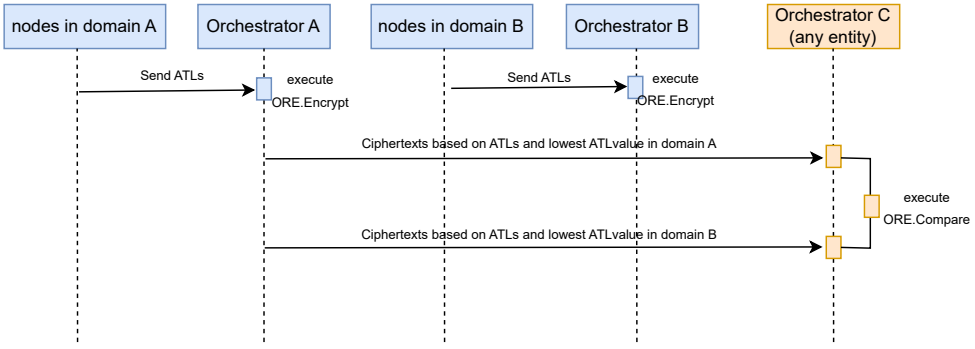

Figure 6.23: The workflow of ORE in CASTOR for different domains

Unforgeability means an adversary, who does not have a valid secret key, cannot forge an aggregated signature to pass the verification algorithm aVerify.

In the CASTOR framework, trust establishment is fundamentally achieved through a composite attestation mechanism. The pivotal challenge we address is the extension of trust evaluations from the granular level of individual network nodes to the holistic level of entire communication paths. This is accomplished by cryptographically combining the trust opinions and integrity proofs of every node along a given path into a single, verifiable claim. By adopting this sequential attestation model, CASTOR not only fulfills its functional requirement of constructing trusted paths but also rigorously enforces critical security properties. Specifically, the mechanism guarantees unforgeability, ensuring that no malicious node can falsify the collective attestation or impersonate a legitimate participant. Furthermore, it provides linkability, creating an auditable cryptographic chain of custody, which is a prerequisite for effective revocation. This enables the system to precisely identify and subsequently revoke any compromised node that attempts to undermine the path's integrity, thereby preserving the overall security of the routing infrastructure.

6.2.11 Crypto Structures & Building Blocks

Motivation of Order-Revealing Encryption. Following the same scenario in the composite attestation mechanism, during communication, there are two cases. In one domain, we need to ensure the ATL of the routers in a path is above a certain value (minimum ATL). Moreover, in cross-domain scenarios, different orchestrators need to exchange information, the order of ATLs without revealing the real values of ATLs. Based on this scenario, we need to compare different ATLs without revealing the information.

From a cryptographic aspect, we need a primitive that can compare the order of private information but without revealing the private information. To solve this issue, we can make use of Order-Revealing Encryption (ORE), which can get the order of plaintexts by comparing associated ciphertexts without revealing the plaintexts.

In 2014, Boneh *et al.* [31] proposed the first Order-Revealing Encryption (ORE) by making use of multi-input functional encryption without obfuscation. ORE is motivated by the problem of answering queries on a remote encrypted database [15, 29]. Consider a remote database holding encrypted pairs (name, salary). The data owner wishes to retrieve all records with a salary greater than t. If salaries are encrypted using an ORE then the database can sort all records on is own from lowest salary to highest. This sorting can be done even when records are inserted sequentially into the database (perhaps by multiple users who share the secret encryption key) and requires no interaction with the data owner(s). To issue the range query the data owner sends the encryption of t under the ORE key. In response, the database first uses binary search on the encrypted salaries to locate the smallest encrypted record R with a salary greater than t and then simply sends all records to the "right" of R back to the user. Thus, for a database

of n records, the database's work is $O(log\ n)$ and requires only one round of interaction with the client, as in the case of a cleartext database. Security of the ORE ensures that the database learns nothing beyond the relative ordering of records and queries.

Workflow of Order-Revealing Encryption. An ORE scheme [43] is denoted by $\Pi = (\text{ORE.Setup}, \text{ORE.Encrypt}, \text{ORE.Compare})$ defined over a well-ordered domain $\mathcal{D}$:

- ORE.Setup$(1^\lambda) \rightarrow$ sk: On input a security parameter λ, the setup algorithm ORE.Setup outputs a secret key sk.
- ORE.Encrypt$(\text{sk}, m) \rightarrow$ ct: On input the secret key sk and a message $m \in \mathcal{D}$, the encrypt algorithm ORE.Encrypt outputs a ciphertext ct.
- ORE.Compare$(\text{ct}_1, \text{ct}_2) \rightarrow b$: On input two ciphertexts ct_1, ct_2, the compare algorithm ORE.Compare outputs a bit $b \in \{0, 1\}$.

An ORE scheme is captured by *correctness*, *indistinguishability under an ordered chosen plaintext attack (IND-OCPA)*. Details about these properties will be introduced in deliverable D3.1.

Applying ORE to CASTOR. In CASTOR project, different applications require trust path routing, in which a path is with varying Actual Trust Level (ATL). As described, CASTOR project consists of both inter- and intra-domain network optimization. Especially, in intra-domain case, every orchestrator can access to all ATLs of all nodes in this domain. When realizing the path routing, every orchestrator can encrypt all ATLs, pass the lowest ATL and all encrypted ATL values to other orchestrators in different domains.As shown in Figure 6.23, we describe the workflow of an ORE scheme used in CASTOR project with taking three domains and orchestrators as an example.

1. Orchestrator A (B) generates the secret key sk_A (sk_B) by using the algorithm ORE.Setup.
2. Nodes in domain A (B) sends ATLs to the orchestrator A (B).
3. Orchestrator A (B) computes each ciphertext associated with each ATL by executing the algorithm ORE.Encrypt.
4. Orchestrator A (B) sends ciphertexts and the lowest ATL value in domain A (B) to the orchestrator C (can be any entity).
5. After receiving ciphertexts, the orchestrator C (*any entity, who can also be orchestrator A or B*) compares different ciphertexts using the algorithm ORE.Compare to get the order of ATLs.

In CASTOR, cross-domain trust evaluation necessitates a privacy-preserving method to compare Trust Levels (ATLs) across different administrative domains. To achieve this, the system employs Order-Revealing Encryption (ORE) schemes. This cryptographic primitive allows the orchestrators from different domains to determine the relative order of their encrypted ATLs, for instance, identifying which domain has the higher or lower trust level without ever decrypting them and revealing the actual numerical values. This capability is crucial for making routing decisions based on the least trusted segment of a path while maintaining the confidentiality of each domain's internal trust assessment.

From a cryptographic perspective, it is important to acknowledge that ORE schemes do not provide the same level of security as Fully Homomorphic Encryption and can leak some information beyond the order. The primary leakage is the equality pattern and the most significant bit of difference between plaintexts, revealing the first position in which two encrypted messages begin to differ. However, under state-of-the-art ORE constructions, this controlled leakage is considered acceptable for many practical applications. Crucially, any external adversary or a curious orchestrator cannot feasibly recover the entire original message (the actual ATL value) from the ciphertext alone. The security of modern ORE ensures that,

despite the partial information leakage, the plaintext data remains confidential and is not fully exposed, thereby striking a viable balance between functional utility and privacy preservation for the CASTOR cross-domain use case.

Chapter 7

CASTOR Methodology

The methodology adopted for the creation of this deliverable is presented in this chapter, beginning with the design of the CASTOR Minimum Viable Product (MVP), which serves to demonstrate the common vision of the consortium. It then continues with the elicitation of the technical (Chapter 9) and use case requirements (Chapter 8) that support the development of the core MVP features.

Within this framework, the methodology described herein has been applied to design the complete CASTOR landscape in terms of both research and specifications. The relationship between the technical and functional requirements and the various use case needs and scenarios is detailed in Chapter 8. Furthermore, the design of the CASTOR MVP has been carefully planned to incorporate the activities scheduled for the project demonstrators, as described in Chapter 10 of this deliverable. This systematic approach ensures a coherent and well-structured foundation for the subsequent stages of the CASTOR project.

7.1 Methodology for MVP Design

As a term introduced by Frank Robinson in 2001, the ”Minimum Viable Product” (MVP) pertains to an agile and lean approach for the validated planning and design of a product. An MVP constitutes a product version with an adequate number of features that can satisfy the needs of initial users, while targeting at offering input for further product improvements in the future. By providing early feedback during product development, an MVP has a lower cost compared to the implementation of a wider stack of features and the subsequent collection of feedback at the end of the implementation process of the product. In addition, an MVP can help newly established companies to discover business opportunities by experimenting on the reactions of customers while trying different business models.

The four use cases of the CASTOR project will contribute to the design of an effective MVP. Even from the initial phase of WP2, they have given input about their activities that should be described in the CASTOR methodology. The demonstrators' engagement contributed to the establishment a clear definition of the project's scope and purpose. This was achieved by delving into the actual benefits that the proposed solution would offer to the end users. The consortium engaged in a collaborative process, where the demonstrators shared their insights and the technical, research, and academic partners analysed the information. This process allowed the consortium to answer the fundamental question of “What Problem You're Solving.” Additionally, by describing the “as-is” and “to-be” scenarios for each demonstrator, the consortium identified areas where the project vision could be refined and demonstrated how the existing workflows could be enhanced with the intervention of CASTOR. Moreover, by acknowledging existing gaps, CASTOR also aims to develop additional valuable services that provide competitive advantages to end-user organisations.

7.1.1 Requirements Definition Process

After discussing the initial vision during the project's DoA preparation, the consortium worked together to refine the value propositions of CASTOR. The consortium partners agreed on the core services outlined in Section 2.1 and conducted an analysis of the State-of-the-Art for each of the identified project innovations, which is further detailed in Section 2.2. This analysis was particularly important given the research nature of CASTOR as it allowed the consortium to gain insights into the competition and understand the current state of the market. The findings from this analysis will be utilised in WP7, which focuses on Dissemination, Standardization, Exploitation, and Impact Creation. This information will drive the definition of future pathways for exploiting the project's outputs and defining the go-to-market strategy for both the framework itself and its individual assets.

Moving forward with the MVP definition, the consortium proceeded to identify the stakeholders within the supply chain context. They also determined the specific actors and entities involved in the envisioned use cases, along with their respective goals (Section 7). This exercise aimed to gain a deeper understanding of the intended beneficiaries for whom the consortium is developing the solution. To achieve this goal, it is necessary to gather and prioritise requirements. Within the context of CASTOR, requirements are gathered in two ways. Firstly, technical requirements are derived from the technical partners within the consortium who are responsible for designing and developing the overall solution. Secondly, requirements are defined based on the use cases or user stories provided by the project's demonstrators. Once all the requirements have been collected, the partner in charge of requirements elicitation, with the consensus of the entire consortium, will prioritise them.

As the project progresses, an essential component of the MVP definition is the implementation of the "Build, Test, and Learn" cycle. This cycle is integrated into the development of the core technical Work Packages 3, 4, and 5, which involve the creation of both early and advanced versions of all project modules, including the CASTOR framework itself. Through an iterative process, all technical partners will develop their respective modules and conduct testing at early and mature trial sites (demonstrators). This approach aims to maximise the value added both to each module and the whole CASTOR framework.

The MVP definition process outlined in this section aims to provide a specific and comprehensive depiction of the CASTOR platform. This includes a detailed examination and analysis of the technical, functional, and security requirements of the various architectural components involved.

At its core, the MVP aligns with the overall vision of the consortium and the adopted product development process, taking into consideration the expectations of the users. Its purpose is to provide tangible value, validate methodological ideas and hypotheses, and serve as a guide for design and development activities throughout the project implementation. The CASTOR MVP is expected to play a crucial role in directing and shaping the ongoing work within the project.

7.2 Requirements Elicitation Methodology

A requirement is a statement which translates or expresses a need and its associated constraints and conditions with the purpose to transform through their analysis the stakeholder, requirement-driven view of desired services into a technical view of a required product that could deliver those services.

The process of eliciting requirements serves as a significant foundation for developing the business value of the CASTOR framework. Since these requirements form the basis for addressing identified needs, it is vital that they are comprehensive, transparent, and accurate. To ensure successful elicitation, it is essential for the individual or organisation responsible to actively involve all relevant stakeholders and engage them in this process. The elicitation of requirements is typically not a one-time event. For example, during the elaboration phase of a project, requirements may be collected through interviews or requirements workshops. As these requirements are utilised to define and validate models or products,

gaps in the requirements may become apparent, necessitating the elicitation of additional details for those newly identified requirements. In the next subsections, an overview of the methods employed within the CASTOR project for defining stakeholder requirements is provided.

7.2.1 Technical Requirements Specification Process

The CASTOR project has employed an agile methodology, characterised by its iterative nature. This methodology emphasises clear communication and understanding between the business, technical, and scientific aspects of the project. It establishes transparent expectations at the project's commencement and at each milestone, fostering collaboration and progress.

The CASTOR approach to system requirement specification initiates with individual interviews conducted with the consortium technical partners. These interviews were carried out using online tools, resulting in the collection of raw requirements. Raw requirements refer to requirements that have not undergone analysis or been formally documented in a well-structured requirement notation. The collected raw requirements are then subjected to an iterative internal process led by the CASTOR system architects to yield more refined results.

During this phase, brainstorming techniques are employed through ad-hoc calls to further enhance the requirements. The outcomes of these brainstorming sessions are subsequently compared with the technical aspects of the system and linked to one or more of the specified value propositions of the project. This iterative process ultimately leads to the identification and formulation of the necessary technical requirements, which are detailed in Chapter 9.

Nevertheless, it is important to acknowledge that requirements derived from interviews can occasionally be unclear. This ambiguity may arise due to potential misinterpretation of stakeholder needs by the system architects, or when stakeholders struggle to comprehend certain questions or lack technical expertise to provide accurate answers. As a result, alongside the process of requirements collection, the CASTOR academic partners have conducted a comprehensive document analysis. This analysis takes the form of a State-of-the-Art review, including an extensive list of industry practices and relevant literature. The purpose of this analysis is twofold: to validate the specified requirements and to identify applicable standards and constraints. By conducting this analysis, the aim is to enhance the quality of the requirements generated within the project.

The process of requirements collection underwent continuous review and examination by the WP2 focus group. The WP2 focus group conducted dedicated teleconferences, involving specific partners or the whole CASTOR consortium, to discuss and assess the gathered requirements. Additionally, one of their responsibilities was to offer valuable feedback on the interview participants. The success of eliciting user requirements relies significantly on the knowledge and experience of the stakeholders, making the input from the WP2 focus group crucial in ensuring the efficacy and accuracy of the requirements gathering process.

7.2.2 Use Case Requirements Specification Process

In addition to the overall input on technical requirements, the use case partners provided a detailed description of their user stories and a technical explanation of how they intended to utilize the CASTOR framework. They also specified the core functionalities they planned to leverage. This allowed for a more precise alignment of requirements with the demonstrators, providing opportunities for further research and enabling the demonstrators to explore multiple implementations, if time and resources permit.

The use case partners collaborated closely with the research partners to refine the requirements and elaborate on the technical details derived from the narratives. This followed an approach commonly observed in agile projects. The refinement process involved translating the narratives into specific user

stories. User stories are concise units of ideas used to provide high-level descriptions of requirements. They depict a particular feature from the perspective of an end user, typically a system user or customer. User stories are designed to be easily understandable and expressible by non-technical partners. Despite being short, typically consisting of a single sentence, user stories possess the unique ability to be self-explanatory and contain sufficient information to describe the requirement. This allows developers to provide a reasonable estimate of the effort required for implementation.

For each of the three distinct CASTOR Use Cases, the demonstrator partners were initially requested to provide user stories that describe the ”to-be” reference scenario. To elicit requirements from these user stories, the perspective that a user story serves as a well-defined requirement is embraced, in the sense that:

- Emphasises the perspective of a specific role that will utilise or be affected by the solution.
- Defines the requirement using language that is meaningful to that role.
- Clarifies the reasoning behind the requirement.
- Facilitates the definition of high-level requirements without delving into detailed specifics very early.
- Takes into account the user’s goals and the business value associated with each requirement.

These aspects are captured using a straightforward textual template to create a comprehensive sentence. While various templates are available, the Connextra template is employed by 70% of practitioners [49].

User Story Title
As a <*type of user*>, I want to <*some goal or action*> so that <*some reason or benefit*>

In an agile project, new or updated user stories may arise at any stage of implementation, leading to changes in the backlog. This behaviour is highly desirable as it ensures a continual focus on aspects that are meaningful to users, while potentially excluding features that may have less importance in terms of the value they offer to both the system and its surrounding environment.

To ensure that the user stories developed by the project’s demonstrators in this step meet the necessary quality and criteria, a user story validation process is incorporated into the methodology. The objective of validating each user story is to assess the extent to which it fulfils the INVEST characteristics. The acronym INVEST serves as a mnemonic for a well-established set of criteria or checklist used to evaluate the quality of a user story. If a user story fails to satisfy any of these criteria, the team may consider rephrasing it or even rewriting it. This technique has been recommended by Mike Cohn [49].

An optimal user story, based on INVEST characteristics, should be:

- Independent: The user story should possess self-contained characteristics, meaning that it does not rely on another user story for its completion or functionality.
- Negotiable: User stories are subject to change and rewriting until they become part of an iteration.
- Valuable: A user story should provide value to the end user.
- Estimable: The size of a user story should always be able to be estimated by the team.
- Small: User stories should be sized appropriately to ensure they can be effectively planned, tasked and prioritised with a reasonable level of certainty.
- Testable: The user story or its related description should include the necessary information to facilitate the development of tests.

The user stories described for each of the three CASTOR Use Cases are presented in Section 7 as part of the corresponding “to-be” reference scenarios.

Chapter 8

CASTOR Use Cases

8.1 High-Level Introduction of the CASTOR Use Cases Towards Trusted Traffic Engineering process

The overarching architecture of CASTOR spans across the Compute Continuum, allowing stakeholders to ensure secure data transmission over trusted paths. The evaluation and validation of the CASTOR framework and inner-workings requires a careful selection of realistic environments with diverse characteristics. To this end, CASTOR introduces four diverse and carefully selected use cases stemming from time-constrained and demanding application domains (primarily in the context of Automotive & Aerospace safety-critical domains). This allows the incorporation of ensuring path provisioning for services with varying path profile requirements (e.g., different network- and trust-related objectives). Through such application services of mixed criticality, the endmost goal is to be able to evaluate the **performance footprint of the CASTOR process into the "vanilla" (segment and source) routing protocol stack against a diverse conditions so as to have a wide coverage on the overhead that each technology strand (see System Model in Chapter 3) may impose:** first of all, into the overarching inter- and intra-domain forwarding and subsequently on the applications' operational profile.

The second driving factor in the use case specification lies in the definition of the overarching Key Performance Indicators (KPIs) for each evaluation/experimentation scenario. One strategic approach towards the KPI specification is to decouple the application operations from the (multi-) path establishment and control at the routing plane. However, *we have to note that at this early stage of the project, there cannot be a definitive answer on whether CASTOR may hedge wide applicability to real-world deployments due to significant boundaries that may be posed to the operational profile of the employing applications.* Recall that one of the core design principles is transparency and seamless (to-the-service) establishment and maintenance of the required SSLAs (during runtime) which if not manifested can significantly affect the adoption of such dynamic solutions to their existing static state - either during boot-up [27] or based on predefined paths [45]. This calls for a detailed benchmarking and critical evaluation on all aspects of the CASTOR CC-wide framework. Consequently, besides the well-defined and regulated V2X application use cases where there are strict KPI requirements so as to not compromise the safety characteristics of the deployed services, we aim to capture the CASTOR overhead through delta measurements in order to ensure that the overall performance of the application traffic does not exceed acceptable margins.

In what follows, we delve into the details of each one of the four envisioned use cases. Starting from a short summary of the key characteristics and the focal points of each use case (Tables 8.1 - 8.4), we present the current practices of traffic engineering in each of the pilot domains and critically reflect on the needs that CASTOR aims to address. This culminates in the specification of concrete scenario compounds (defined in the form of "*user stories*") associated with KPIs that will guide the evaluation of each one of the CASTOR functional components as detailed in Table 6.1. As all four use cases focus on

the performance overhead that CASTOR introduces in the forwarding plane, a separate Proof-of-Concept is presented in Section 8.6, reflecting on the validation of the CASTOR capabilities at the orchestration layer and the control service responsible for the path exploration process (including all properties of path construction, registration and enforcement). As we dive deeper into technical discussions and approach the first integrated release of the CASTOR framework, the goal is to review this initial set of KPIs, so as to incorporate the challenges and additional overhead posed by inter-domain service provisioning.

Table 8.1: UC1 - Highly Available & Secure Airspace Monitoring in Urban Air Mobility (UAM) Environments

UC1 - Highly Available & Secure Airspace Monitoring in Urban Air Mobility (UAM) Environments
Urban Air Mobility environments depend on a continuous flow of accurate surveillance information to maintain awareness of airspace state and activity. Such environments are highly dynamic and operators must trust that their surveillance and reporting feeds are not only timely but also protected from misrouting and tampering. Traditional network monitoring and adaptation strategies offer little insight into the real-time trustworthiness of the network path over which surveillance data is shared. We must assume that successfully established network paths are trustworthy and remain so. Such assumptions can result in critical surveillance lapses in the event that a network intrusion remains undetected or unaddressed while manual intervention and troubleshooting procedures are carried out. This use case explores how CASTOR can help airspace monitoring applications adapt their behaviour in real time. **Specifically, it aims to employ CASTOR in order to engrain link-level trust guarantees as part of the establishment of end-to-end service connectivity.** In contrast to the rest of the use cases in CASTOR where they consdider accumulated path-level trust guarantees, these scenarios aim to shed light on the robustness and flexibility of the CASTOR framework to evaluate the trust posture of the infrastructure topology, recommend and enforce accurate traffic engineering policies. When the network path is trustworthy, highly sensitive information such as radar health and sensor capabilities can be shared confidently with other actors. When CASTOR detects and reports that the path no longer satisfies the integrity or confidentiality expectations of the surveillance source, the source can automatically respond and dial down the sharing of highly sensitive data while maintaining the distribution of less sensitive (but nevertheless safety critical) airspace activity observations. **To further evaluate the agility of the trust awareness and the multi-path control offered by CASTOR, the presented scenarios consider the dynamic transition to fallback pre-established SSLAs, when any of the primary SSLA objectives gets compromised.** We seek to evaluate how CASTOR can instrument, monitor and adapt network paths automatically in the face of sample intrusions, compromises, and failures of the underlying path infrastructure. We seek to explore this within the context of a single network domain in which we monitor a highly sensitive area such as an airport and also to push beyond into network paths stretching across multiple domains mirroring the real world deployment scenarios of centralized surveillance collection and observation points that serve as synchronization points for the distributed surveillance ecosystem. To keep the evaluation aligned with our lab conditions rather than production-grade networks, we intentionally avoid asserting industry SLA targets such as absolute latency, jitter, or loss. Instead, we first benchmark a baseline profile of our lab network and then express all CASTOR results as Δ-KPIs that capture incremental overheads and improvements relative to that baseline. Examples include Δlatency, Δloss, ΔCPU, and tier flip propagation time. This approach keeps outcomes reproducible in our environment while still indicating their plausibility for production deployments.

Table 8.2: UC2 - Trustworthy Communications of First Responder Mobile Units and the Compute Continuum

UC2 - Trustworthy Communications of First Responder Mobile Units and the Compute Continuum
This use case focuses on maintaining secure operational readiness for first-responder OBUs, which rely on timely PKI certificate provisioning and secure OTA firmware updates. These devices operate in highly dynamic, safety-critical environments where network conditions are unpredictable and backend connectivity may involve multiple administrative domains. Since first-responder vehicles often move across regions covered by different operators or jurisdictions, their security services—such as certificate renewal from external PKIs—must function seamlessly across domains. Cross-domain operation is essential because the PKI or OTA backend may not reside within the same network domain as the responder's current connectivity. Without proper trust exposure and policy alignment between domains, these security procedures would be vulnerable to misconfigured, untrusted, or underperforming network segments. Within the well-regulated V2X communication fabric, **CASTOR is the first of its kind to evaluate the incorporation of trust objectives towards the end-to-end V2X service provisioning**: from the far-edge OBU elements, the V2X Application Server, OTA servers, and external PKI infrastructures. Even though there are existing projects that have yielded significant results towards trust assurances at an application service level (e.g., CONNECT [135]), CASTOR introduces such concepts at the routing plane. Through continuous SSLA compliance checks, trust semantics exchange, and dynamic path validation, CASTOR ensures that certificate downloads and firmware updates remain secure, consistent, and resilient even as vehicles roam, network integrity fluctuates, or cross-domain routing is required.

Table 8.3: UC3 - Priority-based Trusted Messaging & Scalable Performance for CCAM Applications

UC3 - Priority-based Trusted Messaging & Scalable Performance for CCAM Applications
In the highly interconnected and complex environments envisioned in Cooperative, Connected and Automated Mobility (CCAM), it is essential to have a framework capable of providing a trust assessment not only on the veracity of data collected from various sources, but also on the reliability, integrity, and end-to-end guarantees of the communication channels that enable seamless service connectivity among CCAM stakeholders across multiple domains. CCAM services rely on the timely and trustworthy exchange of information such as crash alerts, road hazards, traffic updates, and emergency events. These messages (e.g., CAM and DENM) must be delivered with strict guarantees on latency, reliability, integrity, and confidentiality to enable safe and efficient response actions. However, today's communication paths traverse heterogeneous networks and operators, and are largely based on best-effort performance. This means that delays, degraded throughput, or unverified communication routes may directly affect safety-critical operations. The project uses three representative user stories to explore whether critical ITS (Intelligent Transportation System) services can be reliably improved by CASTOR framework, by enabling trustworthy communication paths and service delivery. These stories are: (1) delivering real-time traffic information from road events, (2) supporting emergency responders who depend on timely and trustworthy alerts, and (3) notifying connected vehicles about nearby vulnerable road users such as pedestrians or cyclists. Each case represents a safety-critical function where delays, loss of integrity, or incomplete data could lead to poor decisions or dangerous outcomes. Building on top of the existing CASTOR overhead considerations, **this use case goes beyond the single-domain scenarios and evaluates how CASTOR affects the safety profile of the aforementioned functions in cross-domain scenarios.** Eventually, the goal is to understand what type of trust mechanisms, orchestration logic, or network guarantees may be needed, and to use the user stories as tangible benchmarks that guide the exploration.

Table 8.4: UC4 - Future-Proofing Next-Generation Unmanned Aerial Vehicles Communications towards Critical Infrastructure Sustainability

UC4 - Future-Proofing Next-Generation Unmanned Aerial Vehicles Communications towards Critical Infrastructure Sustainability
Data flows of UAVs in modern 5G systems traverse the internet through a network where performance is guaranteed using QoS mechanisms but there is no assurance of trust. With traffic exiting the 5G network towards the data plane, there is no visibility if the routers forwarding the traffic are secure, compromised, or degraded. Blind spots emerge in this situation where sensitive information such as mission telemetry and inspection data are routed through nodes that ensure performance but are not trustworthy. The goal of this use case is to introduce continuous trust assessment and dynamic path selection to address the blind spots. CASTOR fills that role by performing a continuous, evidence-based trust attestation to the routers and the network paths of the system, using a trust scores, computed and applied to routes to select the optimal path and rerouting to routes with better trust score when necessary. The scenarios will be used to validate this use case. In the first scenario CASTOR will be used to ensure that the UAV mission inspection data that exit the 5G network will reach the destination though low-latency, tamper-evident, and trustworthy paths, eliminating the risk of compromised routers degrading confidentiality or integrity. The second scenario, will integrate a risk index that is provided by the MNO of the 5G network that will be taken into account during the trust assessment, demonstrating how external risks can directly influence trust-guided routing. The third one is an exploration scenario that focuses on the core of the 5G network and not in the data plane, on a distributed UPF deployment ensuring inter-UPF communication is trustworthy and remains performant, with user plane data remaining encrypted and opaque. In this context, **the use case aims assess CASTOR's capacity to support trust-aware inter-domain connectivity for 5G verticals, taking into account the operational and trust-related constraints of shared backhaul domains.** Since the scenarios of Use Case 4 will be evaluated in a lab environment, first, the baseline values (such as latency, packet loss etc) will be established through the nominal operation user stories, taking into consideration potential biases introduced by the emulation environment of the lab (e.g., delays introduced through the virtualisation layer of the emulations). Then, the CASTOR user stories will benchmark the overhead that CASTOR adds against the reference values of the nominal user stories.

8.2 Highly Available & Secure Airspace Monitoring in Urban Air Mobility (UAM) Environments

Before introducing this use case, it may be beneficial to begin with a glossary of terms as much of the terminology and acronyms will be unfamiliar to readers not averse with this domain.

Table 8.5: Short Glossary of Terms from Airspace Management

Term	Description
ADS-B	**Automatic Dependent Surveillance–Broadcast**. Cooperative surveillance technology in which aircraft broadcast their position and velocity; ground receivers ingest these messages.
CISP	**Common Information Service Provider**. Broker that aggregates and disseminates shared airspace information—such as restrictions, weather, and surveillance summaries—to authorised consumers.
COP	**Common Operating Picture**. A shared, near-real-time fused view of the airspace built from multiple data sources.
Edge/on-prem cache (USSP/CISP edge)	Local deployment of cloud services at the site to reduce latency and maintain operation during WAN outages.

Term	Description
Electric Vertical TakeOff and Landing (eVTOL) aircraft	An electrically powered aircraft capable of taking off, hovering, and landing vertically without a runway, typically used for urban air mobility operations.
GCS	**Ground Control Station**. Operator workstation that commands the UAV and receives telemetry data.
LAN / Campus network	Operator-managed local network domain interconnecting sensors, GCS, and edge services.
UAM	**Urban Air Mobility**. Low-altitude movement of people or goods within cities using UAVs or electric Vertical TakeOff and Landing (eVTOL) vehicles over dense, dynamic routes.
UAV	**Unmanned Aerial Vehicle**. Aircraft operated without an onboard pilot; commonly referred to as a drone.
U-space / UTM	**Unmanned Traffic Management**. Digital infrastructure and rules enabling safe, scalable UAV operations—flight planning, conformance monitoring, and strategic deconfliction.
USSP	**U-Space Service Supplier**. Provider that validates flight plans, monitors conformance, and mediates operator access to U-space services.
Vertiport	**U**rban take-off and landing site for UAVs or eVTOLs equipped with associated ground infrastructure.

Unmanned airspace is going through explosive growth as UAVs take on increasing responsibilities from package delivery to media services, from taxis to inspection and emergency response [120]. Traffic density is rising just as steeply: satisfying urban demand could mean tens of thousands of low-altitude flights per hour over a single metropolis - Paris is often cited at 87,000 flights/h in mature scenarios [55]. To keep pace, U-space/Unmanned Traffic Management (UTM) architectures increasingly shuttle flight plans, surveillance records, and separation commands (to keep the airspace orderly and avoid collisions between occupants) across public cloud and internet links. As a result, a growing patchwork of UAV operators, ground sensors, USSPs, and CISPs are now competing for bandwidth and trust on shared networks in order to share real-time situational data (airspace occupancy, obstacles, wind patterns) and maintain a combined view of airspace for all stakeholders in the ecosystem. The core problem: we cannot tell if the routers that carry this traffic constantly remain trustworthy.

Key domains in this complex, multi-domain connectivity landscape:

Ground Infrastructure on Private LANs At each distribution hub or vertiport, Ground Control Stations (GCS) and ground sensor arrays connect over isolated, operator-managed LANs. This domain guarantees ultra-low, deterministic latencies ($\leq$ 100 ms), fine-grained QoS, zero-trust enforcement, and local auditability.

USSP & CISP on Commercial Public Cloud USSPs and CISPs run in the public cloud to leverage elastic scalability for peak delivery periods, built-in global redundancy, and advanced multi-tenant security controls. CISPs aggregate telemetry from distributed ground sensors (e.g., Collins Aerospace Skyler™ radars) to furnish real-time airspace maps and obstacle alerts.

8.2.1 System Model, Communication Interfaces, and Protocols

The domain of Unmanned Aerial Vehicle (UAV) surveillance is undergoing iterative refinement globally, with diverse deployment models emerging across continents and countries under the guidance of various regulatory and operational bodies. In CASTOR, we adopt a deployment topology based on the European U-Space model [61] in which U-Space Service Providers manage airspace zones and access common airspace information through a CISP.

8.2.1.1 System Components

In Figure 8.1, we present a deployment view with two airspace zones being monitored and managed by two sites. Each site connects to a USSP running on the public cloud. The key system components are

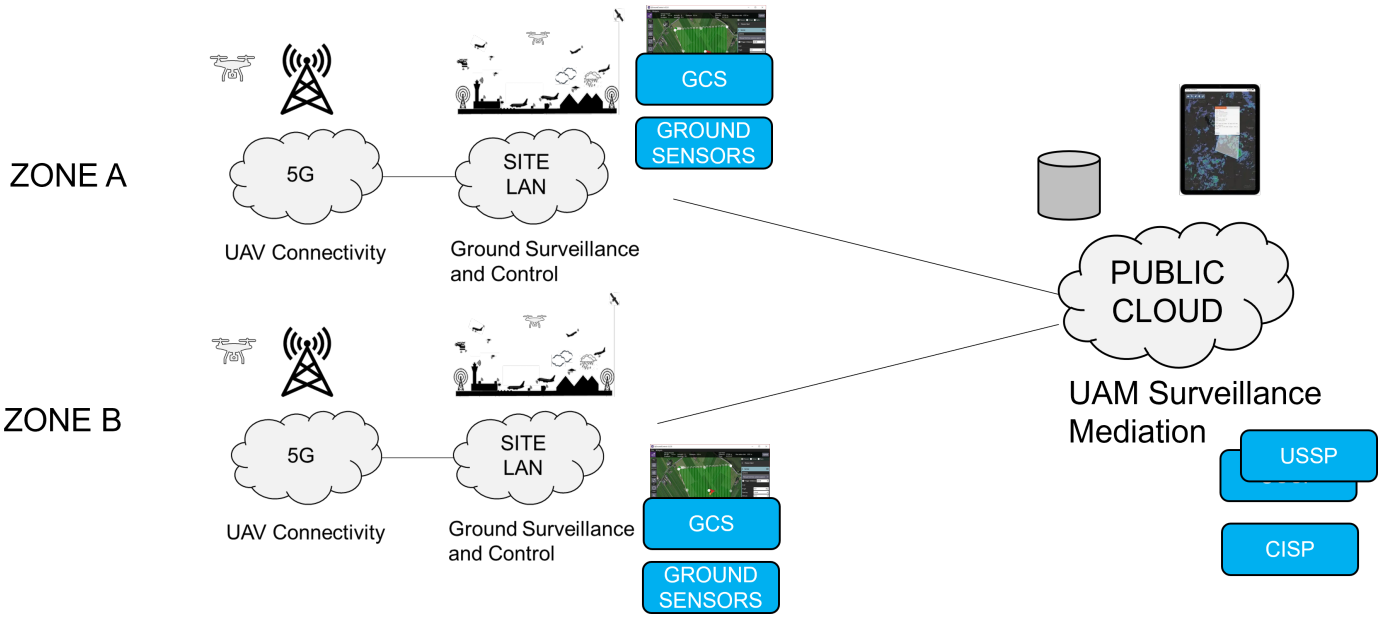

Figure 8.1: Deployment of Key System Components

as follows:

Mobile Network Our use case envisions UAVs using cellular networks (private or public) for connectivity to UAV Ground Control Stations.

- **UAV.** Unmanned Aerial Vehicles such as drones are tethered to UAV Ground Control Stations through 5G links. UAVs receive flight plans before launch which lay out a succession of waypoints that the UAV must adhere to when travelling from launch to destination sites.

Zone Local Area Network Each operational airspace zone has an area of coverage which is populated by UAVs with corresponding Ground Control Stations and surveillance equipment such as radars. A local LAN deployed and maintained by the zone operator offers a connectivity backbone to local components

- **GCS.** UAVs are managed by operators through a Ground Control Station (GCS) that is used to instruct UAVs on their movements and receive continuous telemetry updates from the UAV as it travels.
- **Ground Sensors.** Nodes such as radars and ADS-B receivers form a terrestrial surveillance infrastructure that maintains a continuously updated view of the airspace overhead. Such data is sent to the USSP for analysis.

Public Cloud For reasons of scalability and availability, USSPs and CISPs are deployed on public cloud infrastructure with connectivity from the Zone LAN conducted through a VPN.

- **USSP.** U-Space Service Providers interface with UAV operators through their GCS and act as their mediator with the rest of the airspace ecosystem – from confirming flight plan validity to publishing and consuming data from the CISP. The USSP ensures that operators adhere to the operational and safety requirements of U-Space.
- **CISP.** Acts as the single source of truth for static and dynamic data (e.g., airspace restrictions, surveillance data, weather) required for U-space operations. It collects, processes, and disseminates data from multiple sources, such as air traffic control systems, ground sensors, weather data and predictions. The CISP is essentially an airspace surveillance broker that pools common airspace information and brokers an accurate and detailed view of the airspace.

It may exchange a high-level view of the unmanned airspace with a central SWIM node from which it is distributed to interested commercial airspace operators. It additionally consumes information from the SWIM node relevant to U-Space operations such as weather alerts and airspace restrictions.

8.2.1.2 Intra and inter-domain communication interfaces

As we have seen, the operation and management of an unmanned airspace zone involves multiple network domains. We begin with the relatively straightforward task of provisioning a UAV flight plan. Before the plan can be deployed to the UAV, it must be validated. The GCS, deployed in the local zone network, is used to draft and submit the plan to the USSP on the cloud for validation. This communication from the GCS (Ground Infrastructure/LAN Domain) to the USSP (Public Cloud Domain) requires inter-domain communication. This path typically traverses numerous intervening networks, including multiple Internet Service Providers (ISPs), which is why it often uses a VPN for security. The USSP applies its knowledge by consuming relevant data from the CISP, such as no-fly zones, weather conditions, and information about other USSPs' approved flight path reservations, to ensure no conflicts exist. If all checks pass and the USSP approves the flight, the GCS will then provision the flight plan to the UAV, which is typically connected via a cellular connection (see Figure 8.2) .

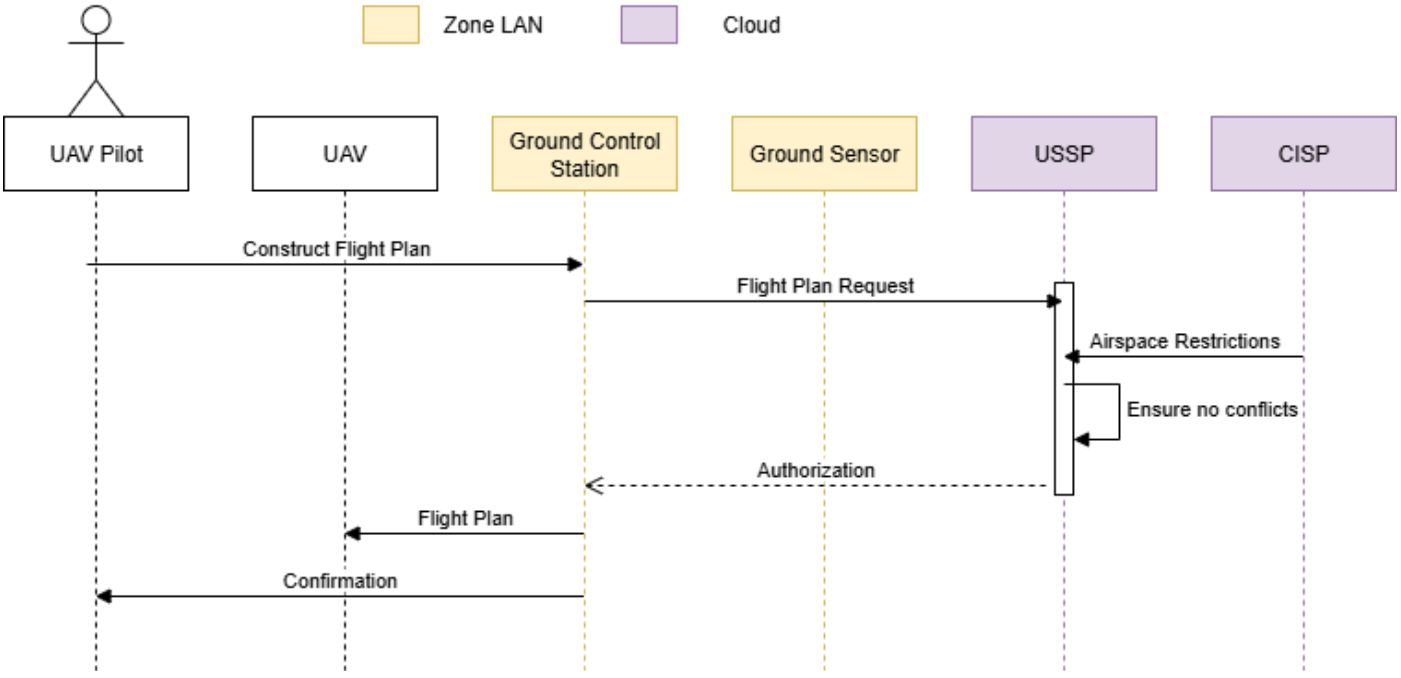

Figure 8.2: Flight Plan Registration Information Flow

An active airspace is under continuous management and surveillance. Telemetry from UAVs in flight is continuously streamed to the GCS and position updates are being streamed to the USSP which ensures that the aircraft is operating within agreed flight volume boundaries. Local surveillance data from a ground radar are streamed to the CISP from where it is consumed by the USSP as shown in Figure 8.3. This entails a key inter-domain data transfer from the LAN to the Public Cloud domain.

Airspace is comprised of a collection of independent multi-operator airspace zones and neighbouring zones must share a Common Operating Picture of the airspace to improve awareness – to alert zone operators of dynamic surveillance data and airspace restrictions that could warrant consideration and action in the management of their own zones. CISP nodes are responsible for coordinating the sharing of common information amongst the community of airspace zone USSPs. Each USSP is responsible for

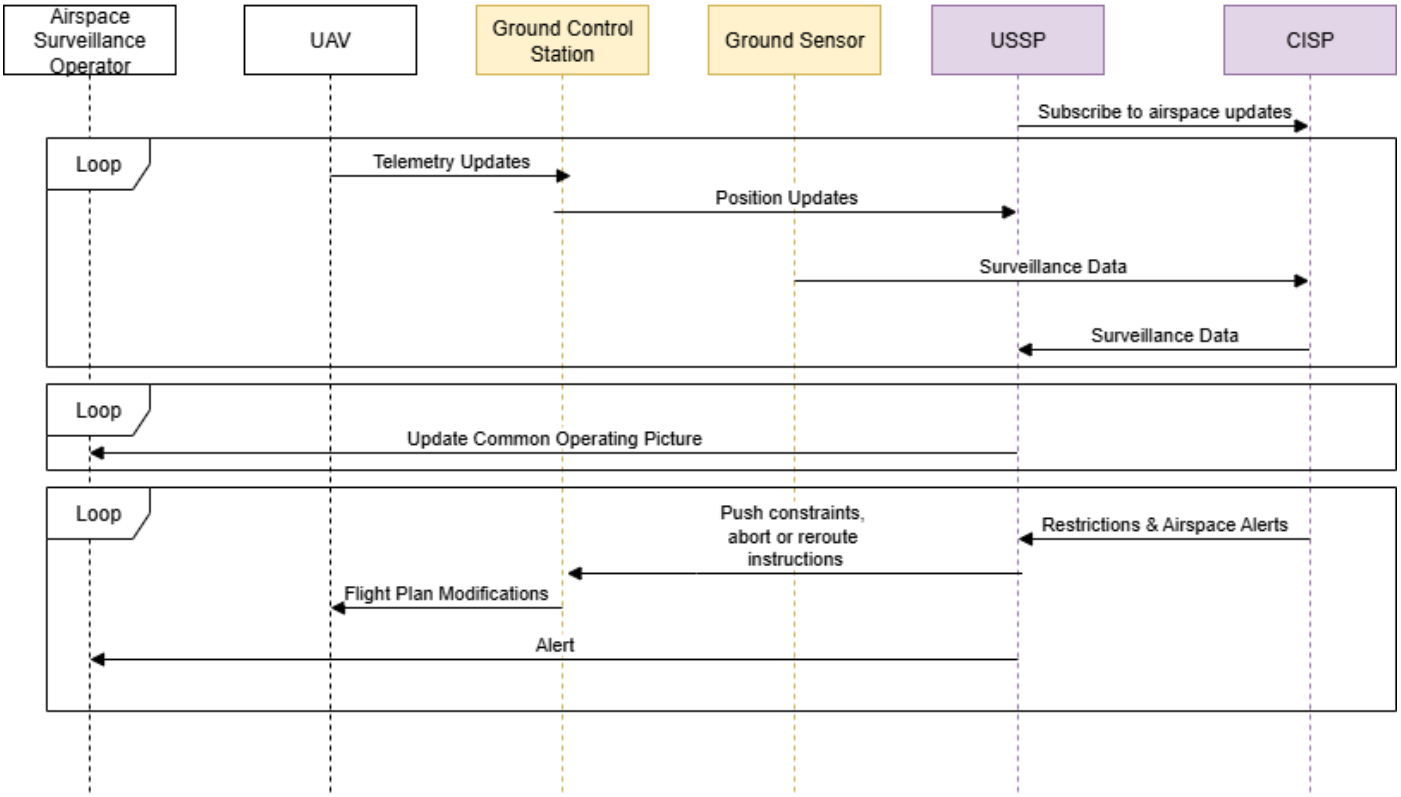

Figure 8.3: Unmanned Airspace Surveillance Information Flow

the management of a single airspace zone. It subscribes to be notified of the publication of relevant data from surveillance sensors and national authorities

8.2.2 "As-is" Scenario

In today's UAV ecosystem, every piece of operational data, whether it's periodic telemetry uplinks from GCS to USSPs, real-time radar/ADS-B surveillance observations streamed to the CISP, or dynamic weather and NOTAM updates pulled into the CISP, traverses a chain of routers whose integrity is simply assumed rather than actively verified. A typical flight begins when a GCS dispatches its routine telemetry report (position, speed, battery health) over a VPN to the USSP; in parallel, authorised sensor gateways publish surveillance observations directly to the CISP for downstream consumers, while the USSP consumes common-information feeds from the CISP. At each handoff, from routers in the public cloud, to routers in private carrier backbones, to edge routers on private LANs of urban UAM hubs—routing decisions are driven exclusively by QoS queues, ACLs and encryption tunnels (some airports already pilot on-prem cache nodes for failsafe operation, but the primary USSP/CISP remains in the public cloud). Typically lacking in this chain is attestation or monitoring of the software image, firmware version, or runtime behaviour of the routers themselves. Because these routers are never dynamically verified, a stealthy compromise (e.g., a backdoored firmware update on a router) can continue in the critical path of data flowing along the path without raising alarms. The network reacts only to link failures or manual reconfiguration, so a degraded router can remain in service for an extended period of time (possibly even days or weeks), silently diverting or delaying critical UAV or sensor traffic. Worse still, when common-information feeds are disseminated via the CISP after traversing unverified networks, downstream consumers ingest them with no visibility of the trust posture of the path they took through private or public routers. A data-fusion platform combining telemetry, radar, and weather feeds has no way to know that one feed skirted an untrusted cloud router and therefore cannot assign lower confidence to that source. This blind trust forces everyone to treat every stream as equally reliable or, conservatively, to discount all sources whenever any single router's security is in doubt - sacrificing both responsiveness and accuracy in real-time

decision support.

On the private LAN at each hub, the same trust blind spot recurs. If an edge router's encryption layer or firmware has drifted from its approved state, downstream systems cannot detect that loss of integrity; they continue to forward or accept traffic purely on QoS settings. Consequently, stale or tampered packets may enter sensor-fusion chains unnoticed, and operators must rely on manual checks or periodic audits to discover the fault - often long after it has already biased their surveillance picture. Across domains from backbone to cloud to LAN, each organization conducts periodic, manual audits of its own routers, but there is no cross-domain, real-time attestation framework. Adjacent zones consume and contribute sensor/common-information feeds via the CISP without any chain-of-trust metadata, leading to inconsistent confidence levels and delayed emergency responses. In practice, today's "As-Is" architecture delivers on throughput and uptime SLAs but remains fundamentally reactive, inflexible in handling mixed-confidentiality sensor data, and entirely blind to the actual integrity of the routers carrying our most critical UAV and ground-sensor traffic. The next section summarises how CASTOR closes this gap.

8.2.3 Collins Use Case needs from CASTOR

The promise of UAVs hinges on uninterrupted streams of telemetry, surveillance feeds and sensor health diagnostics flowing through an end-to-end network of routers - from private-LAN through public-cloud platforms. Today's routing fabric delivers on throughput and latency SLAs but makes no distinction between a fully trusted hardware router in the backbone and an unvetted virtual router in a cloud VPN; nor does it convey any trust metadata alongside published data feeds. As a result, a downstream consumer - whether a CISP aggregating multi-sensor fusions or a logistics operator correlating telemetry from ten different UAVs - has zero visibility into the integrity of the network path that carried each packet. They cannot weight or filter incoming streams based on path trust and must either treat everything as equally reliable or conservatively disregard data whenever any single hop's security posture is uncertain. Local edge nodes (USSP and CISP) deployed in hubs such as airports [1] eliminate Internet latency, but they still rely on local routers whose integrity must be continuously verified.

To bridge this critical gap, CASTOR must deliver three core capabilities:

1. **Router-Centric Trust Attestation & Scoring** CASTOR continuously measures firmware, configuration, and runtime state on every router to maintain an up-to-date Actual Trust Level (ATL).
2. **Dynamic Trust-Aware Traffic Steering** When a path's ATL drops below the Required Trust Level (RTL), CASTOR automatically moves the flow to the nearest compliant route.
3. **Tiered Security Service-Level Agreements (SSLAs)** Publishers can declare primary and fallback SSLAs; CASTOR exposes which tier is active so applications can adapt payload confidentiality.

By exposing **which SSLA** is in effect rather than raw ATL numbers - CASTOR keeps the application in the loop without leaking detailed trust scores, enabling sensors and other publishers to adapt their payload confidentiality dynamically while preserving overall data continuity.

These core functions of continuous router integrity monitoring and automated rerouting when trust thresholds are not met empower our UAV ecosystem to maintain high throughput and low latency, guaranteeing that every critical packet travels only over provably trustworthy paths - even when individual routers degrade or fail.

[1] This is for practical reasons such as keeping the safety-critical loop inside the airport LAN and removing cloud dependency from the critical operations path. It is also for security reasons as there is often high resolution and privacy-sensitive sensor data that is not shared outside of the airport.

UC1 Network/Trust Properties of Interest	
Name	**Description**
Integrity	We need to have confidence that surveillance data has not been manipulated enroute as this can have direct repercussions on how airspace is kept safe
Performance	Surveillance data has a short shelf life. It is gathered from multiple sources and amalgamated to build a realtime dynamic view of the airspace. Late data cannot contribute and result in incomplete or even misleading views of the airspace
Observability/Auditability	Operators must see SSLA compliance, ATL changes, and reroute events for accountability and forensics.
Availability/Resilience	Drops/outages blind operators.
Confidentiality	Knowing the current trustworthiness of the network path enables the restriction of confidential data streams such as radar health and capabilities to times that we are assured the network path has high link level integrity
Provenance	When consuming data from a shared CISP, consumers need to see the originating path's trust tier to weight/triage data (e.g., Trust Indicator TI=High/Low) where a Trust Indicator is derived from the current trustworthiness of the network path (as assessed by CASTOR) within the context of the SSLAs associated with the originating application.

Table 8.6: Network and Trust properties as service-level objectives

8.2.4 To-be Reference Scenario 1: On-Airport Trusted-Routing Loop for Real-Time Surveillance

A hybrid edge/cloud pattern is increasingly recommended for high-consequence hubs: cloud scale for regional coordination, on-site cache for sub-100 ms control. The objective is to ensure that every surveillance packet carried inside a single airport LAN reaches the local UTM platform and ATC screens over a path whose routers satisfy the airport's Secure-SLA. CASTOR must continuously attest to each router, calculate the path ATL, and reroute within milliseconds whenever the path falls below the RTL.

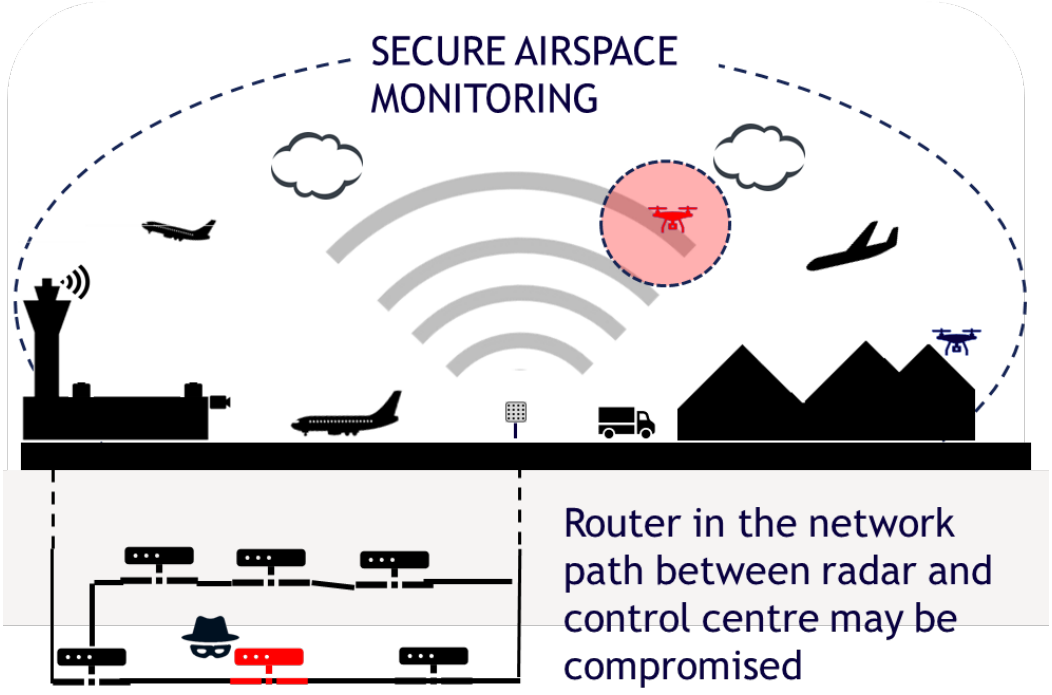

Figure 8.4: Identification of airspace threat potentially compromised

In an airspace monitoring environment, we depend on secure and efficient network communications to deliver real-time situational awareness. Currently, these networks focus on QoS-based traffic prioritization without considering trust in the selection of network paths. This leads to trust blind spots in that efficient but compromised routers can occupy key data transit points. Positioning untrusted routers in the transmission paths of critical data leads to the risk of data being delayed and diverted to thwart critical decision making. When router compromises are detected (through end user alerts or routine checks), it is typically after transmission has been corrupted – corruption that may have been occurring over an extended period of time. Surveillance data is comprised of multiple independent data streams. For assurance purposes, we build in some data redundancy - we strive to have multiple sources of data to assess and fuse together so we are not at the mercy of a single component of the monitoring infrastructure.

Components can misbehave through hardware or software faults, they can break, and they can possess different weaknesses across the surveillance spectrum. These factors are taken into account during the construction and maintenance of highly dynamic volumetric surveillance views. Currently however, industry assumes that all data travels over equally trusted network paths and does not attach weighting significance to the potential presence of a malicious actor in the network path.

With CASTOR, we envision a solution to these challenges. Network paths will be under continuous trustworthiness evaluation through the deployment of trust monitoring and assessment on each router.

Leveraging distributed and global trust assessments, the CASTOR orchestrator will ensure that the QoS and trust needs of a given application's data transmission are satisfied and maintained with dynamic automated network path reconfiguration if needed. With trustworthiness constraints agreed and embedded up front into network operations, surveillance domains can be confident that the trustworthiness demands associated with any of the data being received have been satisfied by the network.

8.2.4.1 Zones & Multi-domain network routing

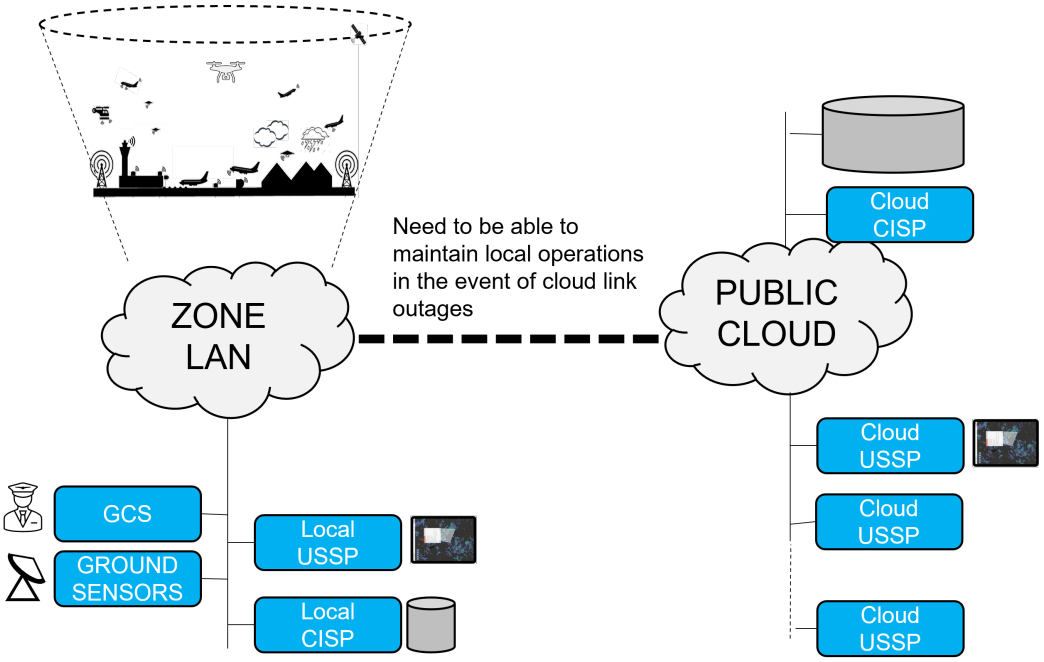

Figure 8.5: The Surveillance Continuum

As shown in Figure 8.5, a single unmanned airspace surveillance zone relies on connectivity beyond the local LAN. The benefits and economics of cloud deployment (elastic scaling, maintenance, redundancy and availability) hold the same attraction for aerospace as they do for any other enterprise. A

typical deployment involves interaction with a cloud-hosted CISP and USSP. Hubs such as airports deploy lightweight 'edge instances' of USSP/CISP on-prem to keep the shortest control loops inside the local LAN and enable continued operation in the event of cloud connectivity issues (edge/on-premise cache deployments are consistent with the deployment patterns described in ICAO UTM Framework [77]). This 'fallback' behaviour contains our network considerations to a single network domain. The 'normal' operation extending across network domains also needs to be addressed which introduces additional challenges from a trusted network path perspective.

In terms of surveillance data, we focus on data produced from a Collins Skyler radar which streams continuous payloads to a CISP node. These payloads include large aircraft detections via Automatic Dependent Surveillance–Broadcast (ADSB) reports, low-altitude UAV detections and periodic radar health and performance data.

8.2.5 To-be Reference Scenario 2: Collaborative Airport Operational Control Centres for Agile Decision Making

Airspace management and monitoring often span multiple administrative zones managed by different entities such as airports, municipalities, or private organizations. Aircraft movements, both manned and unmanned, routinely cross these administrative boundaries. Effective collaboration between neighbouring zones is essential to maintain comprehensive situational awareness, enhance flight safety, and respond rapidly and effectively to airspace incidents or emergencies.

While the surveillance capabilities of a single airspace zone operator are dimensioned and deployed to satisfy the safety needs of that zone, this is a cooperative ecosystem where operators need to share and leverage insights from neighbouring zones and regional authorities to extend visibility and collectively implement regional imperatives such as dynamic no-fly zones and emergency traffic rerouting.

Inter-zone sharing of airspace data (Figure 8.6) introduces significant challenges, including inconsistent data quality, potential security vulnerabilities, and varying trust levels associated with each zone's infrastructure. Without a reliable mechanism to assess and communicate data trustworthiness, operational decision-making across zones becomes slower, less effective, and potentially compromised.

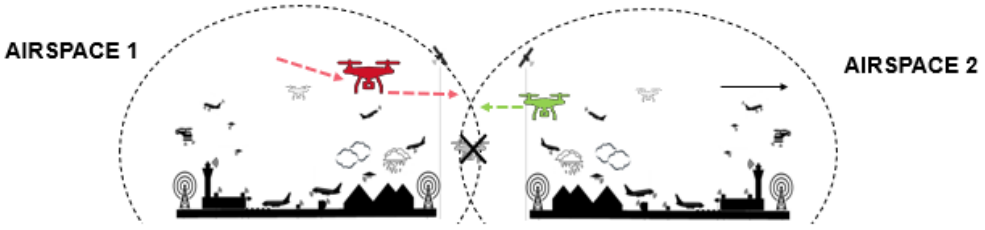

Figure 8.6: Multiple airspace domains require coordination and cooperation

In the current operational paradigm, different airspace surveillance zones exchange critical information to build a broader operational picture. Sensor observations (e.g., radar/ADS-B) and common information (e.g., dynamic airspace restrictions, weather) are disseminated via the shared CISP, enabling authorised consumers to subscribe by area of responsibility.

Figure 8.7 illustrates this existing data-sharing architecture, depicting how various zones contribute to, and consume from, these common information services to build a broader operational picture.

While the model illustrated above enables information flow, it also exposes a major vulnerability: the lack of verifiable trust and integrity assurance for data originating from external sources or passing through multiple domains. This absence of a standardized, end-to-end trust framework means that each zone independently evaluates data reliability, often through manual processes or simplistic criteria, which can

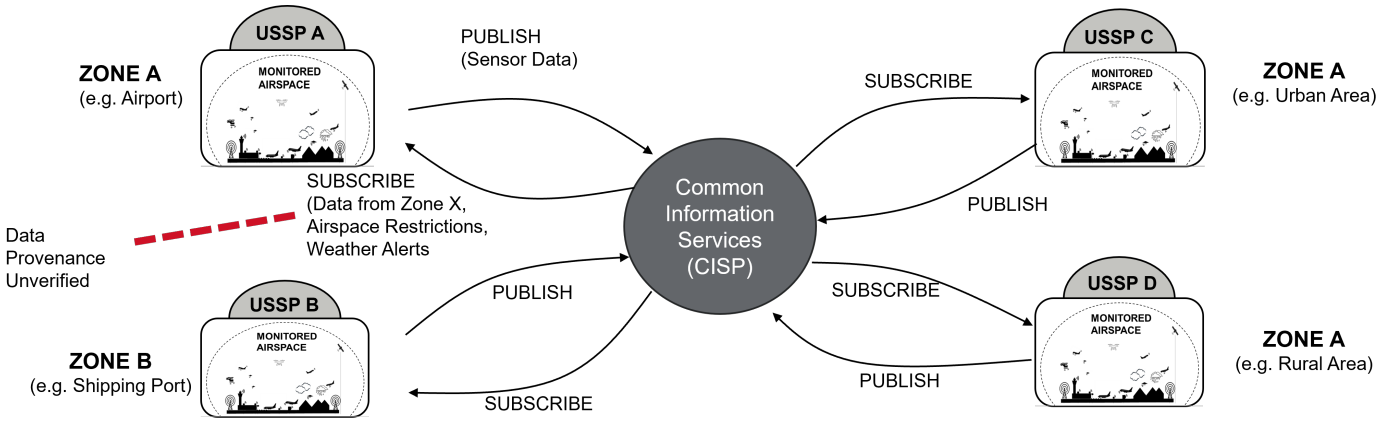

Figure 8.7: Current Inter-Zone Data Exchange Model

lead to delays and inconsistent decision making. Each zone has limited visibility into the trustworthiness of incoming data, making it difficult to assess whether the information has been securely transmitted and is reliable for decision making. Emergency coordination or rapid response actions can be hindered by uncertainty over the trustworthiness and timeliness of the incoming information.

With CASTOR, we seek to directly address this multi-domain hazard through a mechanism to relay high level trust assessments alongside the data itself when it traverses domain boundaries (e.g., via the CISP). The diagram below illustrates this improved scenario, where trustworthiness indicators associated with data from different originating zones become visible to consuming domains. This newfound transparency in data provenance allows receiving domains to intelligently weigh their decisions, prioritize actions, and more effectively reconcile potentially conflicting information.

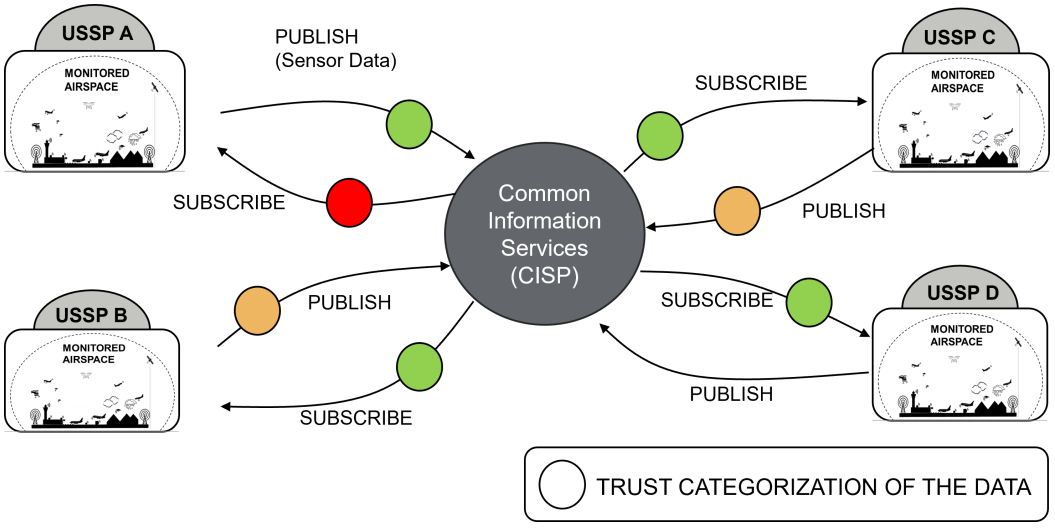

Figure 8.8: Exchanging trust assessments with data can improve operational security and efficiency

The following scenarios focus on LAN, campus, and cloud routers - the critical path for U-space surveillance traffic.

8.2.6 Reference Scenario 1 User Stories

A Skyler radar unit at a major airport is scanning the airspace, detecting all aircraft and UAVs in its range. The radar identifies a rogue drone flying near restricted airspace, potentially heading toward a busy flight corridor or an aircraft approach path. The radar relays real-time tracking data to the local CISP. From here it is observed by the site USSP from where the operator is alerted to the presence of the rogue

aircraft. This data is mission-critical - delayed or tampered information could allow the rogue drone to evade detection or cause a serious safety incident near passenger aircraft.

In today's deployments, if a router is unknowingly compromised and silently drops or delays surveillance traffic, the issue is often detected only indirectly when the USSP notices missing updates or the TCP session stalls. Operators must then investigate manually, and the time to isolate the faulty router and restore a clean path is dominated by human procedures rather than network mechanisms. CASTOR's value proposition is that it detects trust degradation at the infrastructure layer and restores compliant routing automatically.

8.2.6.1 CA.US1a – Nominal Operation

CA.US1a - Nominal Operation

As the airport surveillance radar publisher, I want my continuous observation stream delivered to the on-site CISP over a local path that satisfies my **{Performance, Availability}** needs, so that the USSP presents untampered, real-time tracks of detected UAVs.

User Story Confirmation

The radar establishes a https connection to the CISP and is allocated a session to authenticate and subsequently receive broker endpoints and topic mappings. The radar will connect to the endpoints using the MQTT protocol and exchange structured JSON messages representing radar state, ADS-B detections, tracks, heartbeats, and health. A demonstration USSP Console will visualise detections and highlight those that do not correspond to known aircraft. Each detection will have associated visual cues to relay the trustworthiness of the detection.

User Story Workflow

In Figure 8.9, we see a high-level flow of normal data exchanges between a co-located radar and CISP node. Data is published to the CISP node by the radar and then consumed by a USSP which continuously analyzes the data stream. If the USSP observes a detected object that it does not have in its approved inventory, then it generates an alert to notify the operator.

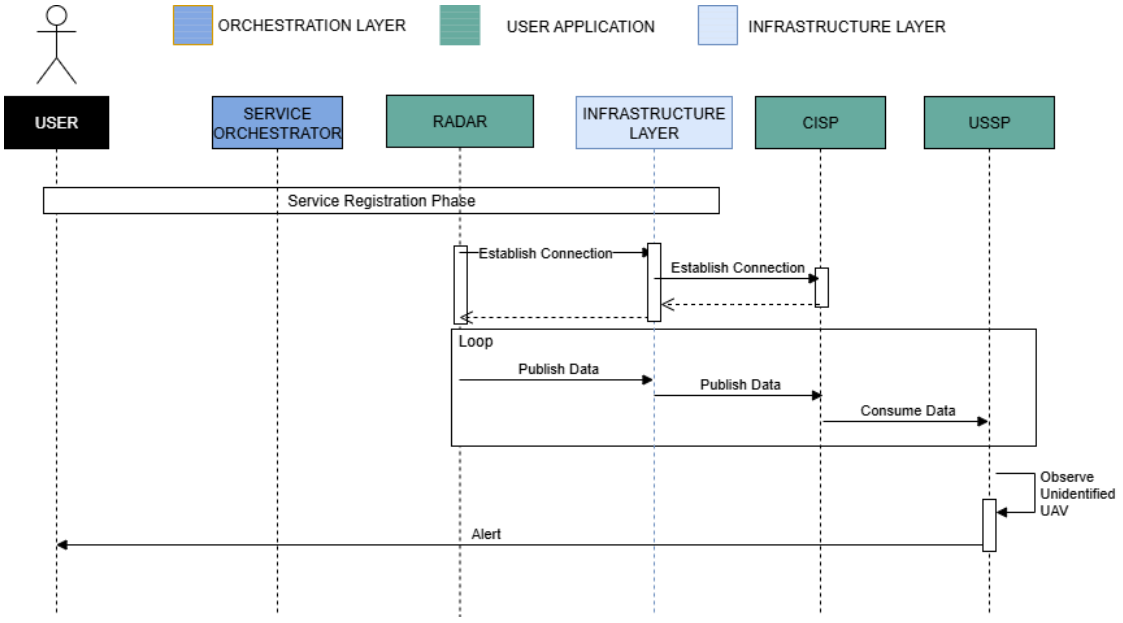

Figure 8.9: Normal behaviour without CASTOR

In Figure 8.10, we see an example of a router being manipulated to drop a key data stream so that the USSP never observes the threat and raises an alarm. The *Black Hole* represents the dropping of data (such as redirecting to /dev/null).

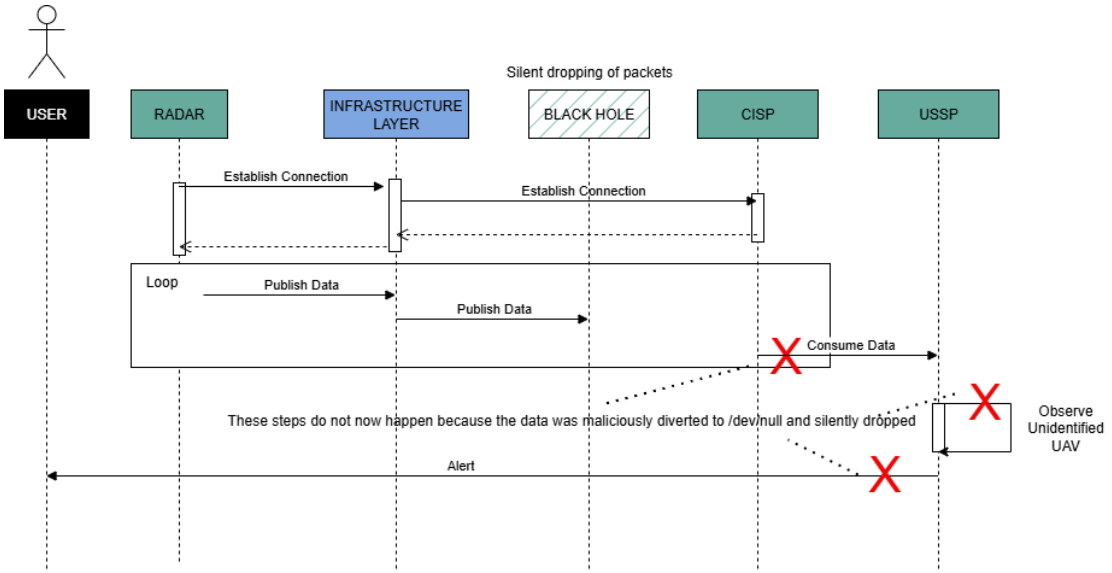

Figure 8.10: Attack behaviour without CASTOR

Although we depict the example of a router manipulation, a similar scenario unfolds in the event of other attack types, router malfunctions or link issues that would also result in the USSP being starved of key intelligence originating from the site radar.

Reference Values

CASTOR is not involved in this user story – this is verification that the system operates without CASTOR. The purpose is to capture baseline measurements of latency, jitter, and resource usage under controlled lab conditions. These results form the ground truth against which CASTOR-enabled scenarios (CA.US1b, CA.US1c) will later be compared. The figures provide a reference snapshot of how the system performs in its unmodified state. Subsequent user stories will focus on the relative change introduced by CASTOR mechanisms such as router attestation and trust-aware routing.

Table 8.7: Reference Values for Use Case 1, Scenario 1

Measurement	Description	Value / Scenario
Base latency	Radar $\rightarrow$ CISP, including campus routers & virtual switches)	Average and 95th percentile one-way delay. To be gathered and serve as ground-truth path delay (Typical zone production target < 100 ms)
Base Jitter	p95Latency – p50Latency	(Typical zone production target ≤ 10 ms)
Base-CPU / memory	Mean vCPU % & RAM per vRouter	To be gathered and serve as ground-truth resource consumption

These baseline measurements carry no pass/fail criteria. Subsequent user stories will introduce Δ-*KPIs*. We believe this approach is sensible for evaluation in a lab environment which cannot hope to accurately reproduce a fully dimensioned real-world deployment. Evaluation of CASTOR should not rely on faithfully duplicating such a deployment with all the physical distances and heterogeneous vendor equipment involved. Rather, we want to focus on identifying the potential overhead that CASTOR could incur and observe whether such overhead could be reasonably subsumed into a commercial deployment.

8.2.6.2 CA.US1b – Performance/Integrity Degradation

CA.US1b – Performance Degradation

As the radar surveillance application, I want my data stream to be automatically rerouted whenever the path's latency or integrity conditions fall below the SSLA, satisfying my {**Performance, Availability, Integrity**} needs, so that data destined for the USSP remains fresh and accurate.

User Story Confirmation

In a CASTOR-enabled deployment, the application flow remains the same as US1a: the radar performs its brief HTTP bootstrap with the CISP to obtain broker details/credentials, then publishes surveillance data to the same topics. The difference is that, in parallel, the site operator has provisioned an SSLA with CASTOR, and domain routers periodically emit ATL reports. The Orchestrator continuously computes end-to-end ATL for the active radar→CISP path and compares it to the route's RTL thresholds; while compliant, no behaviour changes are visible to the applications.

If performance degrades (e.g., a hop's ATL drops below RTL), the Orchestrator selects a compliant alternate path and applies a network-level reroute. This underlay change is transparent to the radar and CISP: broker endpoints, topics, and credentials are unchanged; the radar's transport session is preserved. From the USSP's perspective, track updates remain fresh and continuous.

User Story Workflow

In Figure 8.11, we see how CASTOR will detect the compromised situation and immediately initiate corrective actions to maintain the flow of critical data and ensures that operational integrity is maintained.

CASTOR KPIs

The following observational deltas describe the metrics to be measured during CA.US1b. They are evaluation objectives used to characterise CASTOR's runtime impact compared to the baseline conditions established in CA.US1a. The intent is to verify that CASTOR's trust-aware rerouting can preserve data freshness and continuity while introducing minimal additional overhead.

Table 8.8: CASTOR KPIs for Use Case 1, Scenario 1, User Story 1(b)

KPI	Definition	Target Value
Δ Latency overhead	Average latency with CASTOR minus the average latency captured during operation without CASTOR. Latency is measured Radar $\rightarrow$ CISP, including campus routers & virtual switches)	$\leq$ +5 %
Δ CPU overhead	Percentage of computing resources required at the router level for instantiating the CASTOR TCB exposing all security functions and trust extensions – CPU Resources% Base per router.	$\leq$ +10 % when instantiating the CASTOR TNDE as an "*untrusted app*" without isolation (no TEE equipped), $\leq$ +30 % TNDE artifacts running isolated in enclaves (with TEE)
Δ Bandwidth Control-plane share	ATL reports + reroute messages as % of data-plane bandwidth.	$\leq$ 1%

KPI	Definition	Target Value
Service-interruption duration	Duration between the compromise of a router (in a manner CASTOR is configured to detect) leading to an SSLA violation and the moment normal end-to-end data delivery resumes over a compliant path. This includes CASTOR's detection latency and the subsequent reroute, and is measured solely from the CISP's observation of surveillance-message arrival times (no clock synchronisation required)	$\leq 5s$;

These measurements will help determine whether CASTOR maintains service continuity under transient degradation conditions without incurring significant delay, loss, or resource overhead. In this user story, latency is measured end-to-end across the data plane (Radar $\rightarrow$ CISP), independent of any control-plane activity. The measurement tooling records the time of packet departure and arrival and remains oblivious to whether a path switch has occurred. Accordingly, the observed latency represents the steady-state performance of the network both before and after CASTOR performs a reroute, not the duration of the reroute decision process itself. This ensures that any reported Δ-latency reflects only the intrinsic characteristics of the forwarding path and not the orchestration or path reconfiguration mechanics.

Similarly, the vRouter CPU and memory overhead will vary depending on the number and type of active trust mechanisms enabled in a given test configuration (e.g., attestation depth, reporting frequency, or encryption scope). This overhead is therefore not expected to remain constant across all CASTOR experiments. Where additional integrity verification or telemetry incurs measurable processing cost, the results will help characterise the trade-off between increased resource utilisation and improved network trustworthiness. The objective is not to minimise CPU consumption in isolation, but to understand how much computational overhead is acceptable to achieve stronger security assurance within realistic deployment constraints.

The service-interruption duration metric captures the practical impact of an attack in which a router silently drops or diverts surveillance traffic. In a non-CASTOR environment, such failures are detected only through application symptoms (e.g., stalled TCP sessions or missing heartbeats), followed by human investigation - often taking minutes or longer. Indeed, when spanning multiple domains this can take a lot longer to coordinate. In contrast, CASTOR detects trust degradation at the network layer, triggers a reroute, and restores data visibility automatically.

This metric therefore measures the entire "gap" seen at the CISP: from the moment the attack begins, through CASTOR's detection and SSLA evaluation, to the point where a compliant path is re-established and new surveillance messages arrive. The publisher remains unaware during the attack; only the consumer-side message arrival timestamps are required. The objective is not zero interruption - some brief visibility loss is unavoidable - but a significantly shorter outage compared to traditional operational processes.

8.2.6.3 CA.US1c – Trust Degradation Alert

Collins Aerospace use networks under the control and supervision of third party suppliers but also operate as in independent network operator ourselves in the airports domain in which we deploy and manage secure network backbones. From this perspective, our interest in CASTOR extends beyond users to deployers. One aspect we would like to see offered by CASTOR is alerting to circumstances in the network that CASTOR is uniquely positioned to identify and bring to our attention.

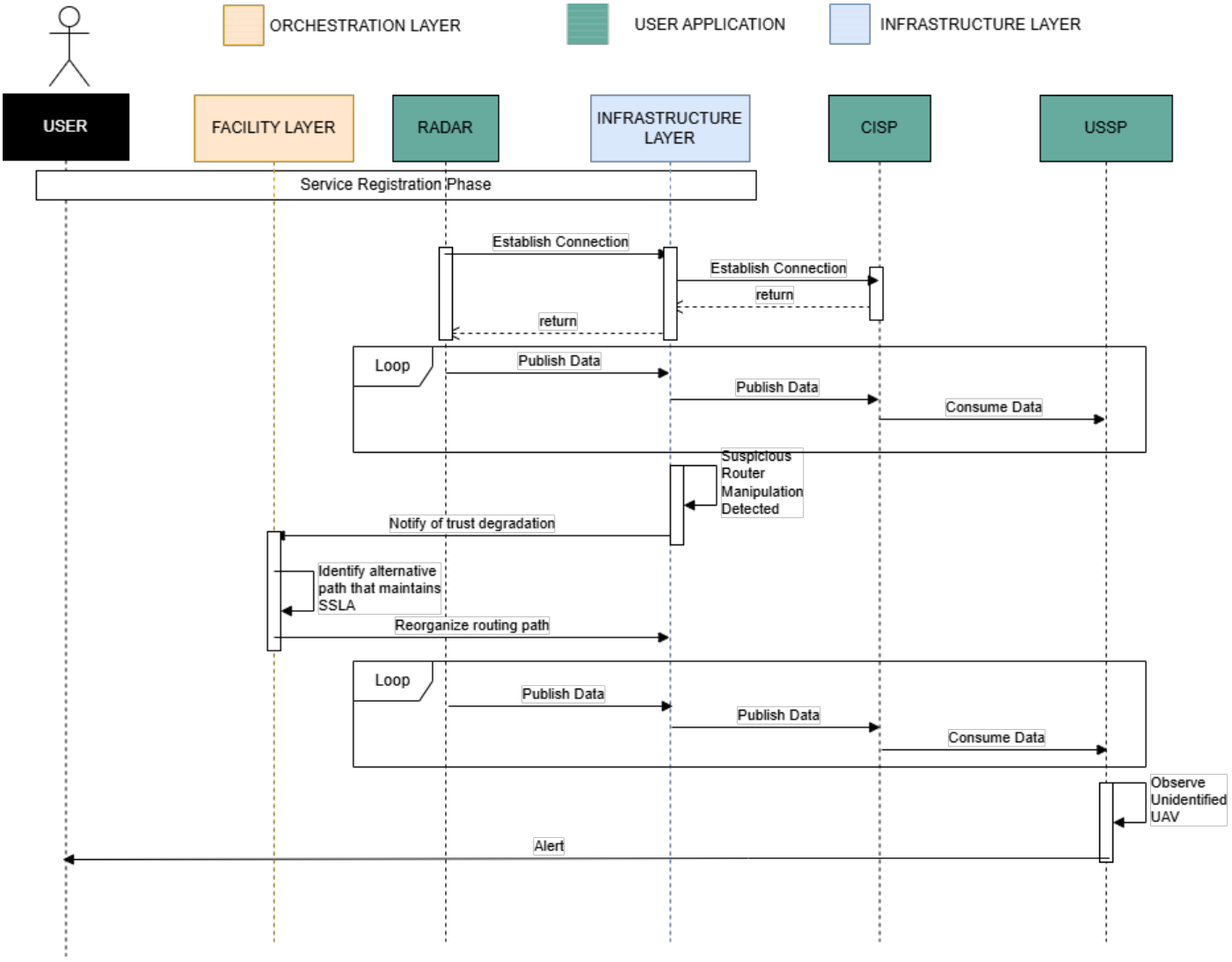

Figure 8.11: Attack prevented from starving CISP of attack-revealing data

CA.US1c – Trust Degradation Alert

As the NetOps engineer, I want to receive a timely alert whenever CASTOR removes a router from a network path because its ATL falls below a flow's RTL, satisfying my **{Observability, Integrity}** needs, so that I can investigate and take remedial action before service degrades.

User Story Confirmation

CASTOR continuously (i) aggregates ATL Reports from routers, (ii) computes end-to-end ATL for the radar→CISP flow, and (iii) compares it to the flow's RTL. This runs under the applications, so the radar/-CISP/USSP stack behaves exactly as in US1a/US1b unless intervention is required. When a router's trust posture significantly degrades (i.e., its ATL falls below the level needed to contribute to an SSLA), CASTOR marks the path non-compliant and (as in US1b) may reroute to a compliant path to protect mission-critical freshness. Independently of application symptoms, CASTOR raises a NetOps alert. That alert is delivered via the Facility Layer interface and includes the router identity and domain along with actions taken (e.g., router removed from path, reroute applied). This means NetOps is informed even when we experience no visible impact due to rapid remediation, and the event is recorded for audit and cross-correlation via the CASTOR Telemetry API.

User Story Workflow When we use the term 'significant' in Figure 8.12, then we refer to situations in which the ATL of a router degrades from a point where that router could contribute to the satisfaction of an SSLA to a point in which it cannot.

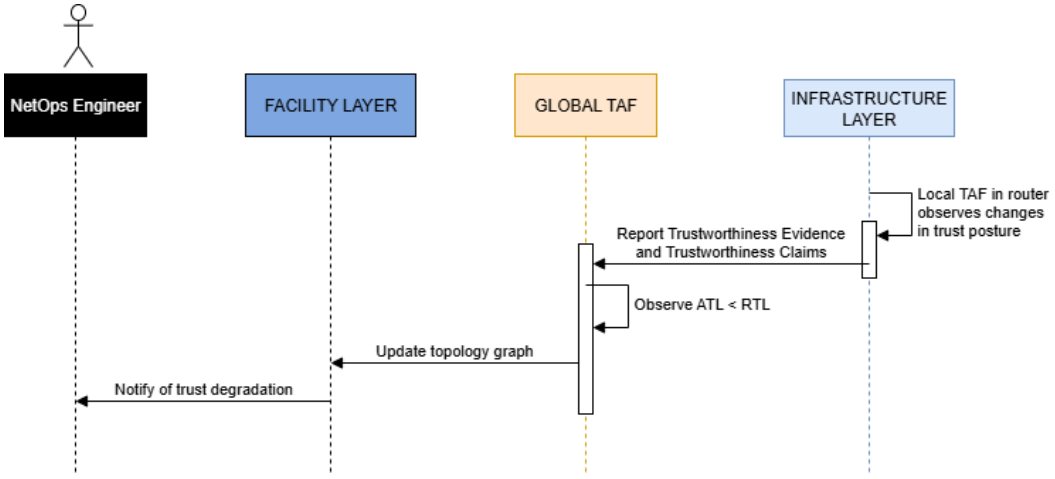

Figure 8.12: Significant ATL degradation results in operator notification

Note that even though we only focus on trust degradation alerts in Figure 8.12, there is also a continuous stream of telemetry data being captured by the Service Orchestrator and exposed through the Telemetry API that is available to the NetOps Engineer. This has been described in Chapter 6.

CASTOR KPIs

The purpose of this user story is to ensure alerts are generated and relayed onwards appropriately in a timely manner. This enables network engineering staff to observe and respond to trust weakening activities in the network while also feeding into key audit trails maintained by network engineering to perform meta analysis and correlation. The key observability targets we have identified are listed in Table 8.9.

Table 8.9: CASTOR KPIs for Use Case 1 - User Story 1(c) NetOps Alerting

KPI	Definition	Target Value
Alert-delivery latency	Time from router downgrade decision to receipt of alert by the NetOps monitoring system	< 400 ms (does not include Global TAF calculations)
Alert accuracy	Qualitative measure of how reliably CASTOR alerts reflect genuine, tested trust-degradation scenarios. This entry does not claim coverage of all attack types; it applies only to the specific lab-tested cases to be documented in the test plan.	Qualitative goal: alerts should accurately reflect genuine degradations in tested scenarios while minimising false flags.
ATL Report Freshness	Time to detect events-of-interest causing trust level reduction ($ATL < RTL$) and their subsequent sharing to the Global TAF for further processing	Typical $\leq$ 400ms
	NOTE: As detailed in the core trust assessment challenges in Chapter 4, such hard timing constraints may not be sufficient for on-the-fly trust opinion calculations consdering also the time required for the collection of fresh evidence. In such corner cases, the Global TAF needs to make a decision (based on pre-configured policies) on whether to wait for fresh evidence or converge to a trust decision manifesting on the trust-related information received in a previous trust assessment epoch (increasing, of course, the inherent uncertainty of the decision itself). This interplay between accuracy and freshness of the trust evaluations constitutes a key dimension that will be evaluated as part of the CASTOR Trust Assessment Framework.	

Alert-accuracy figures refer only to the subset of tested trust-degradations (for example: firmware drift, configuration tampering detectable via attestation, and selective packet blackholing) and the specific attestation signals available in the experimental setup. They are conditional on the defined threat model and instrumentation and do not imply exhaustive coverage of all attack vectors. The evaluation will report true/false-positive and true/false-negative counts for each exercised scenario, together with discussion of any observed false flags or benign events that triggered alerts.

8.2.7 Reference Scenario 2 User Stories

A radar in Zone A detects an unidentified aircraft flying at an unusual altitude and trajectory, potentially posing a risk. Such an observation is relevant beyond surveillance Zone A – neighbouring zones will also likely be interested as the aircraft may travel to their zones or the event may even lead into a wider picture of concerning airspace activity across multiple zones. The radar data from Zone A is streamed to a central CISP on the cloud from where a USSP responsible for the management of a neighbouring airspace zone B consumes the observational data and assesses the risk and relevance of the observation. Upon deciding that such an observation warrants concern, it alerts the airspace operator.

When sharing particularly sensitive data such as radar capabilities and current performance metrics with third parties, the operator would like to know that the network path is trustworthy and would like to be made aware of any significant drops in trustworthiness – even if just temporary – so that it can refrain from transmitting such data until trust is (re)established. Observational data such as aircraft track detections are crucial to share with neighbouring zones regardless of whether a trustworthy path can be achieved or not. In the event that the path is not trustworthy, the operator would ideally like to relay this information with the observations so that neighbouring zones consuming such data can make informed decisions.

When consuming data originating from a different provider, an airspace zone B would like some insight into the quality and trust provenance of this data. In the case of radar detections, it would like to know how reliable the measurement was by getting into some insights into the current health of the radar – a key insight that may radars regularly report. In addition, the operator would like to know the network path trust provenance of the observational data recorded by neighbouring zones.

To address these challenges of trust-dependent sharing of radar capabilities and trust-weighted observational data, we rely on the availability of a tiered SSLA capability in CASTOR. We would like a 'maximum trust SSLA' in which the integrity of all routers in the network path must be of a very high level. When this SSLA is being satisfied, then we can confidently transmit highly sensitive data such as radar capabilities and dynamic radar performance. When this link-level high trust SSLA, cannot be achieved then we would like a 'minimum-trust SSLA' as a fallback. For this SSLA, we are not overly concerned with individual router integrity – we just want a performant network path overall that can accommodate the typical SLA metrics in terms of throughput, latency, etc. When this SSLA is in effect (due to the maximum-trust SSLA not being currently achievable), then the radar application will suspend sending the highly sensitive radar capabilities and performance metrics but continue sending the airspace observations. By including insights into the trustworthiness of the network path in effect with the observational data, the ultimate consumers of such data can adopt calibrated reaction strategies.

8.2.7.1 CA.US2a – Baseline consumption (blind trust)

CA.US2a – Baseline consumption (blind trust)

As USSP B in Zone B, I ingest radar observations from neighbouring zones via the shared CISP over a network path that is lacking **{Provenance}** trust metadata, so I must treat all sources as equally reliable.

User Story Confirmation

Zone A's radar (and any other producers) publish surveillance observations to the CISP under area-of-responsibility (AoR) topics (e.g., . . . /obs/aor/region/sector/class). After a brief HTTPS bootstrap to authenticate and discover the AoRs relevant to its airspace, USSP B subscribes by AoR, not by publisher identity. The origin (zone/site/sensor) is carried in each message. No provenance stream is used in this baseline, and no CASTOR overlay is present.

Upon receipt, USSP B decodes and validates the structured surveillance messages, then performs its normal filtering/fusion and freshness checks to maintain a live track picture and trigger standard alerts (e.g., "unknown" by local rules). Because no trust provenance accompanies the feed, USSP B does no weighting by path trust and bases decisions solely on content quality and timeliness. This establishes the blind-trust baseline against which the CASTOR-enabled US2b will be compared.

User Story Workflow

In Figure 8.13, we depict surveillance observations in Zone A being published to a central, shared CISP from which a USSP in Zone B consumes. Although not depicted, the situation is similar for a USSP in Zone A consuming surveillance observations originating in Zone B. Indeed, there may be multiple neighbouring surveillance zones publishing and consuming data through a central, shared CISP. In all cases, data is published to the CISP without any trust provenance indicators leaving each surveillance zone ignorant of any dynamic trust changes that may have occurred in the neighbouring zones and unable to consider and reflect any such trust changes in its overall surveillance analysis.

Reference Values

In common with CA.US1a, CASTOR is not involved in this user story. The purpose here is to record baseline communication characteristics between neighbouring zones so that the CASTOR-enabled scenario

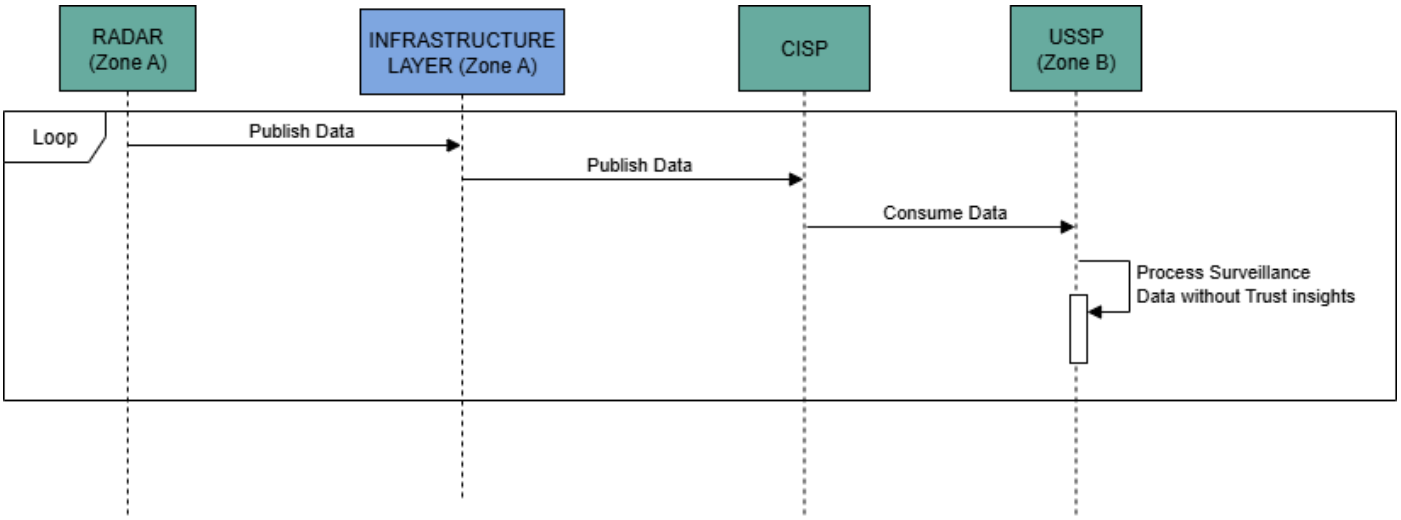

Figure 8.13: Zone B has no insights into the trust provenance of data consumed from the CISP

(CA.US2b) can later be compared under identical lab conditions. The following are reference observation targets to be collected from the live system. They do not represent pass/fail criteria, but serve as a performance baseline for assessing CASTOR's overhead and benefits. These measurements are presented below in Table 8.10

Table 8.10: Reference Values for Use Case 1, Scenario 2

Measurement	Description	Value / Scenario
Base latency	Average and 95th-percentile one-way delay (Zone A Radar $\rightarrow$ Zone B USSP), excluding any processing delay inside the cloud-hosted CISP. This reflects only the inter-zone transport and campus routing, which are the elements affected in later CASTOR-enabled scenarios.	To be gathered and serve as ground-truth path delay. Typical inter-zone target $<$ 100 ms
Base jitter	p95 - p50 of (Zone B USSP packet arrival - Zone A Radar packet departure)	Variability reference for Δ-jitter
Base throughput	Msgs/s and kbps of Radar stream	Establish maximum attainable throughput without CASTOR

These measurements provide a neutral reference of data-plane behaviour and inter-zone latency before any CASTOR components or provenance extensions are introduced. They allow later CASTOR-enabled runs to be expressed as relative deltas (e.g. Δ-latency, Δ-loss, Δ-CPU) rather than absolute targets. The objective is not to validate system performance against operational SLAs, but to quantify the natural variation that occurs in a blind-trust environment so that CASTOR's influence on delay, jitter, and reliability can be properly assessed.

Inter-zone latency in this baseline reflects only the underlying network and cloud topology, without CASTOR involvement. The lab constrains latency to realistic operational ranges so that later CASTOR-enabled scenarios can be expressed as **relative deltas** rather than absolute WAN requirements.

8.2.7.2 CA.US2b - CASTOR tier enforcement (network side)

The system being considered spans two separate administrative domains: Zone A with a radar application, its own routing infrastructure and orchestrator, publishing data to a cloud-based CISP; Zone B with

its routing infrastructure and orchestrator, and a USSP consuming data from the same cloud-based CISP. Two SSLAs are registered in Zone A for the radar application:

- Default SSLA (high trust): Performance: latency/jitter/loss within tight bounds; Availability: $\geq 99.99\%$; Integrity: all path hops must meet/ exceed RTL ($ATL \geq RTL$ per hop). With this SSLA in effect we have the necessary condifentiality assurance in place to share sensor capabilities and health observations to the CISP;
- Fallback SSLA (performance only): same targets for performance and availability. There is no integrity target included - no per-hop RTL requirement (trust not guaranteed). With this SSLA in effect, we do not have the necessary confidentiality assurance in place to share radar health and capabilities data with the CISP.

When the fallback SSLA is in effect, CASTOR should periodically recheck if the default SSLA can be restored. The objective is to maximize the percentage of time that the network path satisfies the default SSLA. The radar's active SSLA is published and updated in realtime as appropriate such that it can be interrogated by the radar application. The radar publishes the current trustworthiness of the network path (as reflected by the SSLA in effect), and any subsequent changes, to a dedicated provenance topic on the CISP. This topic is subscribed to by consuming USSPs so they can be made instantly aware of the current network trustworthiness of the arriving data.

CA.US2b - CASTOR tier enforcement (network side)

As the **Zone B USSP**, I examine the trust indicator associated with all zone A surveillance observations that I consume from the CISP. The Zone A radar is initially provisioned by CASTOR with a network path satisfying the default SSLA with high **{Performance, Integrity, Availability, Confidentiality and Provenance}** needs resulting in a Trust Indicator of High being associated with each observation. An event occurs in Zone A routing infrastructure resulting in the default SSLA no longer being able to be satisfied and the CASTOR orchestrator identifies a path that can satisfy the fallback SSLA. The change in SSLA is observed by the radar application and the Trust Indicator associated with outgoing observations is downgraded to Low. The change in Trust Indicator is immediately observed by the USSP in zone B.

User Story Confirmation

The system spans two domains: Zone A publishes radar observations to a cloud CISP (AoR topics); Zone B's USSP subscribes to those AoR feeds. In parallel, Zone A's radar subscribes to CASTOR's Trust Exposure Layer for its flow. While the Default SSLA holds, CASTOR attests a HIGH trust state for the radar$\rightarrow$CISP path. The radar relays CASTOR's path-trust assertion to the CISP provenance topic for that flow (e.g., prov/flow/{flow-id}/pathtrust) declaring TI=HIGH over a configurable validity window.

A routing event in Zone A degrades integrity below RTL. CASTOR flips the SSLA to Fallback and, as a registered consumer of such an event, notifies the Radar application of the change in SSLA. The radar associates the change in SSLA with a low integrity network path being in operation and publishes this update to the same provenance topic on the CISP. The radar, now being aware that it is transmitting on a path without the confidentiality assurance of a high integrity path, suspends transmission of radar health and capabilities data to the CISP until the high integrity SSLA can be restored, but continues to transmit surveillance observations uninterrupted to the same AoR topics.

USSP B, already consuming observations, also consumes the flow's provenance stream and weights each observation by the current trust window (HIGH $\rightarrow$ full confidence, LOW $\rightarrow$ reduced weight). When CASTOR later restores the Default SSLA, a fresh HIGH assertion is published; USSP B returns to full-confidence processing without any change to topics or application wiring.

User Story Workflow

In Figure 8.14, we depict the sequence of actions we envisage to provision a dual SSLA scenario in which we can fallback to use of a lower integrity path in the event that our preferred high integrity path cannot be provisioned for a period of time. The SSLA that is currently in effect for the Radar application can be queried from the CASTOR Trust Exposure Layer. An application can subscribe to the Trust Exposure Layer to be notified of changes to the SSLA associated with the application's network path. Knowledge of the SSLA in effect, enables the application to adapt its behaviour according to the known trustworthiness of the network path.

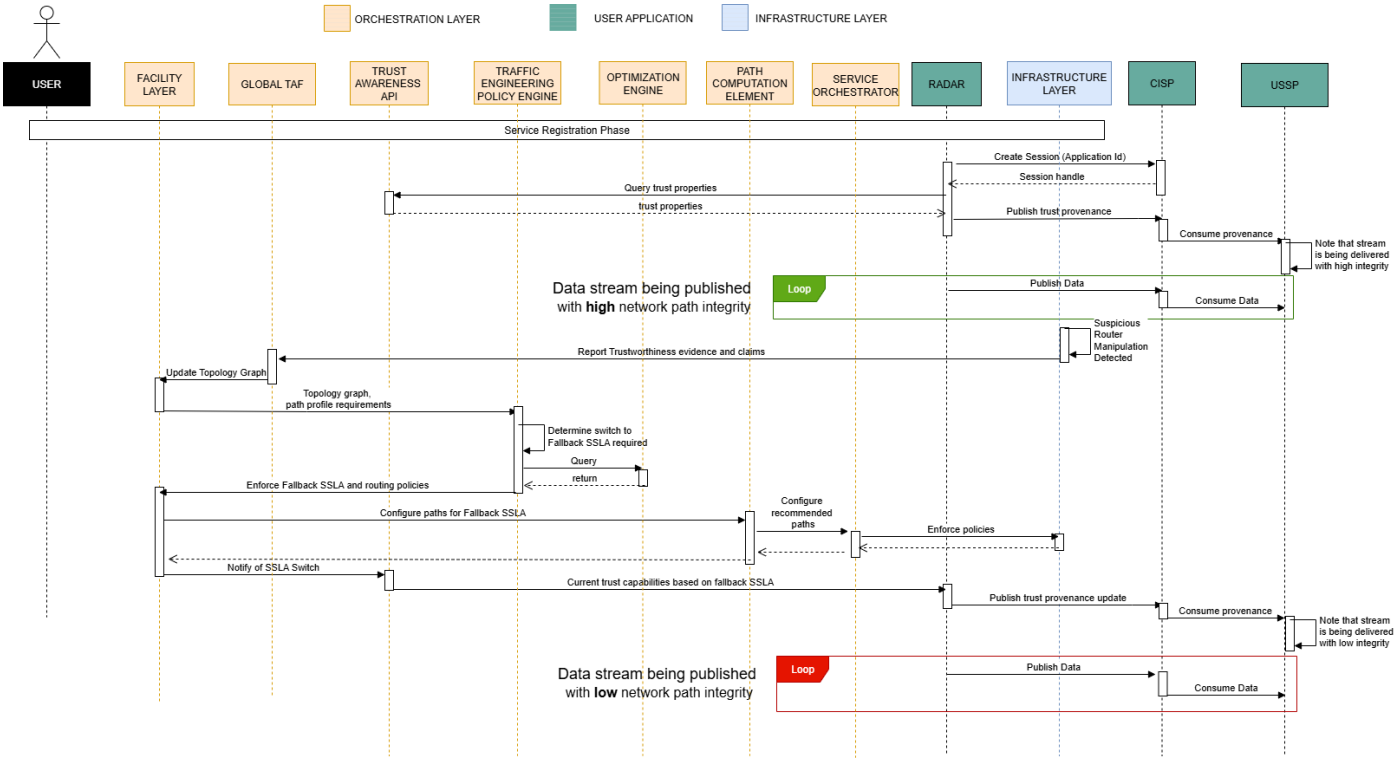

Figure 8.14: Zone B now has insights into the trust provenance of data consumed from the CISP

CASTOR KPIs

Here we have two key objectives. The first is to measure the overhead imposed on our deployment model by the use of CASTOR. To this end we establish baselines through measurements of latency, message loss (if any), jitter and throughput. Our second objective is to assess the performance of SSLA switching in response to observed trust degredation in the network routers. The indicative observation targets presented in Table 8.11 are used to characterise CASTOR's effect and responsiveness under controlled lab conditions. They are not success criteria, but reference points for comparing performance and behaviour across runs.

Table 8.11: CASTOR KPIs for Use Case 1, Scenario 2, User Story 2(b)

KPI	Definition	Target Value
Δ Latency Overhead (cross-domain)	Change in end-to-end latency (p95 receive at USSP B - send from Radar A) compared with baseline.	$\leq$ 5% increase on baseline measured in nominal user story
Δ Loss Overhead	Additional message loss compared with baseline	$\leq$ 0.05% increase on baseline measured in nominal user story

KPI	Definition	Target Value
Δ Jitter Overhead	(p95 - p50) E2E delta versus baseline	$\leq$ 5% increase on baseline measured in nominal user story
Δ Throughput Overhead	Change in sustained message rate (msgs/s or kbps) compared with baseline.	$\leq$ 1% decrease on baseline measured in nominal user story
End-to-End Tier Change Propagation Time	Time from CASTOR's internal SSLA flip decision (t_{flip}) to the point at which the USSP has applied the new trust tier in its data-fusion logic (t_{apply}). This captures CASTOR detection, orchestration, publisher update, and consumer reception as a single observable interval.	p95 $\leq$ 200 ms (lab environment - assuming no need for optimizer involvement).
Tier oscillation rate	We would like to avoid the scenario in which an intermittently faulty router or link results in yo-yo effect between default and fallback SSLAs resulting in excessive adaptation costs. We expect some form of hysteresis to be in place to limit this.	$\leq$ 1 per 10 minute

The Tier Change Propogation Time measurement quantifies the total duration between CASTOR detecting a trust degradation and the USSP applying the new tier - effectively measuring CASTOR's responsiveness and end-to-end coupling to the application layer. This also serves as an implicit qualitative measurement of the integration effectiveness between CASTOR and a third party application.

The Oscillation Rate measurement ensures that the system does not over-react to transient conditions such as a faulty router. Genuine cascades of trustworthiness issues may well require frequent tier oscillating. This cannot be avoided as CASTOR has to adapt to the circumstances it is faced with. What we have in mind is, in particular, a router that is regularly exhibiting an intermittent fault over short intervals. We would expect some form of dwell timers to ensure continuous health windows before re-accepting a previously quarantined router.

8.2.7.3 CA.US2c - Receiver-side trust interrogation

This use case approaches the issue of SSLA conformance from a different perspective. We work with surveillance equipment from a variety of vendors and not all can be immediately updated to query and act on current trustworthiness assessments. Every time new interfaces and capabilities are introduced into a complex multi-vendor ecosystem then we need to be able to accommodate a period of transition in which legacy applications continue to operate uninterrupted. To accommodate this eventuality, we investigate updating the receiver side (the CISP in our case) to query the trustworthiness provenance recorded in the originating domain of the client data. The CISP is modified to query the Trust Exposure layer and subscribe to updates so that it can dynamically categorize incoming data before it is streamed onwards to consumers.

CA.US2c - Receiver-side trust interrogation

As the **Zone B USSP**, I examine the trust indicator associated with all zone A radar observations that I consume from the CISP. Through CASTOR interaction, the CISP is aware of the trustworthiness of published data and relays this information to the **Zone B USSP**, I observe this indicator when I consume the data from the CISP node.
An event occurs in Zone A routing infrastructure resulting in the desired SSLA no longer being satisfiable and the CISP is notified of the updated trust values associated with the network path according to the {**Observability, Provenance**} requirements we expect CASTOR to satisfy. The CISP observes that the integrity constraint is no longer being satisfied and tags all surveillance data with a Trust Indicator of Low. The change in Trust Indicator is immediately observed by the **USSP in zone B**. This remains in effect until network conditions change and the orchestrator is able to re-establish a path satisfying the high levels of integrity laid down in the radar Default SSLA.

User Story Confirmation

When Zone A's radar establishes a session with the CISP, the CISP immediately queries the Trust Exposure Layer for the radar's application flow to determine the current SSLA tier / Trust Indicator and registers for change notifications. The radar then publishes its observations to the usual AoR topics (payloads unchanged). The CISP, having the flow identity from session setup, annotates the stream out-of-band by publishing a provenance message to the flow's provenance topic (e.g., cisp/v1/prov/flow/{flow-id}/pathtrust) that tracks the current Trust Indicator for that flow.

If the network SSLA changes (e.g., Default→Fallback), the Trust Exposure Layer notifies the CISP, which immediately publishes an updated provenance message for the same flow. USSP B, already subscribed to AoR observations, also subscribes to the provenance topic and weights incoming observations accordingly—without any changes to the observation payloads or topic structure.

User Story Workflow

In Figure 8.15, we present the flows involved when the CISP (on the data receiver end) takes ownership of querying the provenance of an incoming data stream by reaching out to the Trust Awareness Layer using the id provided by the application during session creation. The objective is to get the trust properties associated with the network through through which the data from the radar is reaching the CISP. We progress through the situation in which the trust properties associated with the incoming data stream change at some point resulting in the CISP having to update its classification scheme.

CASTOR KPIs

As in CA.US2b, the emphasis remains on the responsiveness and stability of CASTOR's propagation of trust-state changes. Here, the additional element is that the CISP acts as an intermediary: it queries CASTOR's Trust Exposure Layer and relays the resulting trust update to downstream consumers (e.g. USSP B). We therefore retain only the End-to-End Tier Change Propagation Time metric, but extend its scope to reflect this receiver-side chain. This is captured in Table 8.12

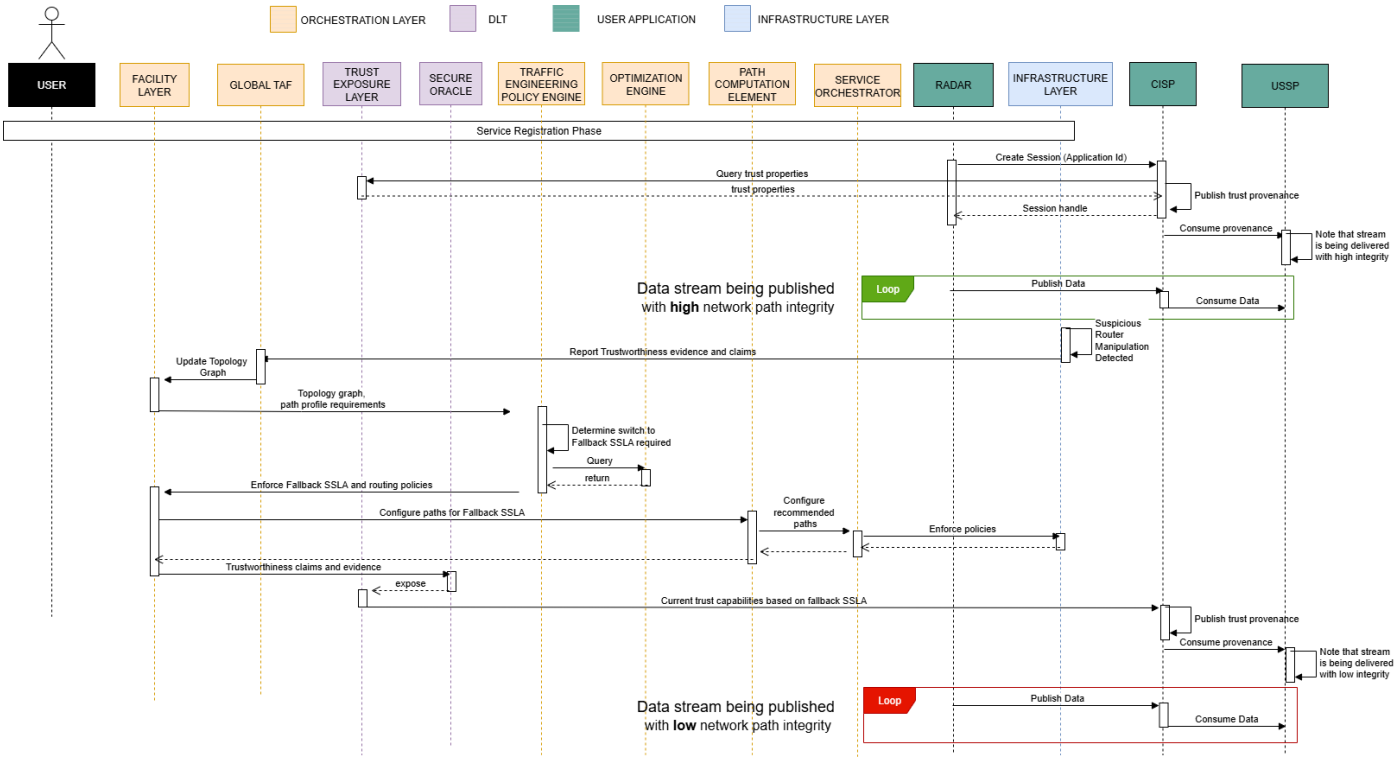

Figure 8.15: Instead of applications tagging with trust indicator, receiver can query origin trustworthiness

Table 8.12: CASTOR KPIs for Use Case 1, Scenario 2, User Story 2(c)

KPI	Definition	Target Value
End-to-End Tier Change Propagation Time (cross-domain)	Time from CASTOR's internal SSLA flip decision (t_{flip}) to the moment the downstream consumer (USSP B) recognises and applies the new trust tier (t_{apply}). This measurement now spans CASTOR's notification to the CISP, the CISP's publication of the updated provenance message, and its consumption by the USSP. It therefore captures cross-domain propagation delay between network, receiver, and consumer.	p95 $\leq$ 300 ms (lab), acknowledging additional inter-domain signalling compared to CA.US2b.

This measurement demonstrates that CASTOR's trust-state transitions remain visible and timely even when propagated via a CASTOR-aware intermediary (the CISP) rather than the original publisher. The small relaxation of the indicative goal (from $\leq$ 200 ms $\rightarrow$ $\leq$ 300 ms) reflects the extra messaging steps between domains. Success in this test confirms that CASTOR's trust-awareness can be extended to legacy or third-party systems without modifying existing data producers.

It is important to note that inter-domain network latencies - particularly those arising from internet or carrier backbones between Zone A, the cloud CISP, and Zone B lie outside the control of CASTOR. The indicative propagation target of 300ms includes headroom to absorb such variability in the lab. In a real deployment, these latencies would be dimensioned, monitored, and engineered through separate network-planning processes. CASTOR's responsibility is limited to ensuring that trust-state changes are propagated promptly across the available infrastructure; it cannot compensate for under-provisioned or highly variable inter-domain transport paths.

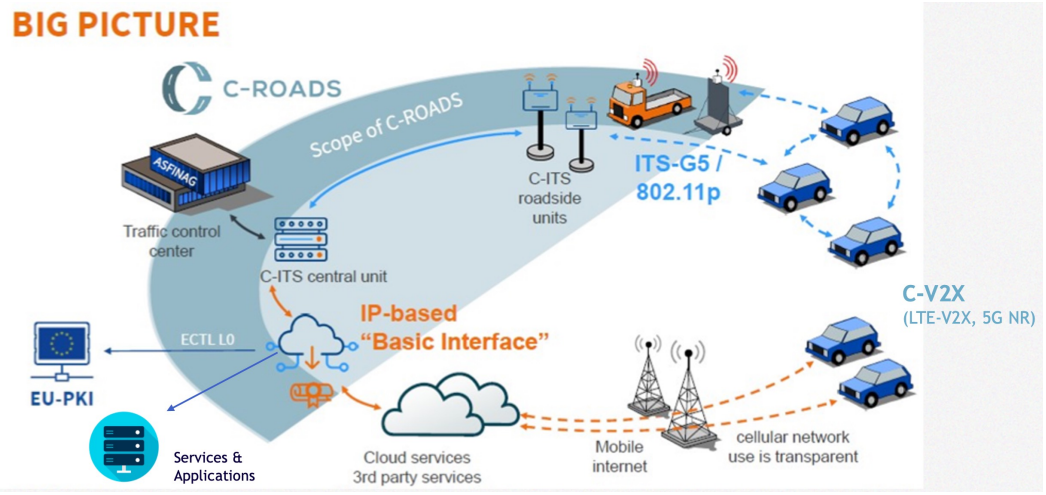

Figure 8.16: The big picture of C-Roads topology of European C-ITS systems

8.3 Trustworthy Communications of First Responder Mobile Units and the Compute Continuum

First responder units operate in highly dynamic and safety-critical environments, where uninterrupted secure communication and trustworthy system behaviour are essential. To maintain this trust, onboard units (OBUs) must frequently obtain updated security credentials from a Public Key Infrastructure (PKI) and stay current with firmware delivered via Over-the-Air (OTA) mechanisms. These procedures are foundational prerequisites for ensuring that any subsequent V2X communication performed by the vehicles—such as signing or verifying messages—is based on uncompromised software and fresh cryptographic credentials. Unlike conventional automotive systems that rely on extensive OEM backend infrastructures, first responder OBUs are compact, lightweight devices used by various authorities and deployed across heterogeneous operational contexts. Their mobility and large geographical coverage expose them to untrusted or partially transparent mobile network segments, which increases the importance of secure backend connectivity.

In this CASTOR use case, the focus is on securing the backend trust chain required to support these operations, specifically the communication paths between the V2X Application Server and the PKI/OTA servers. The use case captures scenarios in which OBUs must retrieve certificates or firmware through a V2N2V workflow, where the “N” represents potentially diverse and cross-domain network infrastructures. CASTOR introduces mechanisms to establish verified, high-integrity network paths across these infrastructures, ensuring that backend communication remains protected even when traversing multiple operators or administrative domains. This is particularly significant for first responder deployments, where agencies often rely on networks owned by different public or private entities, and where trust cannot be assumed merely because standard encryption exists. Cross-domain trust provisioning—supported by the CASTOR Trust Exposure Layer and orchestrator interactions—ensures that each domain can advertise its trust capabilities, allowing secure routing decisions to be made end-to-end.

By orchestrating trusted, policy-compliant, and dynamically monitored paths for certificate and update delivery, CASTOR ensures that backend operations remain robust against integrity breaches, delay, or malicious interference. As a consequence, the OBUs maintain fresh certificates, validated software, and updated security configurations—establishing the foundation upon which trustworthy V2X communication is later performed in real-world first responder scenarios. While message types such as CAMs and DENMs are not directly in scope for this use case, their credibility and authenticity in field operations depend entirely on the integrity and trustworthiness of these backend processes that CASTOR strengthens.

Defining mechanisms to enable performance and trust between first responders’ mobile units and the

compute continuum is a primordial goal of CASTOR, while first responders may be operated under different policy regimes and governed by different authorities.

The use case demonstration is split into the following two categories, which can be considered the archetype of many related use cases:

- UC 2.1 Digital certificate download for secure V2X communication.
- UC 2.2 Over-The-Air (OTA) firmware update of security-sensitive communication devices.

Both UCs are based on V2X communication, sharing a hybrid CCAM topology including diverse communication devices/technologies and mixed architectures. While not challenging in terms of performance (latency and throughput), they are demanding in the sense that they deal with (safety, security) sensitive data where reliability and provenance of data (i.e., trustworthy sources) are primordial. Furthermore, one needs to fit CASTOR's technology solutions to an existing and standardised solution framework.

8.3.1 "As-is" Scenario

C-ITS networks integrate diverse computing platforms and technologies, such as Roadside Units (RSUs), Onboard Units (OBUs, UEs), Aftermarket Safety Devices (ASDs), and other vehicular communication endpoints and services. This represents a hybrid communication system whose specification is included and has also been implemented and set up in several European countries in the framework of C-ROADS. Figure 8.16 shows the big picture of the hybrid topology of C-ROADS.

The actual route/path of transmission of data packets in the network is built on a best-effort basis without any specific requirements for the network. This means that no specific traffic engineering solutions are required or need to be developed by either the network owner/operator or the C-ITS applications to transmit the data. The end-to-end communication topology of ITS service access in V2X communication systems from the perspective of a V2X service provider is shown in Figure 8.17. The figure also focuses on the two services in the V2X vertical that the scenarios will rely on, namely a Public Key Infrastructure (PKI) backend and an OEM server issuing Over-the-Air (OTA) updates.

Typically, the network requirements in standard C-ITS use cases are as follows:

- End-to-end latency: less than 80 ms
- Throughput: 15-20 kbps, with occasional maximum of up to Mbps range
- V2N traffic is generally transmitted using TCP/IP and optional application layer protocols (e.g., MQTT)
- For end-to-end communication requesting the ITS certificate, HTTP is used over TCP/IP. Usually, a supplementary cryptographic layer, such as TLS, is also required.

8.3.1.1 UC 2.1: Trusted digital certificate download

The Cooperative Credential Management System (CCMS) secures communication of connected vehicles and road infrastructure by managing digital certificates, in line with European certificate management standards, regardless of the technology (ITS-G5 or C-V2X), based on bilateral trust between communication endpoints. CCMS provides primary mechanisms that support the management of trust across the system. The V2X Public Key Infrastructure (PKI) is the physical implementation of the CCMS policy.

One element of trustworthy C-ITS communication is to obtain certificates for secure communications from a trusted agent in a trustworthy way. During vehicle operations, they periodically request and receive

messages (certificates) and decide whether to trust those messages based on the PKI ruleset. The ruleset and access mechanism must comply with the standards and guidance of key European entities, including the EC, ETSI, Car2Car, and C-ROADS.

By using certificates, one can affirm that a communication endpoint has met stringent security requirements, which may include:

- Physical security requirements: Measures to protect the physical integrity of communication devices and infrastructure (host)
- Cybersecurity requirements: Specifications for host processors, operating systems, key management, and cryptographic protocols.
- Certification content: Processes to justify the entitlement of endpoints to particular contents in their certificates (e.g., an emergency responder's ability to perform emergency alert broadcast, etc), jurisdiction.

E2E message integrity is ensured by means of cryptographic techniques; however, the successful certificate update may be compromised (delayed or temporarily limited, or blocked) in a hostile environment.

CCMS secures the semantic correctness of the reception of trusted contents, but not the trusted path of the data access.

The focus of this scenario is on the periodic communication between OBU-equipped vehicles and the backend Public Key Infrastructure (PKI) to issue short-term anonymous credentials (pseudonyms) for the secure and privacy-preserving exchange of Cooperative Awareness Messages (CAMs).

8.3.1.2 UC 2.2: Over-The-Air (OTA) update of security sensitive communication devices

This scenario focuses on remote asset management and control of responders by transmitting special codes and commands and software/firmware updates from the backend in a V2N2V scenario. Today, over-the-air (OTA) updates for responder fleets are typically handled by vendor-specific backend systems that push software or firmware packages to vehicles over standard IP-based communication channels. Solutions such as RAUC, Mender, or custom-built update frameworks manage the packaging, verification, and installation process on the device side. These systems generally rely on the public internet or cellular networks using standard transport protocols (typically HTTPS over TCP) and do not perform any network-level manipulation of the traffic. There is no traffic steering, path selection based on trust level, or adaptation to varying network conditions beyond what the underlying transport protocols inherently provide (e.g., TCP retransmissions). Update delivery is usually best-effort, meaning that failures or interruptions are retried at the application layer, and the system must wait for the next connectivity window to resume. There is limited integration with PKI or security orchestration beyond verifying digital signatures of the update payload, and no mechanism for routing updates through specific trusted nodes or dynamically reconfiguring paths in case of partial network outages. As a result, while current OTA solutions are functional, they do not exploit network-level intelligence to optimize delivery performance, guarantee timeliness, or adapt to degraded trust conditions — all of which are in scope for CASTOR.

8.3.2 System Model and Communication

The system architecture for the emergency responder use case builds on established V2X entities and services, integrating vehicles, backend systems, and supporting network infrastructure. Core components include responder vehicles equipped with on-board units (OBUs), backend servers for OTA update

management, and the Public Key Infrastructure (PKI) providing enrollment, issuance, and revocation services. A V2X service provider's backend gateway to OEM servers and other supplementary services is the V2X Application Server, supporting communication based on V2N and V2N2V interactions. These interactions typically use cellular or broadband IP connectivity, secured through TLS-based protocols for OTA delivery and certificate renewal. Together, these components form the basis for secure software lifecycle management and resilient identity provisioning, enabling emergency responders to maintain operational trustworthiness in dynamic and potentially degraded environments. In Figure 8.17, we summarize these key system components and relations, described in more detail in the following subsections.

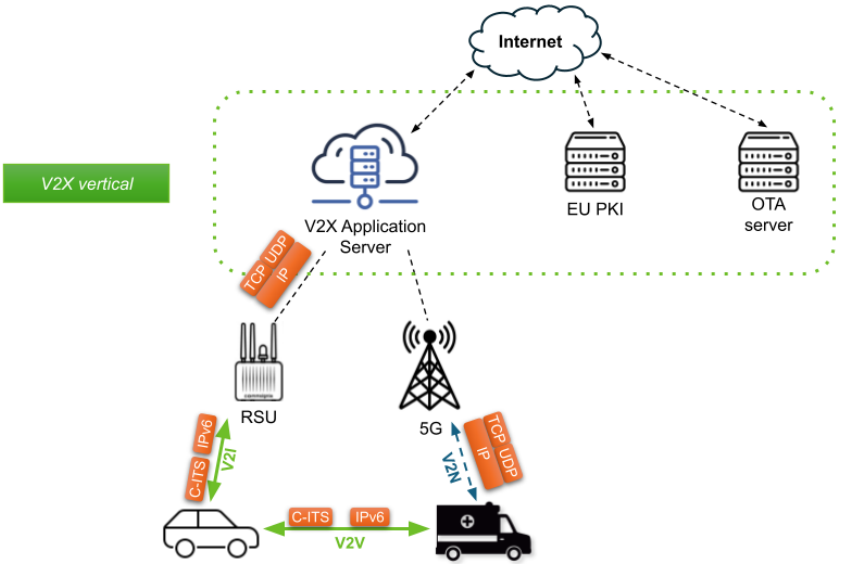

Figure 8.17: End-to-end communication topology of vehicles and C-ITS service access

8.3.2.1 System components

The typical system components of the described V2X use cases are the following (focusing on the UC2 perspective):

- **Vehicle On-board Unit (OBU) -** The OBU is the in-vehicle communication and processing platform that enables V2X services and secure connectivity to backend systems. It typically integrates one or more communication modems to support cellular and short-range (ITS-G5 and/or C-V2X) connectivity, a processing environment for running V2X applications, secure storage for cryptographic material, and interfaces to the vehicle's internal systems (e.g., via CAN bus). In the emergency responder use case, the OBU is responsible for receiving and validating OTA updates, managing PKI certificates, executing secure communication via the V2X Application Server, and ensuring that only authenticated and authorized commands affect the vehicle. It serves as the trusted execution point for network and security functions inside the vehicle.
- **5G network -** 5G networks act as a key enabler for enhanced V2N2V communications by combining high throughput, low latency, and high reliability with programmable interfaces for service-aware networking. Unlike legacy systems, 5G not only improves communication performance but also exposes APIs for advanced functionality such as Quality on Demand, traffic steering, and network slicing. This combination allows emergency responder services to obtain both the capacity and flexibility needed for mission-critical operations. Typical V2N use cases include backend-assisted cooperative awareness, collective perception, and traffic efficiency services. In the emergency responder scenario, 5G also provides the secure and timely connectivity required for OTA software updates and PKI certificate renewals, ensuring vehicles remain trustworthy and operational while in the field.

- **V2X Application Server (AS) -** The V2X Application Server acts as the central interface between vehicles and backend services. It mediates secure and optimized V2N2V communications for cellular-enabled clients, supporting enhanced V2N message routing mechanisms and broadening the horizon for safety-enhancing services (e.g., by connecting pedestrians via cellular). With tighter integration to 5G core networks, it can manage advanced 5G features for clients, such as Quality-on-Demand via available 5G APIs. Additionally, the V2X AS can work as an application gateway, interfacing with different application domains and services. Specifically for UC2:
  - It is the primary application gateway for V2X clients to the CASTOR domain, handling service registration and application-level status monitoring.
  - It acts as the vehicle's interface to the EU PKI trust infrastructure. Through secure V2N communications, it mediates interactions with the PKI by handling certificate enrollment requests, renewals, and revocation checks on behalf of the responder fleet.
  - It serves as the interface between vehicles and the OTA server, mediating update requests, secure delivery, and status reporting over V2N communication channels.
- **EU PKI -** In the V2X ecosystem, the EU PKI provides the trust anchor for secure communication by managing the lifecycle of digital certificates used for authentication, authorization, and message integrity. Its functions include the issuance of enrollment and authorization certificates, renewal of expiring credentials, and distribution of revocation information. By enforcing common security policies and cryptographic standards, the EU PKI ensures interoperability and trustworthiness across manufacturers, operators, and national domains.

  The interface with the V2X AS abstracts the complexity of the PKI from the vehicle side and ensures that updates to regulatory requirements or cryptographic policies can be applied centrally. Importantly, any national or regional regulatory entity—such as a country-specific Security Credential Management System (SCMS) or trust authority—is reached through the same V2X Application Server interface, ensuring a uniform communication path and compliance across jurisdictions.
- **OTA server -** The OTA server is responsible for managing the lifecycle of software and firmware updates for the vehicle fleet, including packaging, version control, integrity protection, and secure distribution. It ensures that updates are delivered reliably and verified before installation, maintaining the operational readiness and trustworthiness of emergency responder vehicles.

  Through the V2X AS as an interface, vehicles can receive updates from the OTA server without needing to directly manage backend protocols or security policies. In the same manner, any other OEM or third-party backend service can be connected to vehicles via the V2X Application Server, providing a uniform and secure integration point for additional services such as telematics, diagnostics, or fleet management.
- **Roadside Unit (RSU) -** While not being directly involved in UC2, RSUs are a general part of C-ITS systems. These devices are generally involved in safety use cases, performing as an interface to smart-intersection infrastructure (sensors, traffic lights). The RSUs usually have a wired connection and can interact with the V2X Application Server for additional functionality.

8.3.2.2 Communication protocols and interfaces

The foundation of Vehicle-to-Network (V2N) communication is based on the TCP/IP protocol stack. Communication between the On-Board Unit (OBU) and the Vehicle-to-Everything (V2X) Application Server can be achieved using RESTful HTTP APIs, which operate in a request/response format. Additionally, MQTT is essential for transmitting V2X messages and potentially some control information. Data may

also be streamed using UDP. For sensitive traffic, such as communications involving a Public Key Infrastructure (PKI) server, it is important to implement additional security measures, such as SSL/TLS encryption, to enhance safety.

The V2X Application Server acts as a gateway from various perspectives. Firstly, it serves as an intra-domain gateway, facilitating V2N2V communication between different ITS stations, such as OBUs and RSUs. Secondly, it serves as an inter-domain gateway for both the application and network layers. On the application layer, the V2X AS connects ITS stations to various services within the V2X domain, such as the PKI server and OEM servers (for example, to facilitate OTA updates). Additionally, on the network layer, it can directly interface with various networks through APIs. This includes 3GPP systems for Quality of Service (QoS) management and other advanced features of 5G cellular systems, as well as the CASTOR network, which ensures secure networking. Figure 8.18 illustrates the functional architecture of hybrid C-ITS networks, highlighting the multi-interface role of the V2X AS.

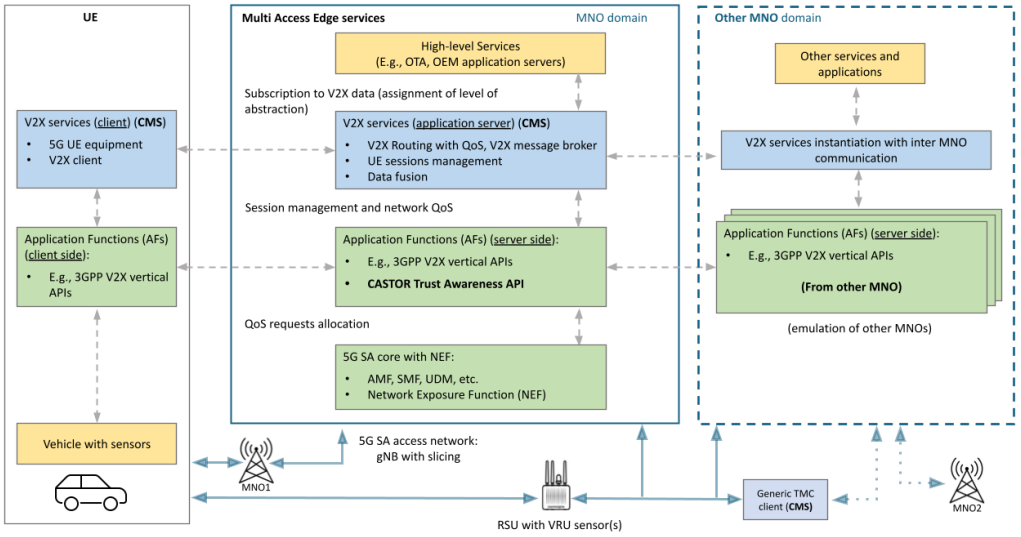

Figure 8.18: Functional architecture of hybrid C-ITS network adopted to CASTOR

8.3.3 Scenario Needs from CASTOR

The emergency responder scenario imposes stringent demands on the CASTOR network, requiring that over-the-air software updates and PKI certificate renewals be delivered reliably, securely, and on time. Because these operations are critical for maintaining the safety and operational readiness of emergency vehicles, the network must provide both robust performance and strong trust guarantees. This means ensuring that data is transmitted without tampering (integrity), is protected from unauthorized disclosure (confidentiality), and can be traced for forensic or compliance purposes (auditability). At the same time, the network must meet key performance requirements: it must deliver updates quickly (throughput), respond with minimal delay (latency), remain operational even under degraded conditions (availability), and scale to handle updates for an entire fleet of responders without service degradation (scalability). Together, these properties form the baseline for what the scenario expects from the CASTOR network. Table 8.13 contains the detailed description of the properties of interest for UC2.

UC2 Network/Trust Properties of Interest	
Name	**Description**
Integrity	Updates and PKI data must not be tampered with in transit.
Confidentiality	Sensitive data should not be visible to unauthorized parties to protect intellectual property and system configuration details.

Auditability/Traceability	All network transactions should be logged for forensic and compliance purposes.
Throughput/Bandwidth	The network must support a sufficient data rate to deliver updates within the required time frame. OTA updates may be large (hundreds of MB), and emergency responders may need them quickly to stay operational.
Latency	The time between request and (at least the initial/partial) response must be low enough to avoid service disruption. This is especially relevant for PKI certificate renewals, where delays could block communications.
Availability/Uptime	The network should remain reachable with high probability. Emergency responders must be able to receive updates or reach other services even during disasters or degraded infrastructure scenarios.
Scalability	The network must support simultaneous update sessions for many vehicles without congestion collapse. Large fleets may need updates at once (e.g., security patch rollout).

Table 8.13: Network and Trust properties as service-level objectives

8.3.4 To-be Reference Scenario 1: Connectivity to V2X PKI over cross-domain path provisioning

In the CASTOR-enabled environment, connectivity between the V2X Application Server and the PKI backend is no longer established through an unmanaged best-effort network, but through trust-orchestrated and policy-compliant paths. When a certificate renewal or enrollment request is initiated, CASTOR ensures that the connection to the PKI—whether hosted in the same or a different administrative domain—is established via network segments that meet the integrity, availability, and confidentiality guarantees defined in the corresponding SSLA.

Each CASTOR domain exposes its verified trust capabilities through the Trust Exposure Layer, enabling orchestrators in different domains to exchange trust semantics and dynamically compose cross-domain routes. This ensures that even when the PKI is managed by a separate operator, the V2X Application Server can rely on a continuously verified, end-to-end trusted communication channel.

Compared to the as-is scenario, where certificate exchanges rely on best-effort routing and unverified intermediaries, CASTOR introduces:

- Cross-domain trust negotiation and validation, ensuring that both endpoints and intermediate nodes meet defined SSLA criteria.
- Dynamic orchestration that continuously monitors path trust and performance, rerouting traffic if a degradation or policy violation occurs.
- Integrated logging and auditability, providing traceable evidence of compliance with security and service-level commitments.

Through these mechanisms, CASTOR transforms certificate management from a reactive, network-agnostic procedure into a proactively secured and orchestrated trust service, ensuring that pseudonym and authorization certificates are always issued and refreshed through managed, verified, and compliant paths.

8.3.5 To-be Reference Scenario 2: OTA Updates over trustworthy paths

In the as-is landscape, OTA updates for emergency responder vehicles are delivered over public or cellular networks using best-effort routing, leaving delivery time, integrity, and availability to chance. The

CASTOR-enabled “to-be” scenario replaces this uncertainty with trust-aware orchestration and SSLA-driven delivery control.

Before an update is initiated, CASTOR computes an optimal trusted path between the V2X Application Server and the OTA backend, based on current network trust scores, integrity attestations, and performance metrics such as available throughput and latency. Only nodes verified through remote attestation participate in the update delivery, and all transmissions are continuously monitored for SSLA compliance.

During the update process, the CASTOR framework:

- Detects any degradation in trust or performance along the active path.
- Notifies the V2X Application Server through the Trust Awareness API, allowing it to pause or safely abort the update if the path no longer satisfies the integrity threshold.
- Supports multiple SSLAs—for instance, a “main” SSLA with high throughput and integrity guarantees, and “fallback” SSLAs maintaining only integrity guarantees—to ensure that updates can continue, when possible, under reduced network conditions.

This approach upgrades OTA updates from a best-effort, transport-layer-dependent service to a network-assured and policy-compliant process. CASTOR thereby enables secure, verifiable, and timely software distribution, aligning update delivery with the operational reliability and safety requirements of emergency response fleets.

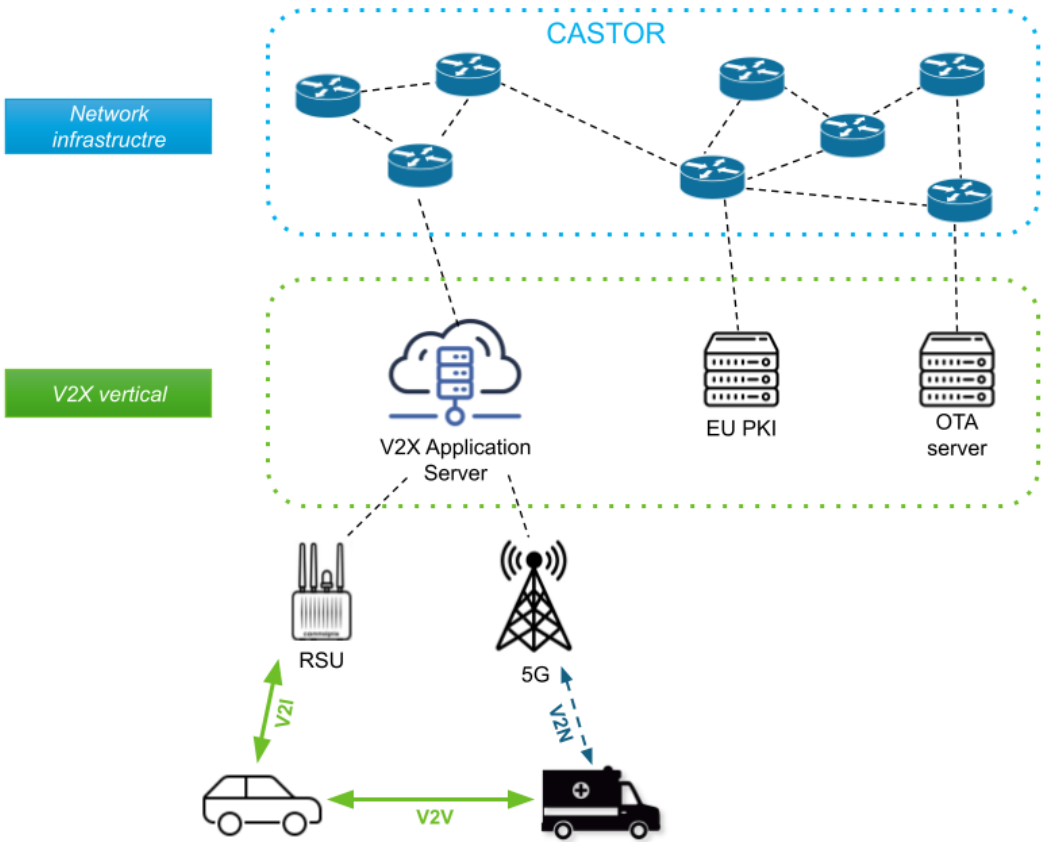

Figure 8.19: C-ITS service access enhanced by CASTOR

The architecture depicted in Figure 8.19 extends the baseline C-ITS topology by introducing a managed network infrastructure layer between the V2X Application Server and backend services (e.g., PKI and OTA servers). In contrast to the “as-is” scenario, where data flows traverse a fully transparent, best-effort network, the CASTOR-enabled “to-be” setup relies on a network that is aware of trust and service-level

semantics. While the detailed CASTOR components are abstracted in this view, the figure conceptually illustrates that communication paths are now provisioned through a partially managed and verifiable infrastructure rather than through an opaque internet segment. This shift allows the C-ITS services to operate over network segments that can enforce or monitor SSLA compliance, thereby enhancing the reliability and trustworthiness of both OTA and PKI interactions.

8.3.6 Reference Scenario User Stories

UC2.US1 V2X Services with different requirement sets

As a V2X service provider, I want to be able to maintain optimal SSLAs for the different types of traffic that can achieve specific **{network, trust}** requirements, so that I can ensure reliable service quality for all applications.

User Story Confirmation

V2X services differ widely in their requirements: some are highly time-sensitive (e.g., safety-critical hazard notifications), while others focus on reliable but less time-critical operations. Each of these service types imposes distinct expectations on the underlying communication system in terms of latency, throughput, availability, integrity, and trust. To ensure reliable service quality, these requirements must be supported independently, without one service type degrading the performance of another, and in parallel, since multiple applications may be active simultaneously. For example, both an OTA update process (UC1.US1a) and a PKI certificate renewal (UC1.US1b) require a trusted network for high integrity. An OTA update typically involves the download of a large file (compared to, e.g., a V2X message), meaning data throughput is one of the key requirements, while a certificate batch is much smaller in size.

The CASTOR framework addresses these challenges by enabling the definition, registration through service provisioning, and enforcement of multiple SLAs or security-aware SLAs (SSLAs) tailored to specific services. It provides mechanisms to select and maintain the most appropriate communication paths that satisfy both network performance and trust guarantees, even under degraded or changing conditions. By doing so, CASTOR helps service providers preserve optimal end-to-end quality for all V2X applications while mitigating the risks of contention, misrouting, or trust erosion.

UC2.US1a Secure update delivery based on trusted network paths

As a V2X service provider, I want to be able to request an optimal network path with **{high integrity and moderate throughput}** (that complies with the SSLAs) to the OTA server, so that I can ensure secure and reliable OTA updates for my clients.

User Story Confirmation

Communication between the OBU, the V2X Application Server, and the OTA server typically relies on secure, standards-based protocols such as HTTPS or MQTT over TLS. Control messages (e.g., update availability checks, status reporting) are exchanged between the OBU and the V2X AS, while update binaries and manifests are fetched from the OTA server.

Update packages are generally digitally signed to guarantee authenticity and integrity, following approaches used by frameworks such as RAUC, Mender, or Uptane. Transport-level encryption (e.g., TLS) ensures confidentiality during transmission, while the content itself may optionally be encrypted if required by the OEM or regulatory policy.

In conventional deployments, the network route between an OTA issuer (e.g., an OEM backend server) and the receiving client (e.g., an OBU) traverses a largely unmanaged and heterogeneous environment. Within such a landscape, the trustworthiness of intermediate nodes and links cannot be guaranteed,

leaving the transmission vulnerable to interception, manipulation, or redirection. A successful man-in-the-middle attack could lead to the delivery of maliciously modified update packages, potentially compromising the safety, reliability, and regulatory compliance of responder vehicles. Even when payloads are cryptographically signed, repeated delivery failures, denial-of-service attempts, or persistent corruption of the transport path can prevent timely patching and increase operational risks.

While TLS provides end-to-end confidentiality and integrity of the transmitted data, it does not ensure the trustworthiness or availability of the intermediate network path. Compromised or misconfigured nodes may still intercept, redirect, or degrade traffic, creating operational and security risks beyond the scope of transport encryption.

The CASTOR framework mitigates these vulnerabilities by enabling the establishment of trusted network paths before software update delivery. Instead of relying solely on best-effort routing, CASTOR integrates trust verification and path optimization into the orchestration process. This involves, for example:

- Remote attestation of network entities along candidate paths, ensuring that only verified and uncompromised nodes participate in transmission.
- Dynamic trust scoring of nodes and links, based on software/hardware integrity, historical reliability, or operator policies.
- Policy-based orchestration that selects a path exceeding a configurable trust threshold while still satisfying required performance characteristics such as throughput and latency.

By combining these mechanisms, CASTOR ensures that OTA updates are delivered over paths that are both trustworthy and performance-compliant, thereby reinforcing the reliability and security of the software lifecycle in emergency responder fleets.

User Story Workflow

Figure 8.20 captures the flow of action concerning UC2.US1a. The service registration and SSLA negotiation phase precedes the operational phase of the OTA service.

The OTA update procedure begins with the vehicle client (OBU) fetching any available updates from the OTA server (step 1). The V2X application server, being the gateway to external C-ITS and OEM services, acts as a proxy. This means all requests from the vehicle client are sent to the V2X Application Server, which then relays incoming requests to the OTA server (step 1.1). The server responses are also relayed back to the vehicle client (step 1.2). The client (OBU) evaluates the metadata and selects the correct package from the list of available update packages selected for download (step 2).

When the update sequence is requested by the client (step 3), the V2X Application Server fetches the current SSLA compliance (step 3.1). Depending on the status, two things can happen: the V2X server can deny the update sequence if the SSLA compliance is violated (step 3.2.1), or proceed and forward the download request to the OTA server (step 3.3). The OTA server then starts the data stream, and the data chunks are forwarded to the client (step 3.3.1) until the last data chunk is sent (step 3.4) and delivered (step 3.2.2). The client confirms the download (step 5) and can proceed with updating the firmware offline.

CASTOR KPIs

The CASTOR KPIs for UC2.US1a are summarized in Table 8.14.

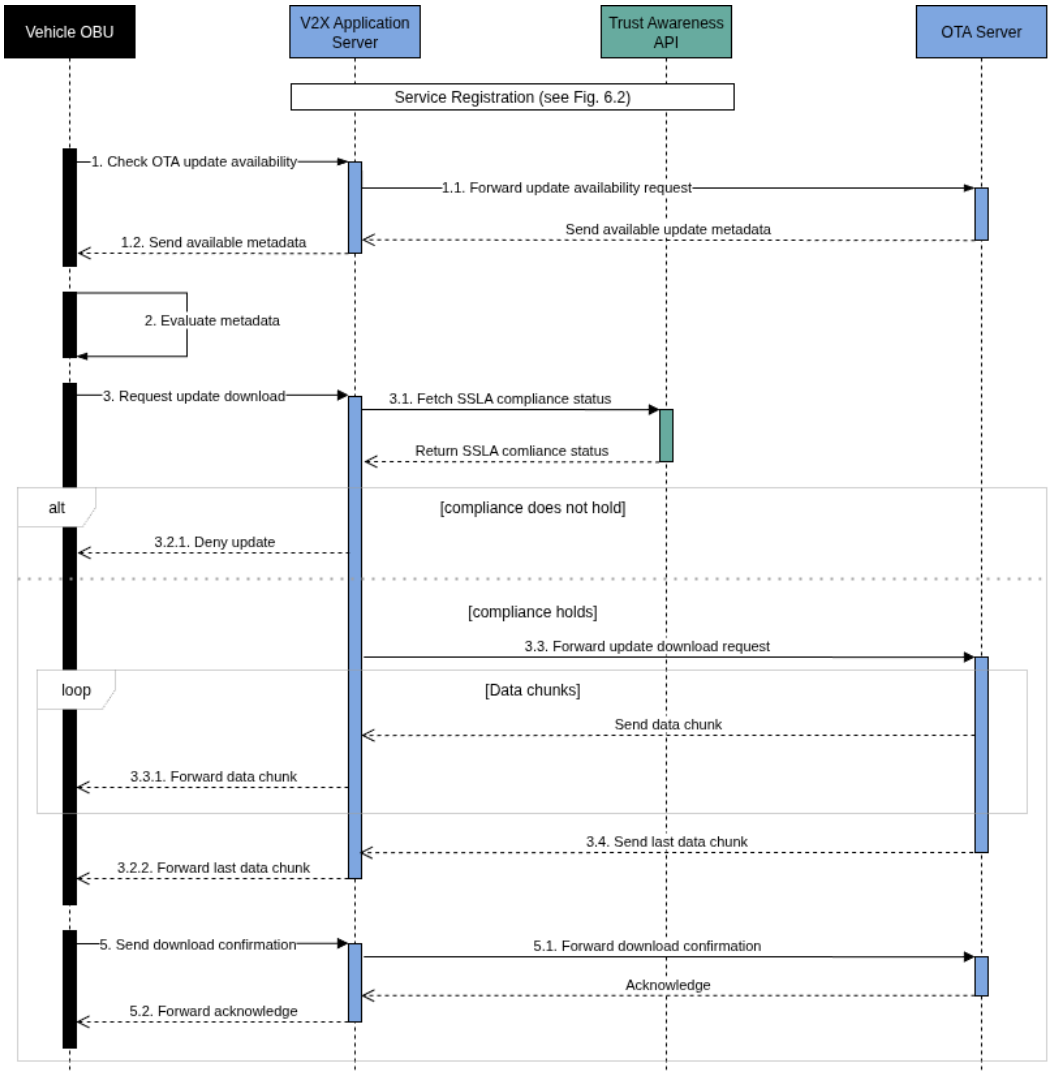

Figure 8.20: UC2.US1a Workflow

Table 8.14: CASTOR KPIs for User Story UC2.US1a

KPI	Definition	Target Value
Moderate throughput	The operation of CASTOR shall still allow for an adequate throughput for the firmware/software upgrade process to take place - it is the availability that takes a lesser role since such updates can also take place in a garage and not during the vehicle's journey. However, CASTOR aims to profile the impact of varying throughput levels on the operational profile of the other safety-critical operations. The size of a complete firmware upgrade typically ranges from 300 MB to over 1 GB, meaning the update should take between 30 and 180 seconds, depending on the size.	≥ 70Mbps
Moderate latency	The CASTOR operation should not induce significant latency to the OTA process. However, this does not refer to CASTOR possibly reducing the priority of such a transmission for allowing the continuity and timly responsiveness of the other safety-critical application. The goal here is to measure the explicit latency that the CASTOR pipeline when xecuted as an atomic process can induce. In this case, there are no strict real-time requirements (unlike in V2X messaging).	≤ 60s

UC2.US1b Secure certificate renewal based on trusted network paths

As a V2X service provider, I want to be able to request an optimal network path with **{high integrity and high availability}** (that complies with the SSLAs) to a PKI server, so that I can ensure secure and reliable certificate updates for my clients.

User Story Confirmation

Certificate renewal is critical for maintaining continuous trust in V2X communications, as vehicles rely on valid credentials to authenticate messages and participate in cooperative safety services. In the C-ITS Certificate Management Architecture (ETSI TS 102 940), this process involves two main phases: enrollment, during which the vehicle or on-board unit (OBU) obtains an initial enrollment certificate that binds its identity to the PKI, and authorization, where the OBU uses its enrollment credentials to request and receive short-lived authorization (pseudonym) certificates. These certificates are rotated periodically to preserve privacy and are required for signing V2X messages on the road.

Both phases depend on a secure and reliable exchange of messages with PKI entities such as the Enrollment Authority and Authorization Authority. While the certificate payloads are small, they are sensitive to latency, trust, and availability constraints—any delay or failure can result in expired credentials or a temporary loss of secure communications. In conventional networks, such requests may traverse untrusted or unstable paths, leading to potential delays, redirection, or Denial-of-Service (DoS).

The CASTOR framework enhances this process by ensuring that enrollment and authorization traffic is routed over trusted, policy-compliant network paths. Through mechanisms such as remote attestation, dynamic trust and availability scoring, and cross-domain orchestration, CASTOR complements the PKI layer by protecting the communication channel itself. This guarantees that C-ITS credential management processes can execute reliably and securely, even under degraded network conditions, thereby maintaining operational trust for emergency responder fleets.

CASTOR KPIs

The CASTOR KPIs for UC2.US1b are summarized in Table 8.15.

Table 8.15: CASTOR KPIs for User Story UC2.US1b

KPI	Definition	Target Value
Low throughput	The operation of CASTOR shall allow for an adequate throughput for the PKI certificate download process to take place - especially considering the short-term certificates that may need to be downloaded in specific “zones”. Certificate batches vary in size, around 3-5 MB per batch. The download does not need to be immediate (sub-second).	$\sim$1 Mbps
Strict latency	The CASTOR operation shall allow for the PKI certificate download process to be executed within the time constraints posed by the “update zones”. The transmission of a data packet should not take longer than the specified value.	$\leq$ 1s
SSLA compliance notification latency overhead	Latency overhead imposed by consuming notifications/events from the CASTOR Trust Awareness API (e.g., fetching SSLA compliance status) when monitoring/controlling the PKI service should be below the specified value.	$\leq$ 10%

UC2.US2 Update adapts to degraded trust

As a V2X service provider, I want to be notified if the active SSLA compliance is compromised, so that transmission over an untrusted, potentially malicious, or underperforming network path can be evaded.

User Story Confirmation

Even after selecting an initially trusted network path from the V2X Application Server to the OTA server, conditions along that path may change during the update process. Nodes or links that were previously trusted may become compromised, misconfigured, or otherwise fail to meet the integrity or performance requirements defined in the active SSLA. Any violation of high-integrity guarantees, in particular, could threaten the security of the update transmission.

The CASTOR framework continuously monitors the trustworthiness and performance of network entities along the selected path. If the monitored path falls below the thresholds defined in the **Main SSLA** (e.g., high-integrity and minimum throughput), or if the high-integrity requirement is violated at any point, the orchestrator generates a notification via the **Trust Awareness API** to the V2X Application Server. The V2X AS integrates this notification into the update process and commands the OBU to either continue the update if a compliant Fallback SSLA is available or abort the transfer if integrity is compromised.

By supporting multiple SSLAs of varying strictness, CASTOR enables adaptive operation: updates can proceed under reduced throughput conditions when only a Fallback SSLA is met, while strict high-integrity requirements are always enforced. This ensures that OTA updates remain secure and reliable, with the V2X AS enforcing aborts whenever trust violations are detected, preserving the operational safety of emergency responder vehicles.

User Story Workflow

Figure 8.21 shows an example workflow for adapting to violated SSLAs during the OTA update process. In this example, the **Main SSLA** specifies high integrity (trust-related requirement) and specific minimum throughput value (network performance-related requirement, see Table 8.14) as requirements to be kept. The **Fallback SSLA** specifies high-integrity only. In this case, high-integrity is a must-have requirement, while temporary violation of the minimum throughput requirement can be tolerable for the OTA update process. Service registration and the negotiation of the Main and Fallback SSLAs precede the OTA update process.

In this user story, the OTA update process is already in progress (see UC2.US1a), and the data chunks are being downloaded from the OTA server. The flow of actions starts with the Main SSLA being violated, after which the Trust Awareness API sends a notification to the V2X AS (step 1). If the Main SSLA is violated, the Fallback SSLA can immediately take its place (according to initial service registration and negotiations). The Trust Awareness API notifies the V2X AS about Fallback SSLA compliance (steps 2.1 and 2.2). The V2X AS can respond with different strategies based on the compliance of the Fallback SSLA(s), e.g., by stopping the OTA update if integrity cannot be ensured (step 3). If the update is aborted, the client (OBU) will immediately abort the process and revert to a stable state (step 4). The client also sends a request to the OTA server (step 5) to stop/cancel the whole update process.

The V2X AS can optionally fetch extra details from the Trust Exposure Layer (step 6) for further investigation. A renegotiation of SSLAs can follow if none of the previously negotiated SSLAs are compliant, potentially using the fetched details during the renegotiation.

CASTOR KPIs

The CASTOR KPIs for UC2.US.2 regarding the CASTOR framework are summarized in Table 8.16.

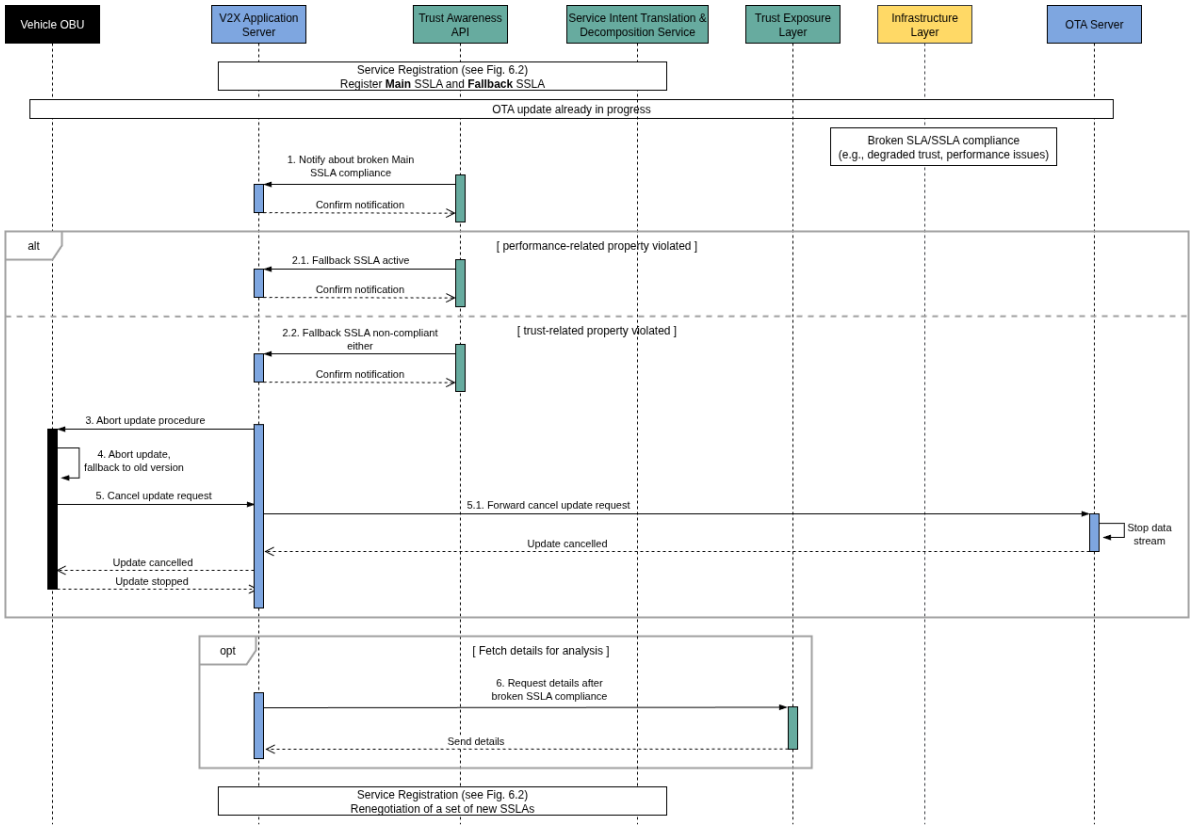

Figure 8.21: UC2.US.2 Workflow

Table 8.16: CASTOR KPIs for user story UC2.US2

KPI	Definition	Target Value
Routing path alternative establishment	The CASTOR framework should find a routing path alternative (without optimization) to enforce the Fallback SSLA if the Main SSLA is violated.	≤ 50 ms (see requirement TE.R.1)
SSLA compliance notification	The CASTOR Trust Awareness API should notify the V2X AS about Fallback SSLA compliance after the Main SSLA is violated before the OTA process ends.	≤ 1 s
Violated SSLA property	The CASTOR Trust Awareness API can reliably inform the V2X Services about the violated SSLA properties.	TRUE
Throughput	The relative amount of time with no throughput guarantee compared to the total time to update.	$\leq 30\%$

UC2.US3 Secure Routing to PKI Server in a different domain

As a V2X service provider, I want to be able to establish trusted communication with a PKI in a different network domain, so that client certificates are always delivered via managed and trusted networks.

User Story Confirmation

In this scenario, a PKI server resides in a different CASTOR domain, potentially managed by a separate orchestrator. Establishing trusted communication across domains is critical to ensure that client certificates are delivered securely and reliably, without exposing the transmission to untrusted or misconfigured network elements. Without proper cross-domain coordination, certificate updates may be delayed, blocked, or compromised, resulting in service disruption or a loss of trust in the V2X system.

The CASTOR framework adds value by enabling **cross-domain trust provisioning**. Each domain exposes its trust capabilities and network guarantees to other domains via the **Trust Exposure Layer**, allowing orchestrators to discover and interact with services across administrative boundaries while respecting SSLA requirements. This mechanism ensures that network paths connecting the V2X Appli-

cation Server to external PKI servers remain compliant with defined trust and performance parameters, even when traversing another domain's infrastructure.

By leveraging cross-domain orchestration and the Trust Exposure Layer, CASTOR guarantees that certificate delivery remains secure, verifiable, and compliant, extending the integrity, availability, and trust assurances of single-domain operations to multi-domain scenarios. This enables emergency responder fleets to obtain certificates reliably from PKI servers in other domains while preserving end-to-end trust.

User Story Workflow

Figure 8.22 shows an example workflow in the cross-domain PKI scenario. Service registration in the domain, the negotiation of the SSLAs, and the exchange of trust capabilities between the two domains precede the PKI certificate renewal process. The underlying CASTOR procedures are hidden in the workflow of this scenario; only the procedures experienced/influenced by the V2X service appear.

When the client (OBU) is close to having expired certificates, the update process begins. The client compiles the appropriate HTTP request (step 1), and it signs the authorization certificate used for the certificate renewal process (step 2). The request with the signed authorization cert is sent to the V2X AS, acting as a proxy to the PKI (step 3). The V2X AS fetches the SSLA status from the Trust Awareness API of its domain (step 3.1). Depending on the SSLA compliance, the certificate update process can be denied (step 3.2) or it can go on by forwarding the request to the PKI server residing in the second domain (step 3.3). The PKI server performs the authorization check (step 3.4) and provides a new set of certificates to the client. The client can decrypt and install the new certificates at the end of the process (step 4).

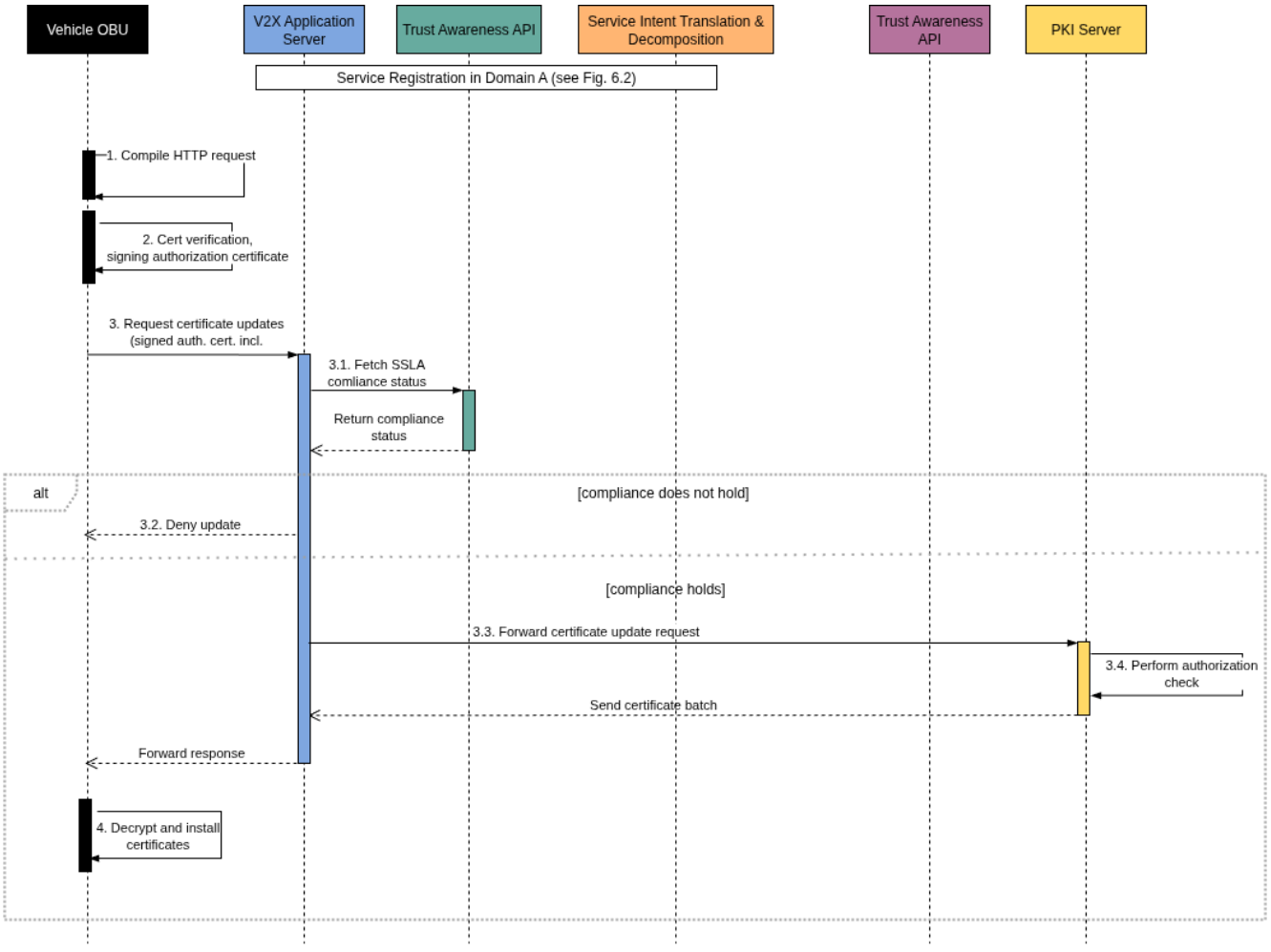

Figure 8.22: UC2.US3 Workflow

CASTOR KPIs

The general KPIs for UC2.US3 regarding the CASTOR framework are summarized in Table 8.17.

Table 8.17: CASTOR KPIs for user story UC2.US3

KPI	Definition	Target Value
Availability	Minimal impact on service availability	Acceptable margins in non-critical properties that could compromise the availability/integrity properties.
Cross-domain latency	The additional latency imposed by the cross-domain service provisioning and policy enforcement should not exceed the specified value.	$\leq 20\%$ increase compared to not using CASTOR

8.4 Priority-based Trusted Messaging & Scalable Performance for CCAM Applications

8.4.1 "As-is" Scenario

European Telecommunications Standards Institute (ETSI) through its working groups develops and harmonizes standards for Intelligent Transportation Systems (ITS), with the aim of providing guidelines and requirements regarding services related to modes of transport and traffic management to enhance safety, efficiency, and mobility on roads and highways. The technical specifications created by ETSI describe different ITS components, such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication systems. Moreover, ETSI defines the real-time data exchange used for different services through different message standards, such as the Cooperative Awareness Message (CAM) [5] and the Decentralized Environmental Notification Message (DENM) [4]. These messages help vehicles and infrastructure share critical information about traffic, hazards, and road conditions. The cooperative awareness message (CAM) provides information to nearby vehicles and roadside units (RSU) about vehicle speed, position (latitude, longitude), direction, and acceleration. These messages are transmitted by the onboard unit (OBU) with a period that depends on the dynamics of the vehicle (e.g., increasing frequency during rapid maneuvers) as well as the radio channel load [5]. Decentralized Environmental Notification Messages (DENM) are event-triggered messages used to notify vehicles and infrastructure about specific road events or hazards. Due to their nature and importance, DENMs are time-critical in order to provide timely warning and response.

Table 8.18 presents several events that trigger the generation and transmission of DENM messages.

8.4.2 System Model

The system architecture for the priority-based trusted messaging applications relies on entities and sensors (e.g., cameras, radars) that are linked and coordinated across different domains. The architecture and its components are illustrated in Figure 8.23.

- **Detection sensors**, such as cameras and radars, are core elements of the architecture and provide real-time data on traffic flow, accidents, and vulnerable road users.
- **GEO Service Provider (GSP)** acts as a central application that collects data from sensors and vehicles. These data are processed and transmitted through secure, policy-controlled paths to other entities, including traffic operators, local emergency responders, and central emergency responders.

Table 8.18: Cause description triggering generation and transmission of DENM message [4]

Cause triggering DENM messages	Details
Traffic	Information about traffic congestion
Accident	Information about accident with possibility of assistance requested (e-call)
Roadworks	Information about road works (e.g., major roadworks, road marking work, street cleaning, etc.)
Adverse conditions	Information about adverse conditions (e.g., flooding, danger of avalanches, landslips, etc.)
Hazards on the road	Information about hazards on the road (e.g., rock falls, earthquake damage, fallen trees, stationary vehicle, etc.)
Wrong way driving	Information about wrong carriageway (e.g., vehicle driving in wrong lane, Vehicle driving in wrong driving direction)
Rescue and recovery work in progress	Information about rescue and recovery (e.g., emergency vehicles, rescue helicopter landing, police activity ongoing, medical emergency ongoing, etc.)
Vehicle breakdown	Information about vehicle breakdown (e.g., broken-down vehicle on fire, broken-down unlit vehicle, lack of fuel, lack of battery, etc.)
Human problem	Information about human health problems (e.g., glycaemia problem, heart problem)
Stationary vehicle	Information about stationary vehicle (e.g., human problem, vehicle breakdown, post-crash, etc.)
Emergency vehicle approaching	Information about emergency vehicle approaching (e.g., emergency vehicle approaching, prioritized vehicle approaching)
Collision risk	Information about collision risk (e.g., longitudinal collision risk, crossing collision risk, lateral collision risk, etc.)
Dangerous situation	Information about dangerous situation (e.g., emergency electronic brake lights, pre-crash system activated, collision risk warning activated, etc.)

- **Traffic operators (TO)** receive congestion alerts to optimize flow management.
- **Local emergency responders (LER)** are provided with authenticated accident notifications, enriched with precise location and severity indicators, so that fast, accurate, and timely emergency measures can be deployed.
- **Central emergency responders (CER)** are alerted in the case of high-impact incidents through trusted cross-domain channels. Based on these alerts, central emergency responders can enable a properly scaled and coordinated response.

8.4.3 Scenario Needs from CASTOR

At the infrastructure level, CAM and DENM messages are processed locally by GEO Service Provider to provide alerts to road users about traffic jams, road hazards, accidents, and to notify emergency services when necessary.

For more critical situations, such as multi-vehicle collisions, local natural disasters, or regional road blockages, both critical and non-critical events are centrally managed, for example, by the Emergency Services or National Highways.

Figure 8.23 illustrates the flow of CAM and DENM messages in case of an accident. These messages are sent over the 5G network to a local GEO Service Provider system for traffic monitoring and emergency service alerts. Major traffic events are transmitted via the internet to a centralized service provider. However, due to the limitations of internet-based communication, messages related to major traffic events could be delayed, altered, or even lost. In the case of non-critical events, these issues typically have minimal impact. However, for critical events, these problems may lead to delayed responses from authorities, potentially resulting in fatalities that could have been prevented with faster intervention. Through CASTOR's framework, messages associated to critical events will be routed over trusted paths, so that the limitations of internet-based communication will be mitigated.

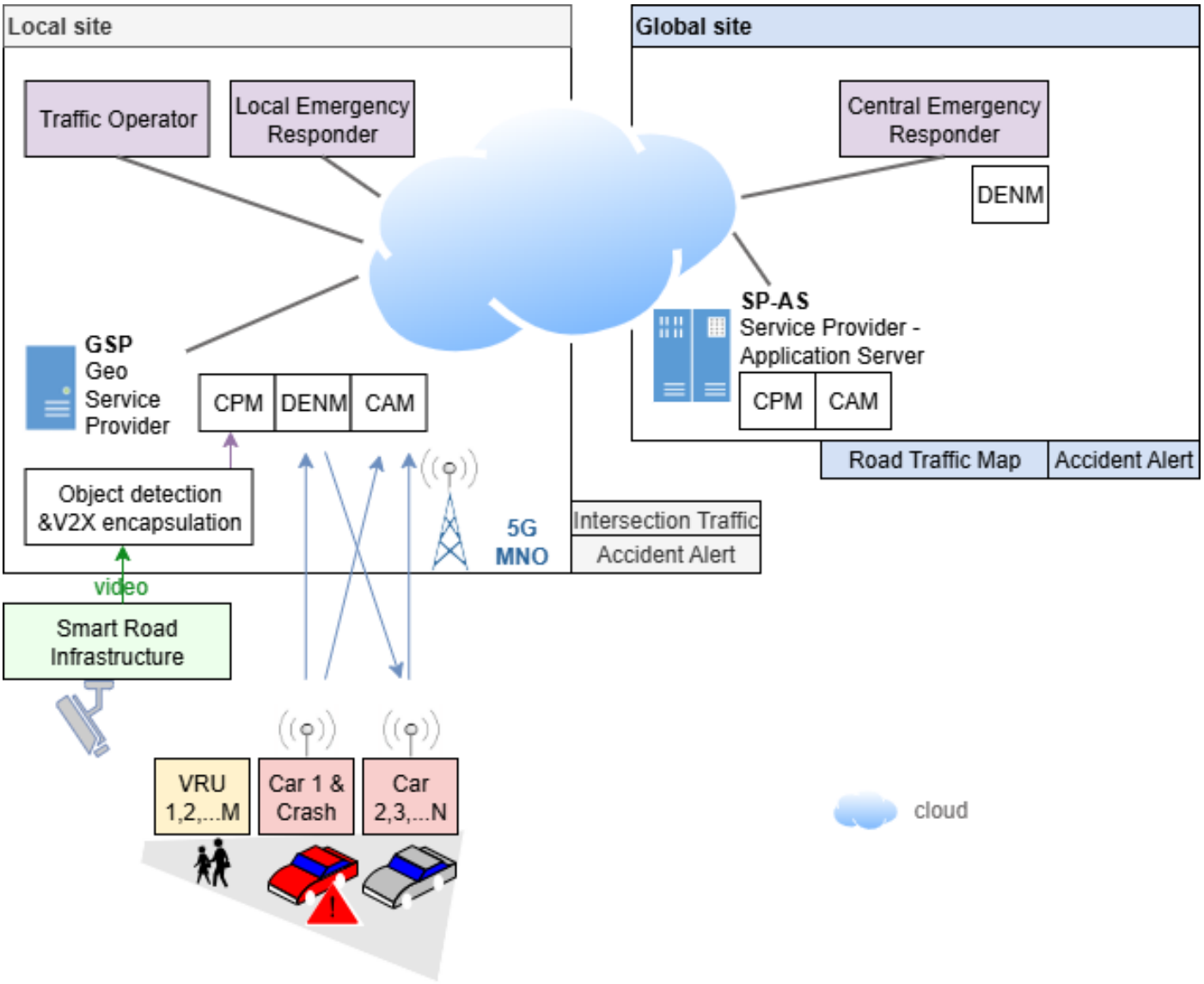

Figure 8.23: Illustration of the network path for CAM and DENM messages

The routing of the messages will be handled by the CASTOR solution which will determine the necessary path for DENM messages.

8.4.4 Improved model using CASTOR's framework

For the #3 use case in CASTOR's framework, Technical University of Iasi (TUIASI) will develop a virtualized OBU that will simulate the generation of CAM and DENM messages. These messages will be routed over the physical 5G infrastructure to a GSP system in Iași. Furthermore, major traffic events will be simulated and sent through Orange's infrastructure to a centralized application server. The CASTOR framework, which operates on top of Orange's infrastructure, will handle the discovery of trusted paths and will route messages associated with major critical events along those paths, as it is shown in Figure 8.24.

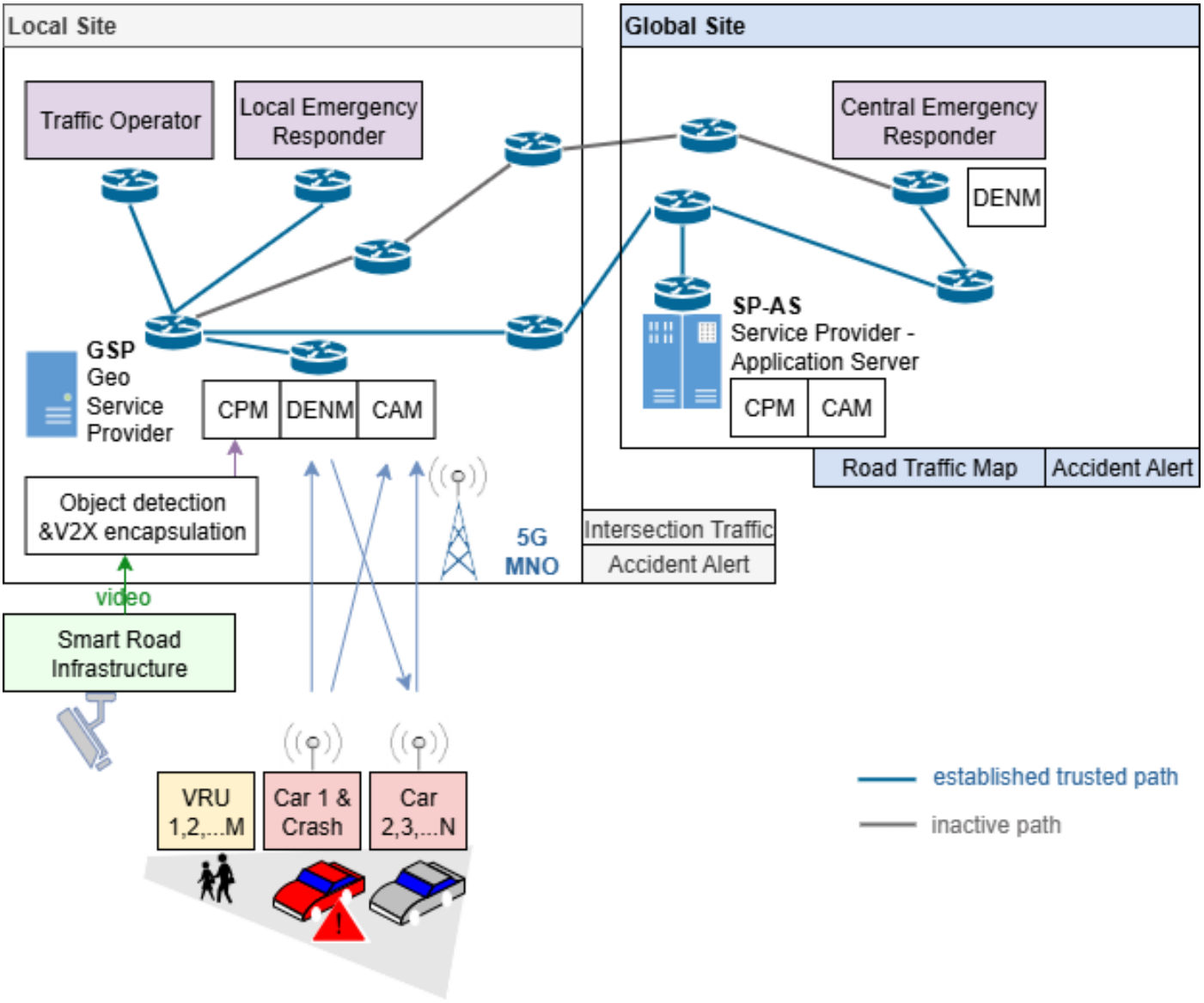

Figure 8.24: Message routing using CASTOR's framework

In the context of #3 use case, three applications will be tested. In the first application, a GEO service provider (GSP) uses CAM messages from a specific area to create an environment model. This model is transmitted to a Traffic Operator (TO) to support traffic statistics and traffic management strategies. In the second application, based on crash alerts received in DENM messages, the GSP informs a Local Emergency Responder (LER) so that the LER can dispatch emergency vehicles. In case the crash is

severe (e.g., a multi-vehicle crash), the GSP informs not only the LER but also a Central Emergency Responder (CER), so that multiple emergency services can be dispatched. In the third application, the GSP processes raw data received from detection nodes (e.g., cameras, radars) deployed at an intersection to detect Vulnerable Road Users (VRUs). Upon detection of a VRU, the GSP alerts vehicles intending to cross the intersection.

The network and security requirements for Trusted Path Routing corresponding to the three applications are presented in Table 8.19.

UC3 Network/Trust Properties of Interest	
Name	**Description**
Integrity	Traffic information and alerts about crash accidents must not be tampered with in transit. The raw data sent by the detection nodes must not be tampered with in transit.
Latency	Traffic information must not be exchanged with great delay, otherwise the traffic prediction will not be accurate. Crash alerts must reach emergency responders with little latency. Detection nodes must communicate their data with very little delay.
Throughput/Bandwidth	The network must support a sufficient data rate to exchange traffic information or raw data.
Confidentiality	Sensitive data shall not be disclosed to unauthorized parties to protect vehicle identities and owners.
Availability	The entities of interest for receiving data shall remain reachable with high probability.

Table 8.19: Network and Trust properties as service-level objectives

8.4.5 Reference Scenario User Stories

UC3.US GEO service provider

As a GEO service provider, I want to be able to communicate with different entities over optimal trusted paths that satisfy specific {**network, trust**} requirements.

User Story Confirmation

The GEO service provider (GSP) interacts with the CASTOR framework to establish SLAs for network and trust specifications. This ensures that GSP can exchange data with different entities in a trusted environment. Moreover, GSP can negotiate different service SLAs in function of application. For example, an SLA with medium availability, integrity, latency, high throughput, and confidentiality is negotiated for the traffic operator application (more details in the following). More strict requirements are necessary for the emergency responder application or the vehicle driver / vulnerable road user application. Another capability of the CASTOR which will be shown is related to the exchange of information across domains in a trusted manner. This capability is detailed in user story US.2 Emergency Responder.

UC3.US1 Traffic operator

As a GEO service provider, I want to have a trusted path {**medium availability, medium integrity, confidentiality, medium latency and high throughput**} through which to be able to provide to a Traffic Operator real-time, accurate, and trustworthy information on potential traffic congestion situations, so that it can implement effective traffic management strategies to reduce delays and improve traffic flow. I want the Traffic Operator to be notified if the selected path's trust level drops below a threshold, such that proper measures can be taken in case the situation persists.

User Story Confirmation

In this user story, the GEO service provider initializes the request to establish a trusted path toward the Traffic Operator (TO) as explained in Figure 6.3, which determines the available trust capabilities, calculates a trusted path, and works with the network management to configure nodes accordingly.

The GEO service provider aggregates multiple CAM messages received from vehicles located in a local region (e.g., city, intersection), and creates an environmental model which is sent to Traffic Operator. This communication is realized through a secure message-oriented middleware. The GEO publishes periodic traffic state updates as structured data to a dedicated channel or endpoint consumed by the Traffic Operator's system. CASTOR dynamically evaluates the trust environment of network elements and communication links. Due to variations in node behaviour or link instability, CASTOR recalculates path trust, triggers reconfiguration, or notifies the GEO of reduced trust levels. The system model interacts with CASTOR in a feedback loop: congestion computations rely on traffic data transmitted through the trusted path, CASTOR continuously monitors and reassesses the trust level, and upon threshold breaches, notifies the GEO, who informs TO and may choose mitigation strategies (e.g., switch to a degraded mode, rely on backup paths, or trigger revalidation). Notifications related to the decrease in trust levels can be indicated to an operator via a dashboard. By monitoring and reassessing, CASTOR acts as a trust orchestration layer, ensuring that the TO not only receives reliable real-time information but also has visibility into the integrity and resilience of the communication path itself.

User Story Workflow

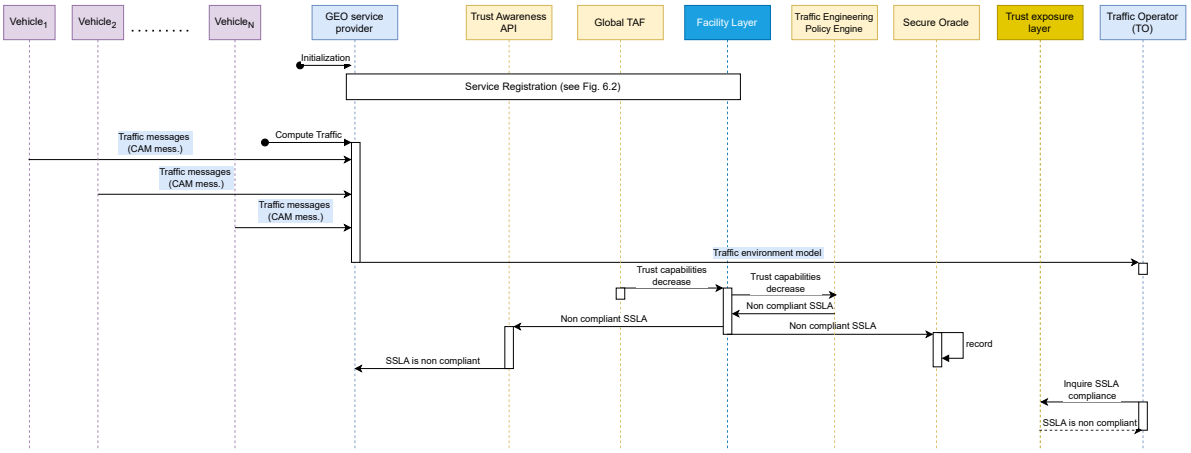

Figure 8.25: UC3.US1 Workflow

UC3.US2 Emergency Responder

As a GEO service provider, I want to have a trusted path **{very high availability, very high integrity, and low latency}** through which to be able to provide to a Local Emergency Responder (nearby), authenticated accident notifications with precise location, severity assessment, and trust level indication, so that it can dispatch appropriate resources and coordinate effective emergency response, without false alarms. In case of severe accidents, I want to have a trusted, cross-domain path **{very high availability, very high integrity, and medium latency}** to a Central Emergency Responder (located in a different domain), that can coordinate a centralized response to the accident, properly scaled to its severity level.

User Story Confirmation

In this user story, the assumption is that trusted paths are already determined and configured by CASTOR, ensuring that both local and cross-domain emergency entities are connected to the GEO service provider (GSP) before any accident event occurs. Vehicles generate authenticated DENM messages,

authentication which is based on security mechanisms presented in [8]. The messages contain accident details, which the GEO service provider aggregates, verifies, and enriches with severity assessments. Furthermore, when an accident is detected, the GEO Service Provider computes a severity score based on different attributes or information, and if the score remains below the escalation threshold, it transmits an authenticated DENM to the Local Emergency Responder via the pre-established trusted path. If severity meets or exceeds the threshold (e.g., a multi-vehicle crash or an accident involving a bus), the GSP sends the accident notification to the Local Emergency Responder and also to Central Emergency Responder (CER)—typically in another domain. The alerts can be displayed to operators at LER or CER on a dashboard, for example. The establishment of cross-domain paths allowing the transmission of such alerts over trusted paths follows the service negotiation and registration processes of Section 6.2.1. After receiving the notification, CER initiates scaled actions by mobilizing emergency services (e.g., activate a Code Red intervention plan, dispatch an air ambulance).

User Story Workflow

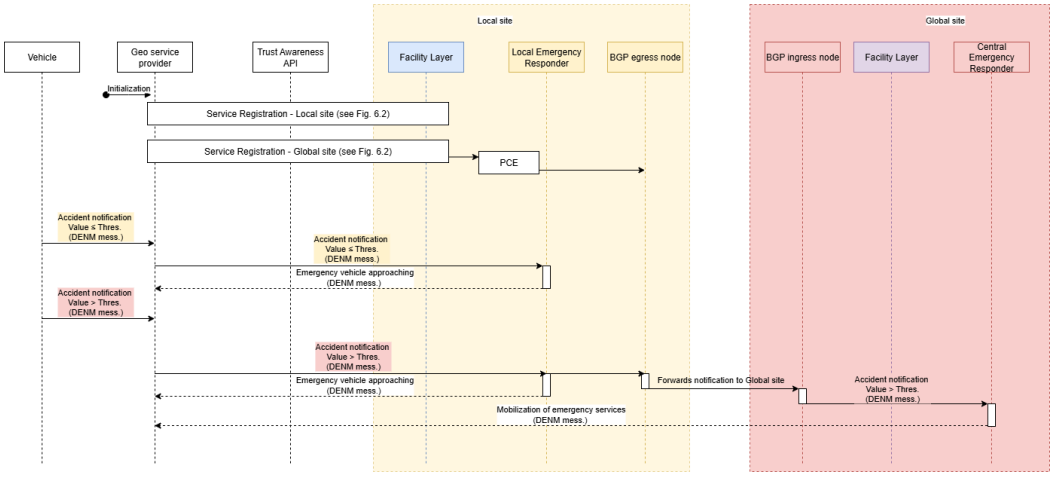

Figure 8.26: UC3.US2 Workflow

UC3.US3 Vehicle driver

As a GEO service provider, I want to have trusted paths **{high availability, high integrity, high throughput, and very low latency}** to detection sensors (e.g., cameras, radars), so that I can receive in real-time trusted information. Based on this information, I will alert in real-time connected Vehicle Drivers navigating a dense traffic intersection about vulnerable road users (VRUs) that may intersect their trajectory, so that they can take timely action and avoid a potential collision. I want to be informed in case the selected path's trust level drops below a threshold, such that I can let the Vehicle Drivers be properly informed.

User Story Confirmation

In this user story, it is assumed that through the interaction between GEO service provider and CASTOR, CASTOR establishes trusted paths between the GEO service provider and detection nodes (e.g., cameras at intersections). Vehicles broadcast CAM messages that contain information about their direction. Processing this information and real-time video streams from cameras, the GEO service provider identifies potential collision with Vulnerable Road User (VRU) and transmits an alert to the vehicle. CASTOR monitors the trustworthiness of the configured path, calculating trust levels that reflect network node integrity, data freshness, and link quality. In case of degradation in the trust level, such that the established SSLA is not compliant, CASTOR can trigger path reconfiguration or notify the GEO service provider of reduced trust. Notification by the Trust Awareness API is sent to GEO if and only if the SSLA is not compliant. If a possible trust degradation occurs so that the SSLA is still compliant, then no notification

is produced. When trust levels remain above the threshold, vehicles receive trustworthy alerts via DENM messages such as “STOP: Collision risk involving VRU”. If trust levels drop below threshold, the GEO service provider notifies the vehicle through a DENM message: “Reduce speed: Possible VRU presence.”

User Story Workflow

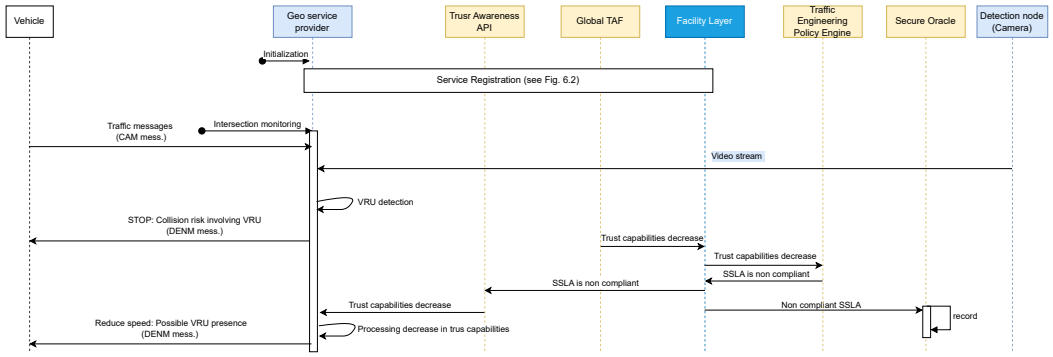

Figure 8.27: UC3.US3 Workflow

CASTOR KPIs

Table 8.20: Quantitative KPIs for user story UC3.US1 - Traffic Operator

KPI	Definition	Target Value
Latency	End-to-end latency for exchanging non-safety messages between the GEO Service Provider and the Traffic Operator, taking into consideration the CASTOR-related overhead, shall remain within the latency limit defined by 5GAA for the Real Time HD Map update service.	$10 - 200sec$ [9, pp. 33-34]
Reliability	Percentage of time the system is operational and capable of delivering the intended service shall fulfil service level reliability specified by 5GAA for the Real Time HD Map update service. This involves measuring operational time relative to downtime or service interruptions, and verifying the integrity of the traffic models at reception.	99% [9, pp. 33-34]

Table 8.21: Qualitative KPIs for user story UC3.US1 - Traffic Operator

KPI	Definition	Target Value
Service degradation	CASTOR framework shall provide alerts of SSLA violation to all involved parties (i.e., GSP and TO)	TRUE
Throughput	CASTOR framework shall accommodate traffic rates as specified in [10]	TRUE

Table 8.22: Quantitative KPIs for user story UC3.US2 - Emergency Operator

KPI	Definition	Target Value
Latency	End-to-end latency for establishing emergency calls between the GEO Service Provider and the Local Emergency Operator (LER), taking into consideration the CASTOR-related overhead, shall remain within the latency limit defined by 5GAA for the xCall service[2]	$\leq 1sec$ [12, pp. 16-18]

[2] 5GAA specifies that in case of an accident, a call with a local emergency responder must be initiated in 1 second[12, pp. 16-18]

KPI	Definition	Target Value
Latency	Local Emergency Operator decides and communicates the emergency to Central Emergency Operator	in less than 5 minutes (300 sec)
Reliability	Percentage of time the system is operational and capable of delivering the intended service shall fulfil service level reliability specified by 5GAA for the XCall service. This involves measuring operational time relative to downtime or service interruptions, and verifying the integrity of the alerts at reception.	$99.x\%$ [12, pp. 16-18]

Table 8.23: Qualitative KPIs for user story UC3.US2 - Emergency Operator

KPI	Definition	Target Value
Capabilities	CASTOR should provision cross-domain services satisfying SSLAs across different domains	TRUE

Table 8.24: Quantitative KPIs for user story UC3.US3 - VRU

KPI	Definition	Target Value
Latency	End-to-end latency for exchanging data between cameras and the GEO Service Provider, taking into consideration the CASTOR-related overhead, shall remain within the latency limit defined by 5GAA for Yielding Right-of-Way to VRU service[12, pp. 39-43]	$200\ msec$ [12, pp. 39-43]
Reliability	Percentage of time the system is operational and capable of delivering the intended service shall fulfil service level reliability specified by 5GAA for Yielding Right-of-Way to VRU service. This involves measuring operational time relative to downtime or service interruptions, and verifying the integrity of transmitted data at reception.	99.9% [12, pp. 39-43]

Table 8.25: Qualitative KPIs for user story UC3.US3 - VRU

KPI	Definition	Target Value
Throughput	CASTOR framework shall accommodate traffic rates according to monitored area and number of detection nodes	TRUE

8.5 Future-Proofing Next-Generation Unmanned Aerial Vehicles Communications towards Critical Infrastructure Sustainability

In the critical infrastructure inspection domain, fleets of Unmanned Aerial Vehicles (UAVs) rely on secure and efficient network communications to coordinate missions, relay telemetry data, and ensure operational continuity. Trust and reliability in network routing are crucial to maintaining real-time situational awareness, ensuring mission success, and protecting sensitive information from cyber threats on the infrastructure.

8.5.1 System Model

Figure 8.28 depicts an abstract view of the topology considered for drone inspection. The architecture is based on the “ETSI TS 123 501 - 5G; System architecture for the 5G System (5GS)” standard. The

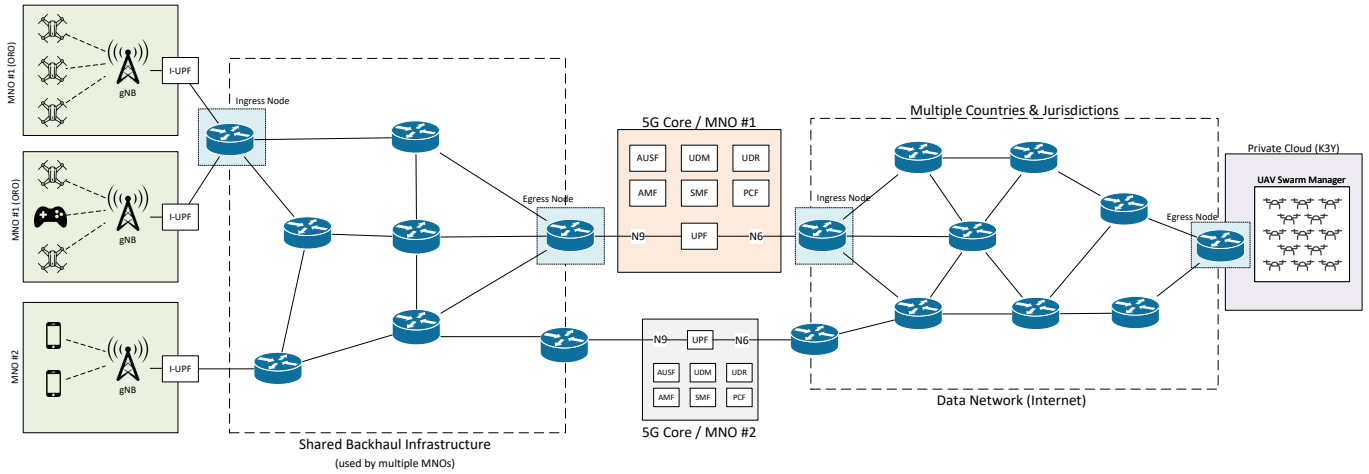

Figure 8.28: UAV-related System Model

topology depicts the network path followed by the 5G-enabled Unmanned Aerial Vehicles (UAVs) to access the UAV Swarm Manager, which coordinates the drone's operation and obtains real-time telemetry and data from the inspection process.

A shared backhaul network is accessed by multiple base stations (gNB), belonging to different Mobile Network Operators (MNOs). The UAVs are positioned to cover the large area that they must cover during the inspection process. Considering the high requirements in terms of latency and bandwidth, multiple intermediate User Plane Functions (I-UPFs) are placed close to the access network, serving one or multiple gNBs. The I-UPFs pass encapsulated traffic to the central UPF at the 5G core through the 5G backhaul, a high-speed network that consists of multiple routing nodes and redundant paths, which is shared by multiple MNOs. Finally, moving towards the data network, the 5G core gets access to the data network, ultimately reaching the UAV swarm manager.

8.5.1.1 System Components

Monitoring and inspection of the energy infrastructure is made possible by using a fleet of UAVs that are deployed in a 5G network. A UAV swarm manager is connected to the other edge of the network, providing commands, updating drone missions and locations, and receiving information collected by drones. The key components of the system are:

1. **Unmanned Aerial Vehicles (UAVs) fleet**: The UAVs are equipped with high-resolution cameras, LiDAR, or other specialized sensors to perform autonomous inspections of critical infrastructure, such as power lines, pipelines, or bridges. The UAVs receive waypoints from the GCS for flight coordination, and transmit live sensor data and telemetry (position, battery status, and system health) to the UAV swarm manager.

2. **Ground Control Station (GCS):** The Ground Control Station serves as the central hub for mission planning, real-time monitoring, and control of the UAV swarm during critical infrastructure inspections. The GCS will be used to define mission inspection parameters, receive waypoints from the Swarm Manager and upload them to the UAVs, and provide an interface for the mission that includes telemetry data, UAVs positions and their respected battery levels and their sensor feeds.

3. **UAV Swarm Manager**: A Swarm Manager, running within a private cloud domain, acts as the orchestrator for multi-UAV missions. The Swarm Manager acts as an abstraction layer capable of selecting and generating the path trajectory of the UAVs to be followed in a compliant mavlink format. Mission waypoints are computed by the Swarm Manager's and published to each UAV over

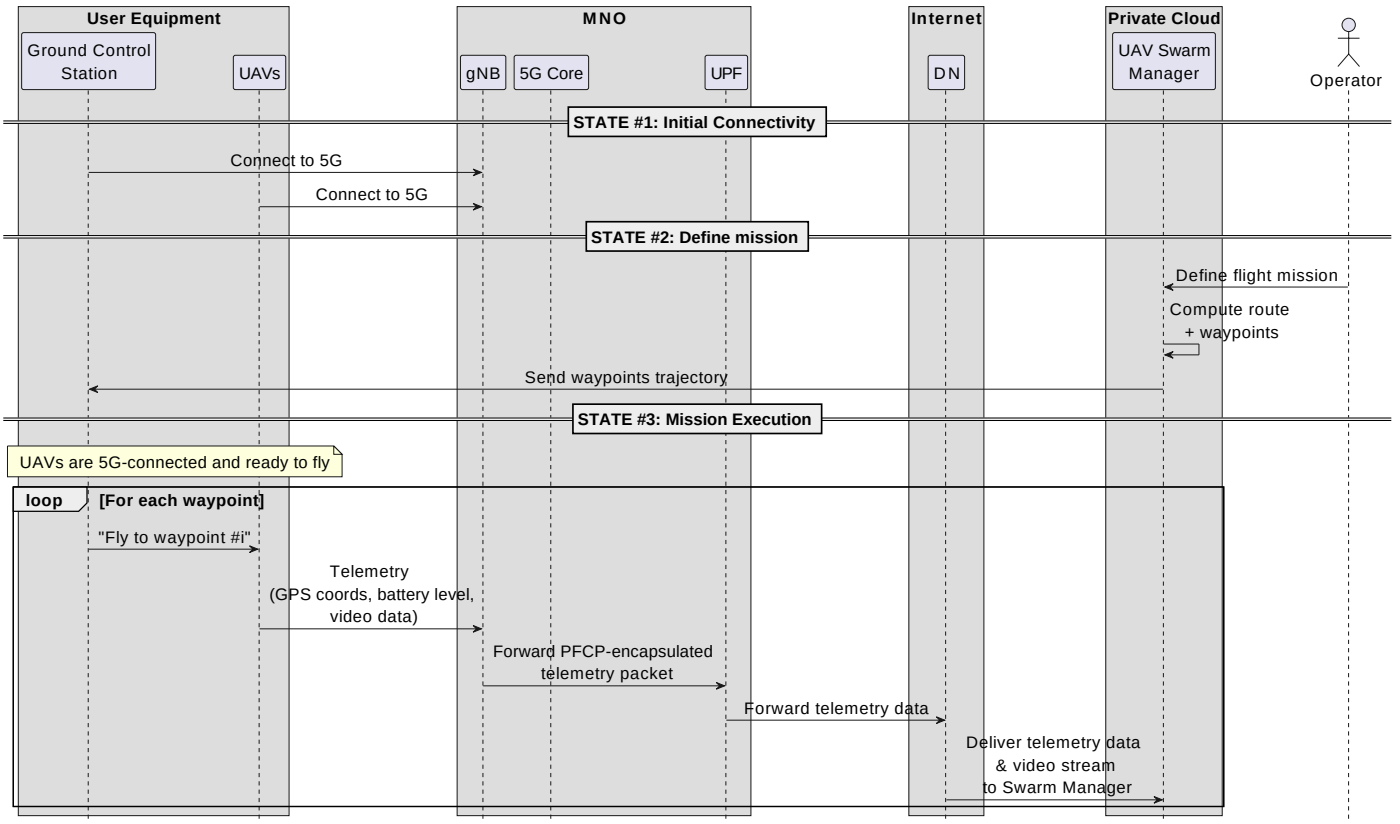

Figure 8.29: The communication flow in UC4

their dedicated 5G PDU sessions. Heartbeat messages and status updates flow continuously from the UAVs back to the Swarm Manager, enabling real-time status updates of each UAV.

4. **5G Core Network**: The 5G network core enables UAVs and the GCS to access the data network and the UAV swarm manager, as well as for the UAVs to reach the GCS. The UAVs are tethered to UAV GCSs through 5G links. UAVs receive flight plans before launch, which lay out a succession of waypoints that the UAV must adhere to when travelling from launch to destination sites.

5. **Routers**: These are intermediary devices that provide redundant paths to forward data between the access network, the 5G core and the private cloud hosting the UAV swarm manager. The routers carry various types of data, including UAV telemetry, sensor data, and control commands across multiple network segments. Routing protocols, such as OSPF and BGP, are used to establish routing paths between the network segments.

8.5.1.2 Communication Flow

Figure 8.29 depicts a high-level view of the communication flow between the components of a UAV operation. The overall communication process is separated into three states. First, the initial connectivity state begins with the UAVs and the GCS synchronize with an available gNB and then sending connection requests to the 5G core through the synchronized gNB. The connection requests are evaluated from the 5G core services, namely Access and Mobility Management Function (AMF) with the help of the Authentication Server Function (AUSF), the Session Management Function (SMF) setups the 5G sessions, and the 5G connectivity is established. The next state is the UAV mission definition, which involves message exchanges between the GCS and the UAV Swarm Manager. In particular, the UAV Mission Operator utilizes the UAV Swarm Manager software to define the mission, the routes and waypoints are calculated by the software, and these are sent to the GCS through the Internet. The User Plane Function (UPF) plays a critical role on transfering data packets (user plane traffic) in or out of the 5G core, essentially bridging

the 5G User Equipment (UEs) with the external networks, the Internet, and the UAV Swarm Manager. Finally, the third state is the mission execution, which can be described as a loop of a sequence of actions repeated while the UAV Swarm Manager is sending new waypoints. The loop starts with the GCS, which sends the waypoints to the respective UAVs (with the help of the 5G core). Then, the UAVs perform the action while continuously transmitting relevant live telemetry in terms of GPS coordinates, battery level, device status and high-quality video feed to the gNB. This user plane traffic reaches the 5G core and, through the UPF service, the user plane traffic goes out to the Internet. The telemetry & video feed data navigate through the Internet in order to reach the UAV Swarm Manager. This loop is active while the UAV operation is ongoing.

8.5.2 "As-is" Scenario

The 5G-enabled UAV swarms provide multiple benefits for the inspection of large-scale critical infrastructure (e.g., power lines) in terms of low-latency and high throughput connectivity. To achieve a large-scale inspection, mission-critical coordination data such as telemetry data, waypoint's locations and video feeds, images and thermal scans, have to be exchanged with a Swarm Manager, which may be hosted on a private cloud, often located even in another country. As such, it is challenging to ensure adequate quality of service, in terms of security and availability of network resources, as the required data transmission involves multiple network nodes with different properties, available resources, and jurisdictions.

In particular, starting from the network edge and the 5G RAN, the UAV data may need to pass through a shared backbone network infrastructure in order to reach the 5G core. Adopting the mobile edge computing model, which aims to enhance the performance of 5G networks by bringing computing power and data processing closer to the location where the data is generated/consumed or close to dedicated accelerators of compute-intensive tasks, the 5G operators may decouple the centralised 5G core system, by bringing some of its components closer to the edge [81]. In this context, multiple UPFs, called intermediate UPFs (I-UPFs), are deployed closer to the access network in order to reduce the computational workload of the central UPF, considering that UPFs comprise the most compute-intensive workload within the packet core [105]. This approach aims to reduce end-to-end latency and back-haul bandwidth consumption, thus preventing congestion in the core network [94]. Hence, the user plane traffic from the gNBs may reach the 5G core (and backwards) through the I-UPFs, however, the 5G core traffic (i.e., the traffic between the I-UPFs and the anchor UPF) uses a shared transport network that is operated by another entity, e.g., by a neutral host that provides a shared IP infrastructure.

Next, once the data exits the 5G Core and enters the Data Network, it must travel through a series of routers, belonging to different network operators, infrastructure providers and countries, until it reaches the Swarm Manager. The routing usually relies on pre-established paths using well-known algorithms, such as OSPF and BGP.

A common inspection mission starts by providing waypoints to the UAVs through the Swarm Manager, while simultaneously uploading mission inspection and telemetry data from each UAV. Assuming the originating point on the UAV swarm manager, and that the UAVs may be located even on a different country, the mission data will traverse through multiple routing nodes, involving multiple network operators and ISPs, possibly with different national jurisdictions. Hence, there are no guarantees that the intermediary nodes (i.e., routers) that forward the sensitive mission data, have their software and firmware integrity checked or that their runtime behaviour is continuously monitored or cryptographically attested, raising the possibility of silent degradation or malicious takeover. For example, if a router along the path to the UAV Swarm Manager and vice versa is operating on outdated firmware, has been misconfigured, or is backdoored via a supply chain attack, the UAVs' data may be intercepted, delayed, or silently altered, without any alarms being raised.

8.5.3 Use Case needs from CASTOR

Using UAVs for mission inspection of power infrastructure depends on uninterrupted telemetry, sensor payloads, and operator command loops traversing a complex end-to-end network, beginning in the 5G RAN to the I-UPF on the edge network, towards the 5G core, reaching the UAV operator's private cloud through the Internet. Today's transport fabric is optimized for bandwidth and latency guarantees but is agnostic to underlying security risks introduced by routing nodes or by traversing through untrusted networks.

It is noteworthy that a cryptographically attested carrier-grade router in the 5G transport core is treated no differently from a legacy, unverified router in a public backbone segment, and no trust metadata accompanies the inspection data as it travels upstream, so that the trustworthiness of the data streams can be assessed. As a result, the UAV operator has neither visibility nor guarantees into the trustworthiness of the networking path followed by each data stream reaching the cloud and backwards, leading the operator to assume all data streams are equally reliable.

To resolve this gap, it is required from CASTOR to deliver three foundational functions for trust-assured UAV inspection operations:

1. **Router Trust Attestation** CASTOR continuously evaluates the reliability and trustworthiness of each intermediary node in the UAV communication path, from the access network to the UAV cloud, assessing each node's firmware integrity and run-time state.
2. **Trust-Weighted Path Selection & Failover** CASTOR automatically inspects nearest alternate paths that comply with a Required Trust Level (RTL) when the trust score of a router falls below the RTL, balancing trustworthiness with latency and throughput.
3. **Trust-Aware Service-Level Security Agreements (SSLAs):** Data publishers create primary SSLAs that trust must comply to as well as fallback SSLAs. CASTOR notifies which tier is currently active, which allows to dynamically adjust the stream path while preserving data continuity.

UC4 Network/Trust Properties of Interest	
Name	**Description**
Integrity	Ensures data is not altered or tampered with during transmission. It is a security measure needed for all data streams, ensuring the authenticity of the data.
Latency	Time taken for packets to traverse the network. Every data stream is expected to have low latency to reach the destination as quickly as possible.
Bandwidth	Represent the networks capacity of carrying data. Adequate bandwidth is essential for transmitting smooth high resolution video feeds.
Reliability	Ensures complete delivery of data streams, needed mission data to traverse the network.
Availability	Guarantees consistent packet delivery of mission data, avoiding gaps in UAV telemetry and trajectory data, preventing unsafe UAV behaviour due to command loss.
Jitter	Measure variation in packet delivery times. Low jitter ensures consistent updates in telemetry data, and predictable time in data streams.
Auditability	Logs must be tamper-evident so post-mission auditors can prove telemetry wasn't altered
Confidentiality	Ensure data is accessible only to authorized entities, helping protect sensitive data for operational security

Table 8.26: Network and Trust properties as service-level objectives

8.5.4 ”To be” Reference Scenario 1: CASTOR in the Data Network

The objective of this use case scenario is to ensure that UAV mission data exiting the 5G core will reach the UAV Swarm Manager in the private cloud, by using paths with routers that satisfy certain SSLAs. CASTOR will continuously attest to the trust of every router in the Data Network, until reaching the private cloud, computing the trust of each path, and rerouting the traffic to a path with higher trust score when the trust score falls below the specified RTL.

This use case scenario requires low-latency and tamper-evident communications to maintain real-time situation awareness. While networks are usually optimised through QoS policies and network slicing, there are no guarantees that, once the traffic leaves the anchor UPF and enters the Data Network, will not pass through a compromised node that could tamper with or compromise the confidentiality of the data flows. This creates blind spots where efficient, yet compromised routers can handle sensitive flows, while it is commonly assumed that all Internet routes are equivalently trustworthy, rarely assigning path weight to the probability of tampering in the forwarding plane.

8.5.5 ”To be” Reference Scenario 2: External Risk Indices as input to CASTOR in Data Network

The objective of this use case scenario to ensure that a risk index provided by an MNO for the UEs and the overall 5G infrastructure, can affect the selection of the optimal path that CASTOR decides during the path assessment, thus minimizing the overall risk. Purpose of the risk index, obtained through internal risk assessment processes of the MNO that are agnostic to CASTOR, is to reflect the overall security risk characterizing the 5G data streams, aggregating multiple inputs such as the risk score of relevant vulnerabilities, the probability and impact of potential incidents, the criticality of the UAV mission, ongoing threat campaigns and other cyber threat intelligence. The risk index expresses how important it is to select secure & trustworthy routers during the data transmission in order to minimize the probability of a risk materialization. Given the above, this scenario will study how the path selection of CASTOR will be influenced by the risk index, balancing the satisfaction of the service quality with the probability of a potential risk materialization on the intermediary routing nodes.

8.5.6 Reference Scenario User Stories for Scenarios 1 and 2: CASTOR in the Data Network and Risk-aware Path Selection

UC4.US1a - Nominal operation

As a UAV mission operator, I need to have quality of service guarantees in the communication flows between the UAVs and the UAV Swarm Manager in terms of **{bandwidth, latency}**, considering the requirements of the diverse data streams, such as telemetry data and high-definition video streams.

User Story Confirmation

The UAV equipment, including both the UAVs and the GCS, establish a MAVLink2 session with the UAV Swarm Manager over the Internet, by initializing IDs and signal presence. After the session establishment, the UAVs continuously stream sensor data towards the UAV Swarm Manager, where the information is analysed and new waypoints for location traversal are calculated and sent back to the UAVs. The UAV Mission Operator should be able, through the UAV Swarm Manager, to see the UAVs online, get and visualize their telemetry as well as to send back trajectory data.

User Story Workflow

1. The UAVs initialize 5G connectivity.

2. The UAVs establish MAVLink2 session with the UAV Swarm Manager over the 5G infrastructure and the Internet, establishing IDs and signal presence.
3. UAVs publish telemetry and video feed data towards the UAV Swarm Manager.
4. A series of routers in Infrastructure Layer forward the user plane data to the UAV Swarm Manager.
5. The UAV Swarm Manager processes data and calculates new trajectory paths.
6. Swarm Manager publishes trajectory update data to the UAVs.
7. The trajectory data follow the path back to the UAVs using the routers in Infrastructure Layer.
8. UAVs consume the trajectory update and adjust their paths.
9. As there are no trust & security guarantees, the UAV mission data get compromised from an untrustworthy intermediate node, through tampering or passive reconnaissance.
10. The UAVs and the UAV Swarm Manager consume the mission data, however, the UAV operation is now compromised without the knowledge of the UAV mission operator.

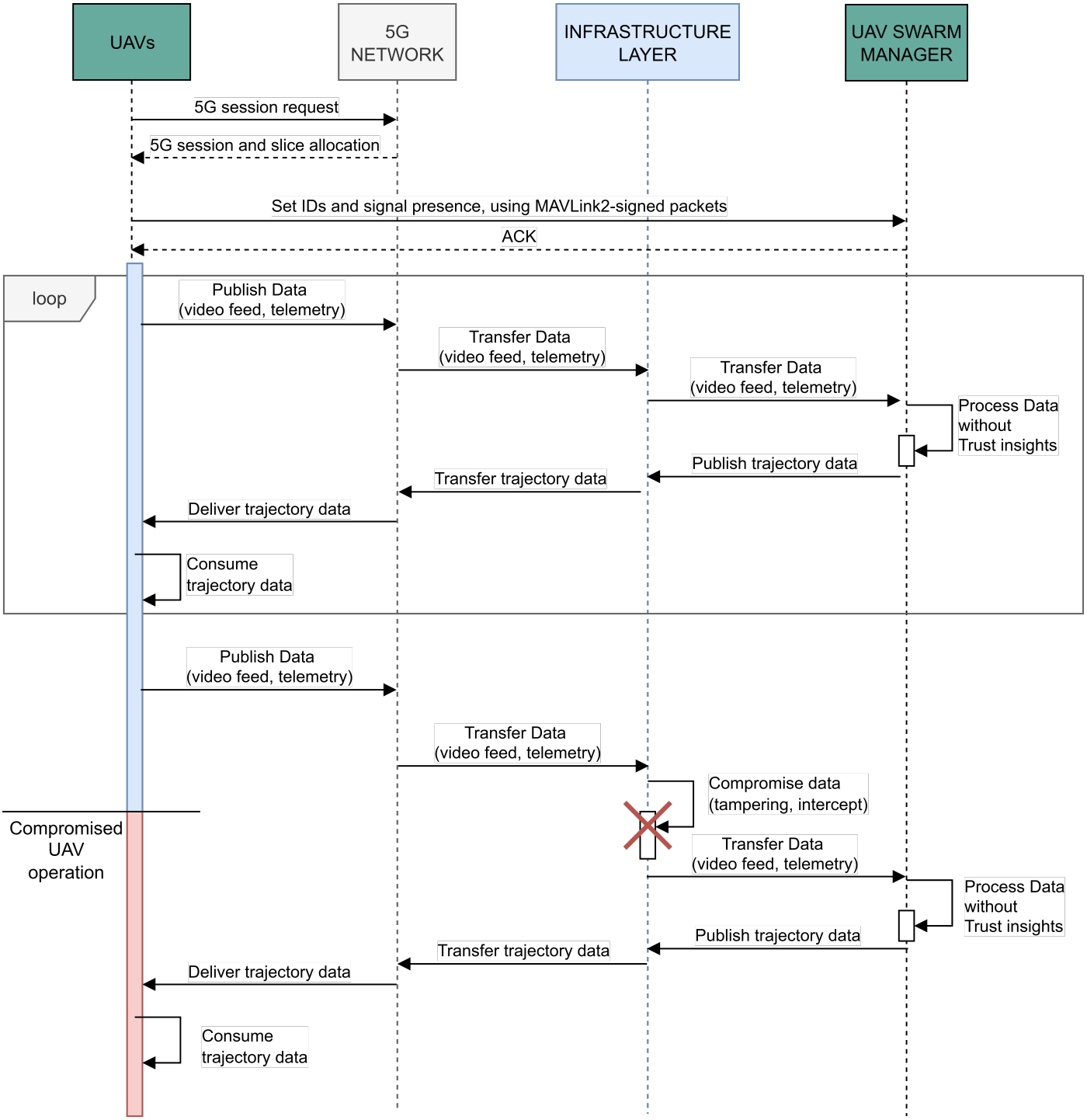

Figure 8.30: Sequence Diagram of UC4.US1a - Nominal operation

Reference Values The following metrics will be gathered and serve and baseline metrics, to be compared with the situation after CASTOR.

Table 8.27: Reference Values for user story UC4.US1a

Measurement	Description	Value / Scenario
Baseline Latency	One-way delay between the UAV and the Swarm Manager.	Obtained in the nominal operation, without CASTOR.
Command Loop RTT	GCS command to UAV acknowledgment round-trip time.	Obtained in the nominal operation, without CASTOR.

UC4.US1b - Fallback SSLA

As the UAV mission operator, if my primary SSLA with **{bandwidth, latency, integrity, confidentiality}** requirements cannot be met, I would like to be notified about it, and then a fallback SSLA with the minimum-required trust should be activated and enforced. If the requirements of the fallback SSLA cannot be guaranteed too, I want to be notified about it, so that I can decide whether to terminate the UAV mission.

User Story Confirmation

A fallback SSLA is defined, along the primary SSLA, to be used when the requirements of the primary SSLA cannot be satisfied. During the trust assessment of the paths, CASTOR verifies whether the primary SSLA is violated or not. Initially, the primary SSLA is violated, triggering CASTOR to enforce the fallback SSLA, by adjusting the infrastructure layer accordingly, and to notify the UAV mission operator about the SSLA degradation. Then, the fallback SSLA is also violated, and as a result, CASTOR notifies the mission operator to take the appropriate action whether to cancel the UAV mission.

User Story Workflow

1. The initial SSLA along with a fallback SSLA are established by CASTOR.
2. Both SSLAs are recorded in CASTOR DLT using the Secure Oracle.
3. The UAV mission unfolds, and UAVs begin to publish data.
4. The Global TAF will update the topology diagram, and it will be shared with the Traffic Engineering Policy Engine for the trust assessment.
5. During the trust assessment, the Traffic Engineering Police Engine detects that the initial SSLA requirements cannot longer be guaranteed.
6. The change in the SSLA is shared and recorded on the Secure Oracle, and the fallback SSLA is now used during the trust assessment.
7. The Traffic Engineering Policy Engine will now test against the fallback SSLA.
8. The fallback SSLA is violated too, and is recorded in the Secure Oracle.
9. CASTOR informs the operator through the Trust Awareness API that no SSLAs can be guaranteed.

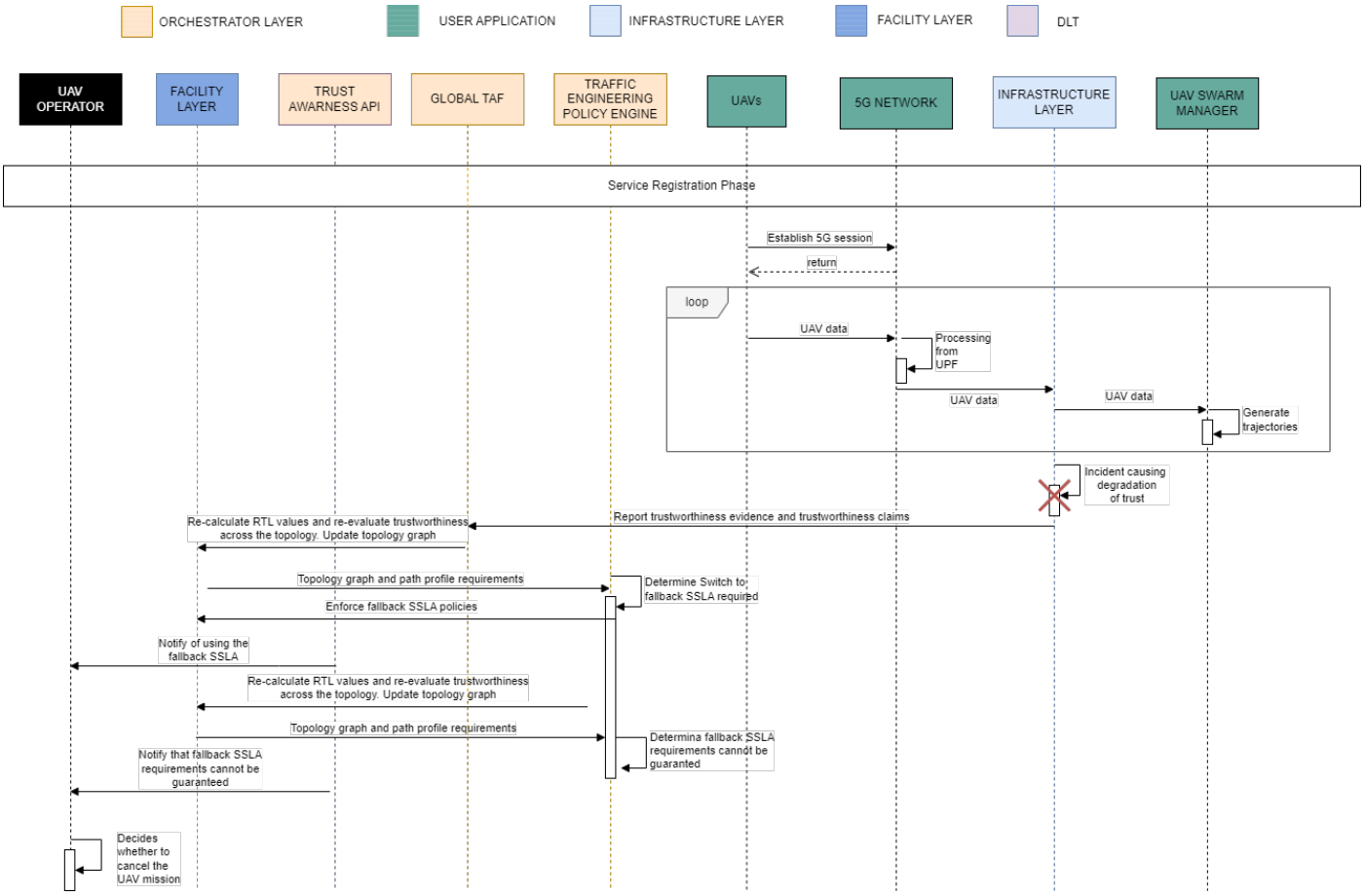

Figure 8.31: US4.US1b: Fallback SSLA

CASTOR KPIs

Table 8.28: CASTOR KPIs for user story UC4.US1b

KPI	Definition	Target Value
Latency within the Data Network	One-way delay between the UAV and the Swarm Manager, considering the overhead introduced by CASTOR.	$\leq 5\%$ compared to the nominal operation.
SSLA violation accuracy	This KPI measures how accurately CASTOR updates the ATL in response to observed misbehaviours (against to what will have been defined as ground truth considering the SW-in-the-loop nature of this setup allowing the truthful characterization of the entire scene). It is defined as the covariance matrix of the error vector on object (misbehaviour) detection ration against the ground truth.	$\geq 95\%$
Routing path alternative establishment	Time needed to converge and determine the paths.	Similar Convergence time to existing rerouting capabilities when no-recalculation takes place (sub-50ms).

KPI	Definition	Target Value
RTL behaviour correctness	RA yielding RTL based on which the Trust Level Estimation will take place.	Trust Decision correctly captures the behaviour pattern (trust increase/decrease) depending on injected misbehaviours.

UC4.US1c - SSLA compliance audit

As the UAV mission operator, I want to periodically review the SSLA compliance {**Auditability**} so that I can verify the integrity and compliance of my UAV operations.

User Story Confirmation

The UAV operator acts as an auditor and wants to perform an SSLA compliance review. CASTOR has recorded every change in the SSLAs during the trust assessment and when the UAV auditor requests an SSLA compliance audit, CASTOR provides the history of the SSLA changes. The auditor then reviews this report, and verifies that all sensitive information travelled only over compliant paths and that every SSLA value change was properly documented.

User Story Workflow

1. A third-party auditor is contracted to perform an SSLA compliance review.
2. The initial SSLA is recorded on the CASTOR DLT using the Secure Oracle.
3. During the execution of the mission, trustworthiness claims will be provided to Global TAF which updates the topology graph.
4. The new proposed paths are shared to the Traffic Execution Policy Engine.
5. The Traffic Engineering policy engine determines if the new paths violate the SSLA or not.
6. Any change in the SSLA decided by the Traffic Execution Policy Engine is recorded on the Secure Oracle.
7. The auditor requests an SSLA compliance report from CASTOR using the Trust Exposure Layer.
8. CASTOR shares with the auditor a record of the SSLA changes that are stored in the Secure Oracle.

User Story Workflow

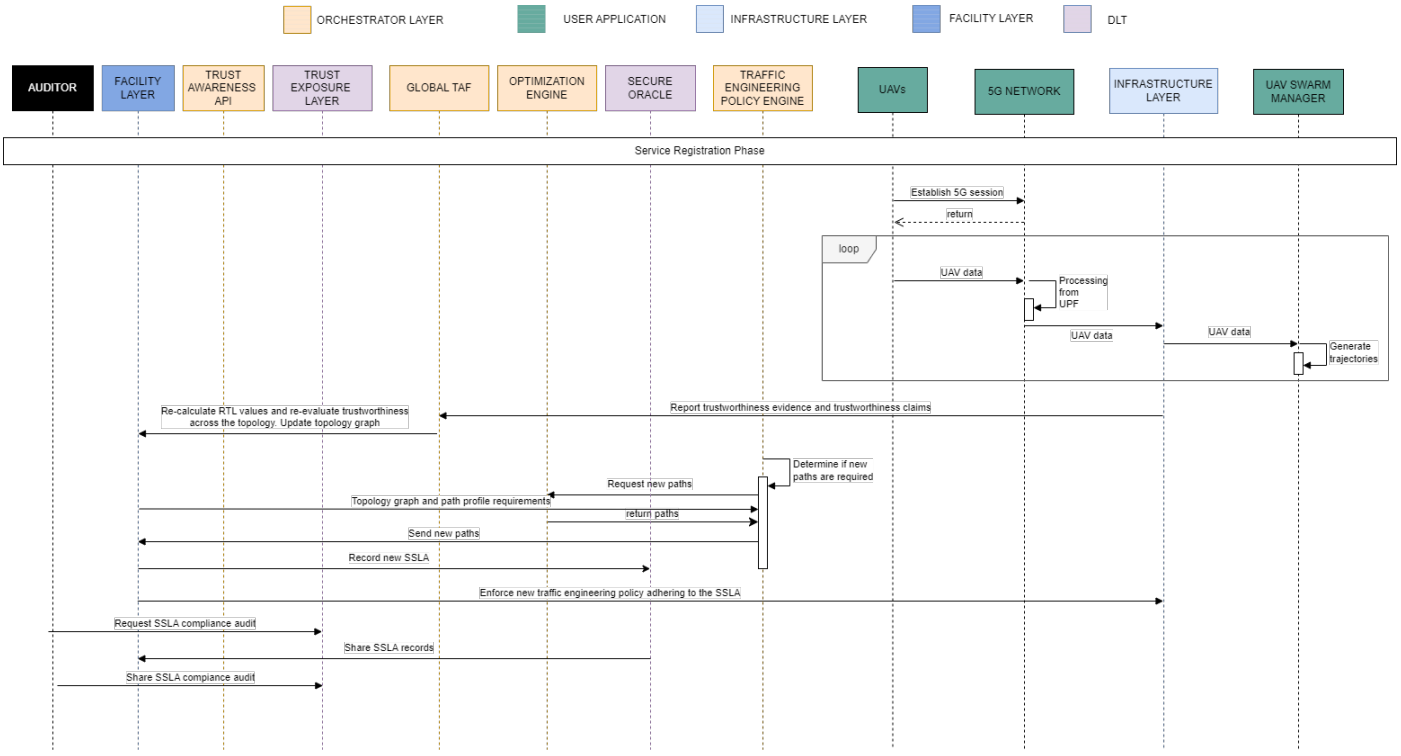

Figure 8.32: UC4.US1c: SSLA compliance report

CASTOR KPIs

Table 8.29: CASTOR KPIs for user story UC4.US1c

KPI	Definition	Target Value
Routing path alternative establishment	Time it takes for the Optimization Engine to recommend alternative paths (multi-path control feature of CASTOR). This might be based on pre-calculated path segments (if no nw service intents have arrived that may results in the overall update of the set of enforced routing policies) and/or freshly calculated paths. In the former case, the goal is to identify CASTOR's sensitivity in topology and temporal changes whereas the latter focuses on the performance bechmarking of the optimization process.	TRUE, this includes the time to enforce the new TE policy to the topology.
SSLA compliance information availability from internal auditor	It measures the latency of accessing the SSLA compliance information, given that the SSLA auditor is part of the Service Provider and therefore uses the Trust Awareness API.	$< 1sec$
SSLA compliance information availability for external auditor	It measures the latency of accessing the SSLA compliance information, given that the SSLA auditor is external entity and therefore uses the DLT-based Trust Exposure Layer.	$< 5sec$

UC4.US2a - Risk-aware path selection

As a UAV mission operator, I want CASTOR to consider the risk index of the 5G network in the selection of network paths that satisfy SSLAs with {**high confidentiality, high integrity, high availability**}.

User Story Confirmation

During the preparedness phase, a risk index is shared by the MNO to CASTOR, which represents the overall risk status of the 5G infrastructure and UEs. Utilizing the risk index, CASTOR performs more accurate risk assessment, which considers the potential vulnerabilities of the 5G infrastructure (e.g., vulnerabilities from the UAVs), guiding CASTOR to enhance the creation of RTL, thus selecting better paths with stricter security guarantees.

User Story Workflow

1. Using the Trust Awareness API, the MNO informs CASTOR about the Risk Index representing the overall risk status of the 5G infrastructure and the UEs.
2. The Risk Index is used by CASTOR for the calculation of the RTL for the trust assessment.
3. The Risk index is persisted on-chain to the Secure Oracle.
4. During the trust assessment, the Risk Assessment Engine retrieves the Risk Index and uses it as input for the calculation of the RTL.
5. Calculate ATL and detect that ATL does not satisfy RTL values.
6. The Global TAF evaluates the current path and updates the topology graph.
7. Traffic Engineering Policy Engine will determine if the path is compliant to the SSLA or not.
8. If the path is non-compliant, another alternative path will be selected by the Optimization Engine.

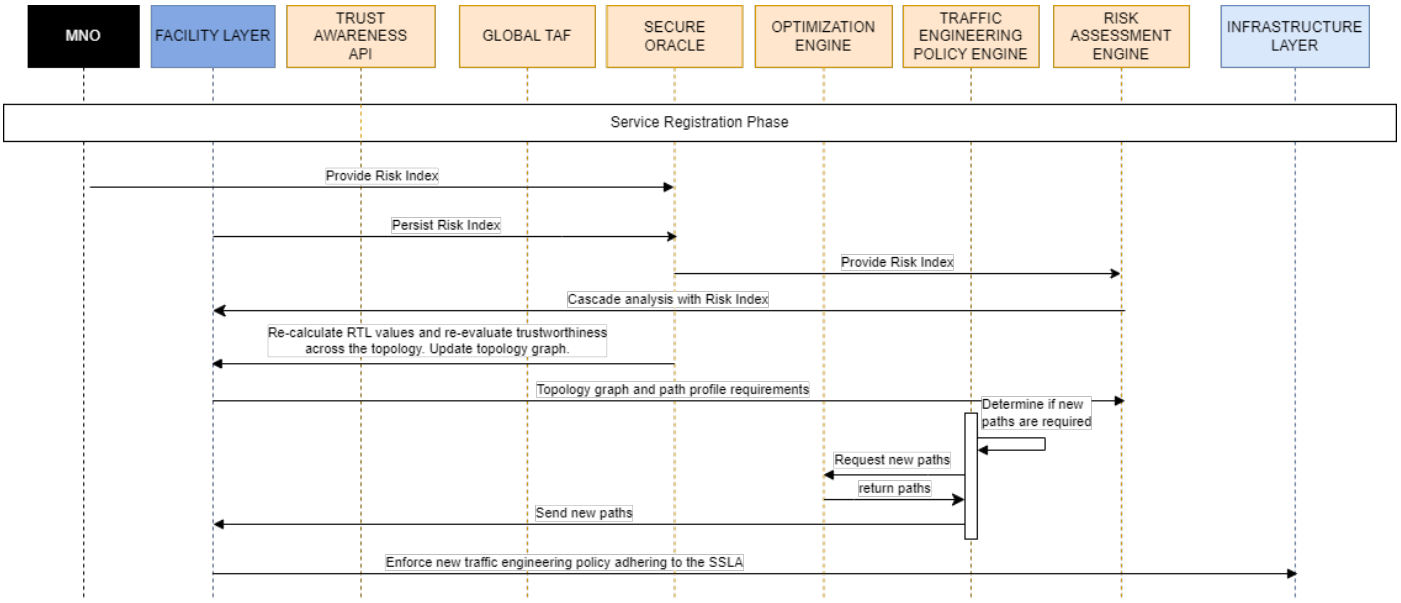

Figure 8.33: UC4.US2a: Risk Index

CASTOR KPIs

Table 8.30: CASTOR KPIs for user story UC4.US2a

KPI	Definition	Target Value
Risk assessment accuracy	Measures the accuracy (and impact therefore) of the risk assessment process on the RTL calculation when accounting for additional demands that may stem from a vulnerable profile of the source AS - expressed through abstract attack likelihood values.	Improved by 20%, compared to the baseline risk assessment accuracy.

8.5.7 ”Nice-to-have” Scenario User Stories: CASTOR in the Shared Back-haul Infrastructure

As an exploration case, this use case scenario delves into a distributed deployment of a 5G core setup, and aims to optimize the performance of the backhaul infrastructure, by choosing the rights paths that will ensure that the 5G core operates at maximum quality in terms of integrity, availability and overall performance.

Low latency is expected to meet end-to-end reliability and integrity across deployments in 5G. Even though QoS policies and network slicing can be applied over UPF to UPF communication (i.e, N9 interface), and security measures are taken to protect the N9 interface, such as VPN and new tunnelling techniques [141], there is no guarantee that the communication over the N9 interface will remain uncompromised from unknown vulnerabilities and misconfigured VPN protocols as well as it will remain reliable and unaffected from outages or overloaded/underperforming network equipment. Given the above, CASTOR brings trust assessment on router level ensuring performance and trustworthiness of UPF-to-UPF communication. CASTOR eliminates this blind spot by dynamically evaluating router trust, ensuring that inter-UPF data paths are both performant and trustworthy.

It is worth noting that this scenario is agnostic on the user plane data, as the user plane data (i.e., the UAV mission data in this use case) is encapsulated during their transmission inside the distributed 5G core. Given that, the use case scenario explores the optimization of the 5G core itself without assumptions on visibility over the plain UAV data.

UC4.US3a - Nominal Operation

As an MNO, I want my data streams between the I-UPFs and the achor UPF to satisfy my **{latency, bandwidth}** requirements, to ensure that the operation of the 5G core adequately satisfies the communication requirements of my clients, including the UAV operator.

User Story Confirmation

The UAV operator starts the UAV mission, while the MNO does not have an SSLA in place. Traffic from UAVs is received from the I-UPFs , and the MNO confirms that the traffic is successfully received to the A-UPF.

User Story Workflow

1. UAV establish 5G session by interacting with the respective 5G services of the 5G core.
2. The UAV traffic goes through the local I-UPF.
3. The I-UPF traffic is encapsulated into GTP-U packets.
4. The GTP-U packets are routed through the backhaul network.

5. The GTP-U packets arrive at the anchor UPF and are decapsulated successfully.
6. At some point, a network failure in the backhaul network reduces the network reliability.
7. Due to the reduction of the network reliability, the GTP-U packets cannot longer be delivered.

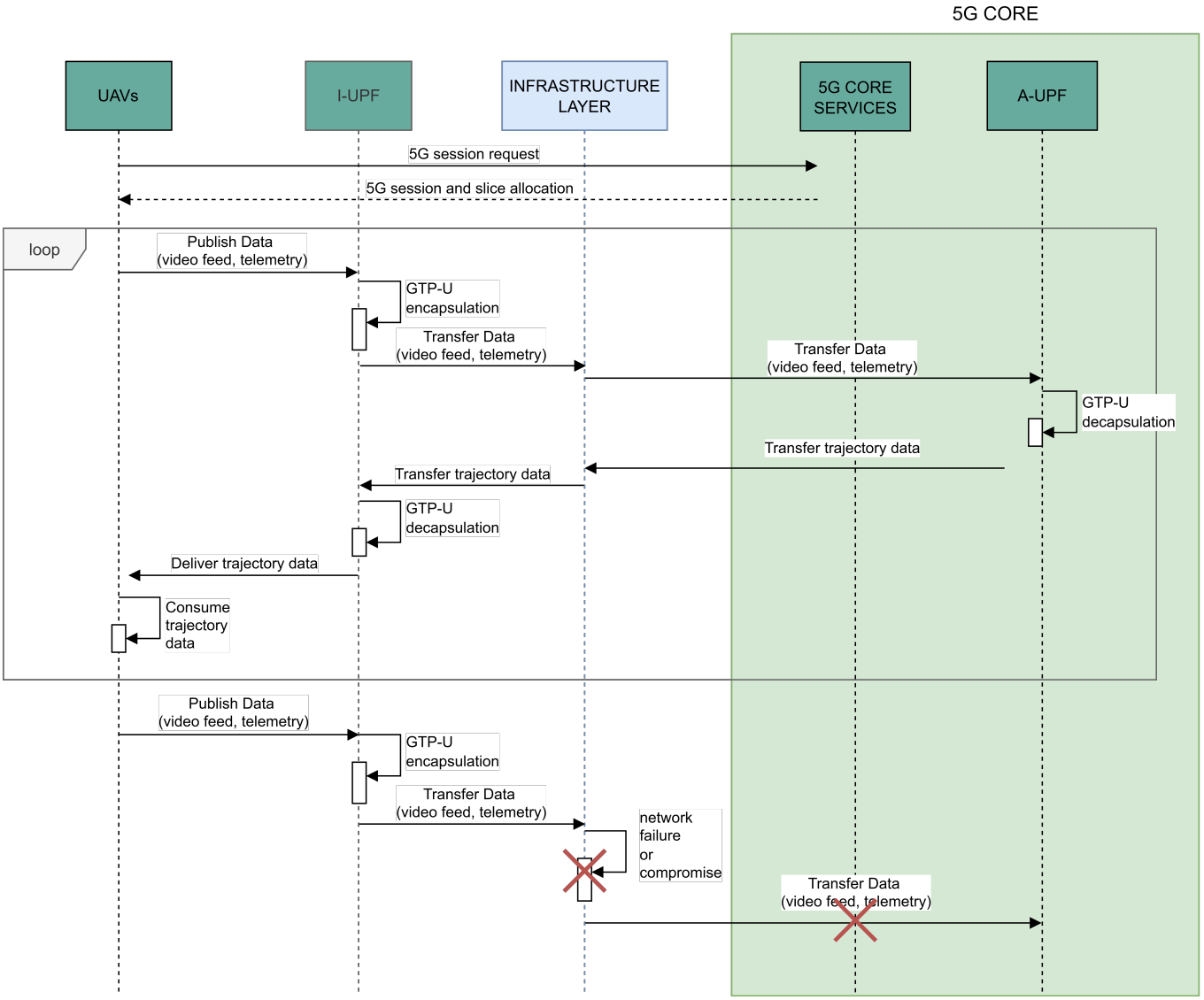

Figure 8.34: UC4.US3a: Nominal Operation

Reference Values

Table 8.31: Reference Values for user story UC4.US3a

Measurement	Description	Value / Scenario
Base Latency	One-way delay between the UAV and the Swarm Manager.	Obtained in the nominal operation, without CASTOR.
Baseline scalability & reliability	% of packet loss considering a failure on the infrastructure layer	Obtained in the nominal operation, without CASTOR.

UC4.US3b - Optimal path selection

As a neutral host operator, I want CASTOR to calculate and establish optimal paths between an I-UPF and the anchor UPF, in order to satisfy the agreed SSLA requirements in terms of {**high availability, high reliability, medium bandwidth, path integrity**}.

User Story Confirmation

The MNO provides the requirements to establish an SSLA in the backhaul network. While a UAV mission operator is in progress, the MNO confirms with the UAV operator that the mission is executed smoothly as a result of the optimal paths established by CASTOR.

User Story Workflow

1. The MNO provisions an SSLA with requirements in terms of high availability, high reliability, medium bandwidth and path integrity.
2. The infrastructure layer reports trustworthiness evidence and trustworthiness claims to the Global TAF.
3. The Global TAF updates the topology graph to the facility layer.
4. The facility layer provides the topology graph and path profile requirements to the traffic engineering policy engine.
5. The traffic engineering policy engine requests new paths from the optimization engine, and sends the new paths to the facility layer.
6. Finally, the facility layer enforces the new traffic engineering policy to the infrastructure layer, in order to adhere to the SSLA.

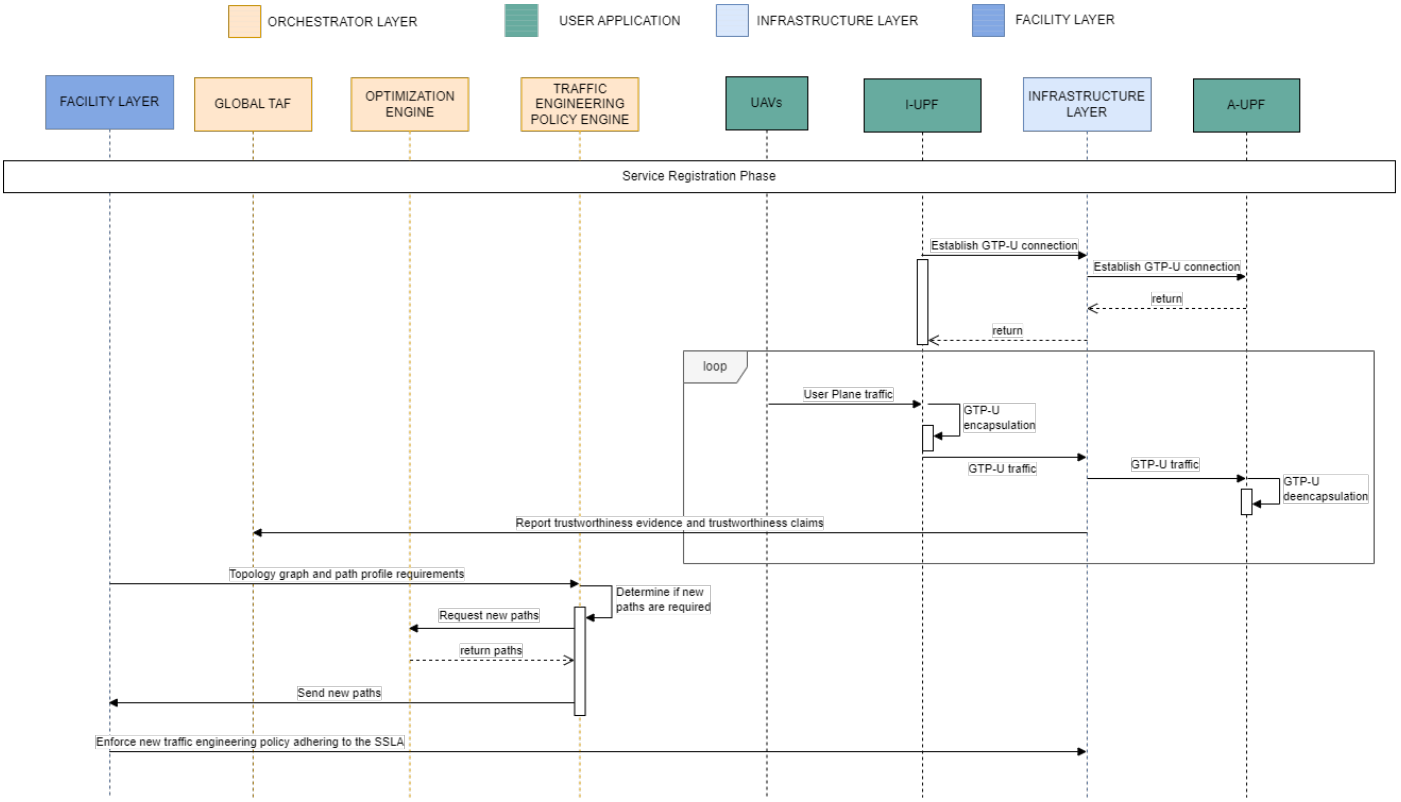

Figure 8.35: Workflow of UC4.UC3b

Table 8.32: CASTOR KPIs for user story UC4.US3b

KPI	Definition	Target Value
Latency within the Data Network	One-way delay between the UAV and the Swarm Manager, considering the overhead introduced by CASTOR.	$\leq 5\%$ compared to the nominal operation.
Reliability	% of packet loss considering both network and integrity compromises on the infrastructure layer, with CASTOR converging on a new path.	+ 20% improvement from the baseline value, with investigation of possible dependencies between CASTOR TE decisions in inter-connected domains.

8.6 Trust-Aware UAV Data Delivery Across Mobile Edge Attachments

Although the existing CASTOR use cases explore a wide range of trust-aware routing scenarios, what is still missing is an integrated demonstration of how CASTOR's orchestration layer interacts with application-level services and far-edge compute nodes, closing the loop between trust assessment, policy enforcement, and application behaviour. To this end, this Proof of Concept (PoC) scenario addresses this gap by placing orchestration, and especially the interaction between CASTOR's orchestrator, application orchestrators, and far-edge domains, at the centre of the scenario.

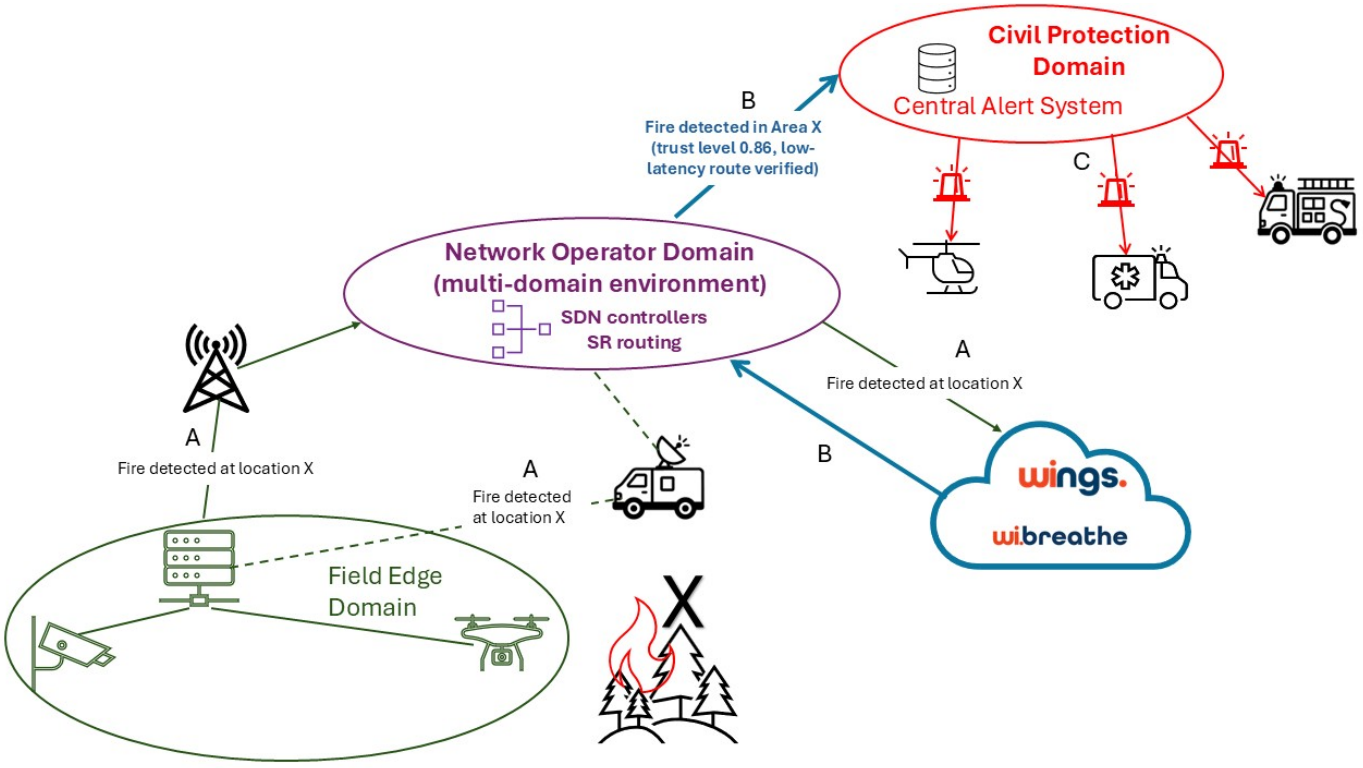

Figure 8.36: PoC Scenario

In Figure 8.36, the main components of the scenario are illustrated. The use case specifically focuses on trusted fire detection and dissemination of alerts. Its novelty lies in the fact that trust-related assessments can also be investigated at the far edge, including entities (e.g., UAVs) themselves acting as both UEs

and compute nodes, where integrity, telemetry provenance, and device posture contribute directly to end-to-end trust decisions. The key system components are the following:

- **Field Edge Domain:** This domain contains all sensing and edge-processing components that generate the fire-alert events:
  - **Edge Sensors & IoT Gateways:** Devices (e.g., thermal cameras, environmental sensors) detect fire indicators and produce alert messages. The gateway aggregates readings, performs local filtering, and prepares the alert for uplink transmission through the network.
  - **UAVs / Edge Compute Nodes:** UAVs can augment sensing or relay information. Importantly, this use case allows exploring trust evaluation at the far edge itself (e.g., UAV firmware state, gateway integrity), something not addressed in prior UCs. This extends CASTOR's trust assessment beyond routers, toward UE/edge-side posture.
- **Network Operator Domain (Unified Domain: Access + Core + Transport):** All connectivity infrastructure — radio access, core network functions, and IP/SR transport — is considered one programmable trust-aware domain: Provides the full forwarding path from the field edge toward Civil Protection systems. Routers, switches, and VNFs supply telemetry, configuration proofs, performance indicators, and trust evidence to CASTOR. SDN and Segment Routing (SR-TE) controllers implement the forwarding policies dictated by CASTOR. CASTOR computes the trust level for each network segment.
- **CASTOR Orchestrator Layer:** CASTOR performs continuous, network-wide trust assessment and trust-aware routing: Aggregates telemetry and trust metadata from routers, SDN controllers, and optionally edge nodes. Evaluates the trust posture of available SR paths and selects the one that satisfies policy constraints. Exposes only the active SSLA tier (not raw trust values) to the application orchestrator (WSMO). Triggers rerouting when trust degrades, preserving service continuity for time-critical fire alerts.
- **WINGS Orchestration Layer (WSMO) and wi.BREATHE application:** This layer is the entry point for fire detection services. It receives field-edge alerts and manages the life cycle of fire-detection workflows. Interacts with CASTOR to get the active trust tier, adapting application behaviour as needed. Orchestrates the application workloads (offloads between edge and cloud domains, etc.)
- **Civil Protection Domain (Central Alert System):** The final destination of validated alerts, which consumes the events exposed by the WSMO to receive fire-alert notifications delivered also through a CASTOR-verified path. Triggers operational response (dispatch vehicles, helicopters, emergency units).

In the figure, the results of three phases are depicted. In Phase A, field-edge alerts are sent to the WSMO/wi.BREATHE application; in Phase B, the fire-detection service notifies the CAS about the incident (fire); and in Phase C, the CAS informs the respective response units.

If the Trust Exposure Layer of CASTOR framework in the Network Operator Domain indicates that no SLA- or SSLA-compliant paths are available, the WSMO interprets this as an insufficient trust budget and orchestrates the UAV/Edge – based data analysis. The resulting fire-detection output is conveyed to the CAS within the Civil Protection Domain. So, in this case, the UAV/Edge must evaluate the trustworthiness of the fire detection itself. Otherwise, the proper indicators, after data pre-processing, are sent to cloud based wi.BREATHE instance, in order to execute the fire detection service. At service/application level, trust is linked to the provenance of the workflow execution, service-plane correctness (if the inputs are partially degraded, e.g. two sensors are offline, the service should not construct artificial trust, but instead explicitly express uncertainty) and adherence to SLO targets relevant to fire detection. Particularly, the provenance of the workflow execution, ensuring that every fire detection decision can be traced back

to the specific model version, configuration parameters, execution domain (field edge domain or WINGS domain, in this case), and the identity that triggered the process. This guarantees verifiable accountability, reproducibility of results, and transparent evidence for the Civil Protection authorities. The fire detection results are sent to the CAS together with a trust estimation and an indication that the information transfer was carried out through a path that meets the SLA requirements, thereby reducing service response time and ensuring that the ongoing situation cannot exhibit significant deviation from the one indicated in the alert signal. This is further supported by the fact that the life cycle of the fire-detection process is on the order of seconds. A high trust level prompts the CAS to immediately propagate the alert messages to the relevant consumers.

Chapter 9

CASTOR Framework Requirements

This chapter, as its name suggests, thoroughly describes the **key requirements** that characterize CASTOR throughout its framework. These requirements encompass security, functional, and non-functional considerations that are essential for the successful operation of the overarching CASTOR Trust Assessment functionalities. Following a **top-down approach** for identifying and delineating the requirements, the chapter initially presents the **overarching security requirements** and needs of the CASTOR framework as a whole (top), and then proceeds to the **functional and non-functional requirements associated with each operation and technology** within the CASTOR traffic engineering process and pipeline (bottom).

As aforementioned, CASTOR architecture coalesces between different technologies stemming from trusted and confidential computing to risk assessment and trust assessment. For each of these technologies or functionalities, we have not only captured the requirements that each technology/functionality must achieve and exhibit whey operated in a stand-alone manner, but also those properties that they need to be able to expose as part of an overall trust execution architecture (e.g., trusted path routing ecosystem). In other words, the main goal is to capture all these dependencies that can allow the overall vision of CASTOR.

Recall that CASTOR evasions to engrain trust as part of traffic engineering process considering adaptive to changes trust mechanisms for capturing the device state (e.g., trustworthiness state) of each element in the path. Thus, all functional requirements should analysed from a functional perspective such as the necessary dependencies to properly operate and for performance stand point as close to real time as possible to be seamless on the service provider's users. Thus, the present chapter details the critical requirements that will form the basis of the core technical requirements and set the scene for the preliminary designs, implementations and evaluations within CASTOR.

Towards this direction, the chapter begins by **establishing the overarching Security Requirements** (see section 9.1) of the framework. These requirements do not target specific artifacts within the system. Instead, this whole set of security requirements capture all the necessary security and trust requirements in the process of the overall CASTOR framework, from the far edge all the way to the orchestration layer. Recall that in order for this goal to be achieved, CASTOR is based on adaptive to changes mechanisms capturing the trust state of the underline devices. Thus, these security requirements include the properties that the far edge and the routing plane should be equipped to provide verifiable evidence for confidentiality, privacy, etc. requirements that drive the abstraction of trust evidence.

Table 9.1: Functional & Non-functional Requirement Categories

Functional & Non-functional Requirement Categories	Description
Trust Assessment Requirements (see sub-section 9.2.1)	These requirements involve all the functional and non-functional requirements to unlock the evidence- based theory of trust assessment in order to establish and maintain the trust level of the router and the path.
Router Operational Assurance (see sub-section 9.2.2)	These requirements involve all the functional and non-functional requirements of router's trust extensions in order to provide/share their trust evidence in a secure and verifiable manner.
Trust-aware Service Assurance (see sub-section 9.2.3)	These requirements involve all the functional and non-functional requirements of the trust aware management of the deployed services, including both (a) how to establish and enforce the trust policies and (b) how to monitor that no violation have been occurred.
Traffic Engineering Requirements (see sub-section 9.2.4)	These requirements involve all the functional and non-functional requirements and are explicitly focused on traffic engineering, capturing both the intra- and inter- domain aspects (cross-domain).

Having set the scene with these requirements, then we proceed with the **specific set of functional and non-functional requirements for each one of the core technologies** (detailed in Chapter 6) we consider in CASTOR (see section 9.2). Within the CASTOR architecture and its key artifacts, there are four crucial subcategories that are investigated: i) the **Trust Assessment Framework**, which defines the mechanisms for assessing and establishing trustworthiness across the routing plane, elevating trust from node-centric to path-centric trust characterizations; ii) the **router operational assurances**, which comprise the foundational hardware and software elements within a network element and are responsible for supporting the execution of trust-related operations through the provision (in a verifiable manner) of the necessary trustworthiness evidence; iii) the **trust-aware service assurance capabilities** that focus on the overall requirements that CASTOR needs to accommodate at the Orchestration Layer and above; and iv) the **traffic engineering requirements** which establish the characteristics in the forwarding plane so as to allow the incorporation of trust characteristics in the routing path provisioning considering both intra- and inter-domain scenarios. All these requirements are also crucial for the long-term success and sustainability of incorporation of robust and flexible trusted path routing considerations supported by the CASTOR framework.

It also has to be noted that most of the requirements discussed above pertain to the functional capabilities of the overall CASTOR framework. However, in certain cases, this chapter also introduces non-functional requirements to emphasize key aspects of the end-to-end CASTOR framework that constitute the driving factors that affect the use case KPIs as defined in Chapter 8. This ensures that no unnecessary overhead is imposed and, consequently, that the operational behaviour of the application workload traversing the provisioned routing policies remains unaffected. Table 9.1 below summarises these four identified requirement categories along with a short description to set the scene for the subsequent sections (see section 9.2).

9.1 Overarching Security Requirements

Table 9.2: SR.1 Device-support for Hardware-based Root of Trust

SR.1	
Title	**Device-support for Hardware-based Root of Trust**
Actors Involved	TNDE
Type	Security Requirement
Description	**Background**: CASTOR's overarching goal is to enable trusted path routing for networks, i.e., a trust-aware path provisioning that takes the security of the network nodes into account. To achieve this, the CASTOR orchestrator must continuously assess the trust level of each network node based on collected security claims. However, this requires the CASTOR orchestrator to be able to establish trust in its device-side components (especially the TNDE) on each network node as they form the device-side trusted computing base (TCB) for the evidence collection and trust assessment of the router nodes. Therefore, network nodes need to be equipped with a root of trust (RoT) that provides the necessary security capabilities for CASTOR's device-side components to report on their integrity and securely generate and manage security claims over the trustworthiness of the device's TNDIs. Otherwise, the CASTOR orchestrator will not be able to establish trust in the device nodes and securely verify the trustworthiness claims of the network topology. **Description**: The network devices must provide a hardware-based root of trust (HW RoT) forming a trust anchor on top of which CASTOR can establish trust in its device-side components (especially the TNDE). The HW RoT needs to provide capabilities for secure measurement, storage, and reporting / attestation. That is, the HW RoT must support the secure measurement of CASTOR's device-side TCB components in a verifiable way, allowing the CASTOR orchestrator to perform a secure remote attestation of the device TCB. Furthermore, the HW RoT needs to support the secure and flexible storage and management of code, data, and cryptographic keys forming the foundation of CASTOR's secure key management (outlined in SR.3), as required for the verifiable generation and sharing (SR.9) of trustworthiness evidence. Finally, CASTOR requires the HW RoT to provide anti-rollback protection capabilities for stored data and deployed software. This is necessary to enable CASTOR to enforce or verify that the intended security policies (SR.4) are enforced by the expected / up-to-date on-device software components (e.g., TNDE), preventing rollback attacks resulting in outdated policy or software versions (SR.5). **Remarks**: CASTOR's device-side components shall be conceptually agnostic to the underline specific HW RoT implementation to support a variety of devices from different vendors. Therefore, CASTOR plans to explore different HW RoTs in the context of virtual routers (vRouters) based on, e.g., secure enclaves (Intel SGX), virtualization, or TPMs, as we will further described in deliverable D3.1.
Connected to other requirements	SR.2, SR.3, SR.4, SR.5, SR.9
KPIs	**Description**: Number of operations supported by the underlying RoT — **Value**: ≥ 3 **operations**

Table 9.3: SR.2 HW-based Isolation of CASTOR's device-side TCB Components

SR.2	
Title	**HW-based Isolation of CASTOR's device-side TCB Components**
Actors Involved	TNDE
Type	Security Requirement

Description	**Background**: Each network node willing to participate in CASTOR's trusted path routing needs to be equipped with CASTOR's device-side components, including CASTOR's TNDE and associated Trace Units (cf. subsection 6.2.8). As described in SR.1, it is crucial that CASTOR can establish trust in its device-side components as they form the TCB for the trust assessment of the network nodes and eventually enable the enforcement of trusted path routing policies. CASTOR strives for a minimal TCB size on the devices to decrease the risk of vulnerabilities and allow for easier code audits. However, as the network devices also host the (potentially complex) software stacks of one or multiple TNDIs that might be target of an attack, CASTOR's device-side TCB components need to be securely isolated. **Description**: CASTOR's device-side trusted computing base (TCB) components must be securely isolated from the remaining network device software stacks (e.g., the NOS and routing services of the TNDIs) in a remotely verifiable manner using hardware-based primitives. That is, the isolation must enable the creation of a trusted execution environment (TEE) based on an underlying RoT (as defined in SR.1) that provides strong (runtime) confidentiality and integrity guarantees for the code and data of CASTOR's device-side TCB components and can be remotely verified by the CASTOR orchestrator. That way, CASTOR can verify that its device-side TCB components can securely generate trustworthiness evidence of TNDIs and enforce the orchestrator's security policies for the trusted path routing. CASTOR's device-side TCB components shall be conceptually agnostic to the specific isolation primitives and RoT implementation (SR.1) to support a variety of devices from different vendors. **Remarks**: (1) CASTOR is responsible for isolating its TCB components from the TNDIs (e.g., vRouters) and enabling secure, independent management and trust assessment of each TNDI (e.g., see SR.6). If a device hosts multiple TNDIs, CASTOR assumes them to be isolatable from each other (potentially in an implementation- or device-specific way), such that TNDIs can not directly compromise each other. (2) CASTOR's device-side TCB components include the TNDE and associated Trace Units (cf. subsection 6.2.8). If additional components are included in the network device, TCB depends on the implementation-specific TCB of the underlying isolation primitives and root of trust (RoT). Depending on the device platform, the isolation could for instance be realized using a CPU TEE (e.g., Intel SGX and Arm TrustZone) or virtualization (e.g., containing TNDIs inside the VMs).
Connected to other requirements	SR.1, SR.6
KPIs	**Description**: Fine-grained control over enclave behaviour **Value**: $< 200\text{ms}$, including TCB layout planning, execution-flow design and dynamic linking of secure modules, clear specification of the interfaces between the trusted and non-trusted worlds, choice of secure-storage mechanisms, and configuration of the attestation mechanisms.

Table 9.4: SR.3 Secure Device Key Management with Platform and TNDI Binding Support

SR.3	
Title	**Secure Device Key Management with Platform and TNDI Binding Support**
Actors Involved	TNDE, TNDI
Type	Security Requirement

Description	**Background**: CASTOR needs to (a) establish trust into the device-side components (e.g., CASTOR TNDE) and (b) establish various communication channels with them in order to perform the local trust assessment (of TNDIs), receive device-side trustworthiness evidence for the global Trust Assessment Framework (TAF), and enforce configurations for the trusted path routing (TPR). Thus, is of paramount importance for all the associated authentication, attestation, and secure communication (channel) establishment tasks, the device side components to manage cryptographic keys in a secure way. **Description**: CASTOR's device-side trusted computing base (TCB) needs to support the secure derivation, management, and storage of cryptographic keys for the required authentication, attestation, and data protection operations. That is, CASTOR's TCB needs to support different types of keys for different cryptographic schemes, such as attestation keys for onboarding or evidence reporting, signing keys to generate verifiable runtime traces of TNDIs, or communication keys for secure communication with the CASTOR orchestrator. Furthermore, CASTOR's device-side TCB needs to provide flexibility in the key derivation process for each key, e.g., to achieve specific bindings. In particular, the device-side TCB needs to support the optional binding of keys to the device platform, CASTOR's device-side TCB components (e.g., the TNDE), and/or a device's onboarded TNDIs in order to allow for strong cryptographic authentication and provenance. **Remarks**: The details of the key derivation, binding, and storage operations might depend on implementation specifics of a network element's supported hardware Root of Trust (SR.1) and isolation mechanisms (SR.2).
Connected to other requirements	SR.1, SR.2, SR.4, SR.6, SR.7, SR.16, TNDE.R.1

	Description	**Value**
KPIs	Efficiency of key hierarchy construction with different types of keys	$< \mathbf{60ms}$ This KPI considers the construction of the appropriate key hierarchies comprising all of the necessary cryptographic primitives and keys, needed in order to support the entire lifecycle of a routing element (i.e., from its JOIN/ONBOARDING phases to its trust related evidence sharing)
	Types of keys to be supported	$\geq \mathbf{4}$ keys For instance these type of keys could be endorsement/identity key, storage key, attestation key, communication/session key and ORE key etc.

Table 9.5: SR.4 Runtime Configuration Integrity Check of Routing Plane Sw/Hw Stack

SR.4	
Title	**Runtime Configuration Integrity Check of Routing Plane Sw/Hw Stack**
Actors Involved	Attestation Source, TNDE, Trust Assessment Framework
Type	Security Requirement

Description	**Background**: A critical aspect in ensuring the trustworthiness of the routing plane - a key objective of the IETF Trusted Path Routing specification - lies in the evaluation of the security posture of the underlying Infrastructure Layer. This involves the validation of the configuration and software integrity of the underlying network elements, enabling the secure provisioning of services with certain security guarantees. However, as we move to higher levels of assurance, this validation may also span across the runtime operational assurance of a router (to be discussed in SR.13). Such guarantees are of paramount importance in communication channels in both the control plane and the data plane. Regarding the former case, it is crucial to safeguard the management interfaces between the orchestration layer and the routing elements. This allows for the secure and confidential configuration and management of the routing elements - and as a consequence the entire topology - throughout their lifecycle. At the same time, in the context of highly sensitive workloads, the latter case involves the exchange of certain security guarantees between the router elements of a ”Trusted Topology” - a term used throughout the IETF Trusted Path Routing specification - to ensure the posture of the provisioned data path. The aforementioned security guarantees on the configuration/software integrity as well as on the operational assurance of the infrastructure layer may vary depending on the policy associated with the high-level domain requirements and the threat model that is taken into account. These may span from design-time security properties (e.g., does a router have secure boot capabilities) up to the behavioural evaluation of critical regions of the routing services; the latter case is further analysed in SR.13. **Description**: The routing plane shall support the continuous verification of software components operating on routing elements in order to guarantee the integrity and trustworthiness of the entire infrastructure layer. By ensuring that only trusted and unmodified components are permitted to execute and interact with each other, the system can maintain a robust security posture. This is particularly crucial for high-criticality services that have a direct impact on the safety of the application users (e.g., road users in the context of the V2X-related use cases of CASTOR). The continuous evaluation of the configuration and software integrity of a routing element can be realized through attestation processes capable of verifying and reporting the trustworthiness of the entire routing plane: from a single network element to a routing path (see SR.14). Attestation enablers and cryptographic techniques have the ability to produce evidence that verifies the reliability of these components. Using this dynamic assessment enables the system to make informed decisions regarding the trustworthiness of individual software components. To achieve this trust characterization, it is essential that all elements are equipped with robust security enablers - e.g., hardware-based Root of Trust - to support attestation processes for securely collecting, storing, and reporting attestation claims to a verifier entity; be it a neighbouring routing element, or a controller entity residing at the orchestration layer. As part of the envisioned in-router TNDE framework, CASTOR aims to provide the necessary attestation enablers that are responsible for: • **Dynamically Assessing the security posture through Verifiable Evidence:** The Attestation Source, as part of the overall CASTOR’s TNDE platform, dynamically evaluates the integrity of software building blocks, allowing only trusted components to execute and interact with other entities in either the data-plane or the control-plane entities. • **Supporting the provisioning of different Levels of Assurance:** To support trustworthiness assessments across different domain requirements, CASTOR will explore the adoption of a ”Levels of Assurance” scheme to map the requirements captured in the path profile catalogue to the types of evidence that must be provided by the Attestation Source. Aligning with the high-level Level of Assurance concepts introduced by ETSI [60], this classification shall establish clear distinctions in trust levels, enabling informed decision-making and the application of appropriate security measures for each component. **Remarks**: The freshness of the attestation evidence - the same applies to other types of trustworthiness evidence - is one crucial aspect that needs to be reflected both in the attestation processes specified in the context of CASTOR as well as the trust assessment calculations. To accommodate any requirements on trust decisions under tight timing constraints, the attestation evidence may be cached. In this case, the CASTOR Trust Assessment Framework shall adjust the derived trust opinion accordingly so as to reflect the ageing of the available evidence in the trust quantification process (e.g., this may lead to a higher uncertainty in the derived trust opinions).

Connected to other requirements	SR.1, SR.2, SR.3, SR.8, SR.9, SR.11, SR.12, SR.13, SR.14	
	Description	**Value**
KPIs	Native Runtime performance for the execution of a local attestation process	$< 200\text{ms}$ considering the instantiation and execution of the attestation process outside of the CASTOR TNDE environment - i.e., outside of a TEE.
	Runtime performance for the execution of a local attestation process	$\leq 10\%$ **overhead**, considering the instantiation and execution of the attestation process as part of the TNDE environment - i.e., within a TEE. The outcome of this process is the trustworthiness evidence from the Attestation Source, which will be processed by the Trust Assessment Framework. (either the TNDE's Local TAF agent or the Global TAF at the orchestration layer).

Table 9.6: SR.5 Dynamic Security Function Placement

SR.5	
Title	**Dynamic Security Function Placement**
Actors Involved	TNDE, Service Orchestrator
Type	Security Requirement
	Background: In emerging B5G/6G environments, it is increasingly necessary for underlying network resources to support services of mixed criticality, each with distinct performance and trust requirements. Within the context of CASTOR, the infrastructure layer must therefore demonstrate specific capabilities that enable the establishment of routing paths with well-defined network and trust guarantees. In Chapter 6, it becomes clear that the CASTOR TNDE constitutes the core in-router enabler for providing the trust-related guarantees to the Orchestration Layer so that it can make informed decisions on (through the Optimization Engine) whether to take into consideration specific network segments as part of the traffic engineering process. Based on the trustworthiness evidence that can be collected during runtime, the Global TAF is able to evaluate whether or not a router instance (i.e., TNDI) can act as a Path Element in a path profile with specific trust requirements as specified by the network operator. Through the enforcement of a Trust Policy, the Orchestration Layer is able to instruct a TNDE to provide guarantees that it can satisfy a particular Level of Assurance (see SR.4). In other words, this introduces the need for re-configuring the underlying Trusted Computing Base in order to securely monitor specific regions of the target TNDI environment and allow the construction of particular trustworthiness claims associated with the operator's requirements. For example, in the case of commodity workflows there may be no requirement with respect to the trust capabilities of a TNDI - i.e., thus, no Trust Policy needs to be activated. However, when considering sensitive workflows that require strong guarantees of confidentiality and integrity in the provisioned routing paths, this creates the need to securely extract trustworthiness evidence from the underlying routers. Such evidence may range from simple attestation quotes ensuring the secure launch of a TNDI (e.g., a network element has securely booted) or more complex attestation reports that provide runtime guarantees on the runtime integrity of critical router services. **Description**: Given the diversity of the trust characteristics offered in a service catalogue, and the dynamic nature of the routing plane (e.g., as new services are being fulfilled), it becomes essential that the security mechanisms can be dynamically placed, ensuring an optimal use of the resources required to achieve a required Level of Assurance. Specifically, due to the dynamic nature of the routing plane, existing guarantees may not always be sufficient to ensure that the previously computed ATL values continue to satisfy the established RTL constraints. There are multiple reasons that may lead to this situation. The overall risk posture may worsen—for example, following the discovery of a new zero-day vulnerability—requiring the router to provide stronger trustworthiness evidence to maintain its Level of Assurance.

	Alternatively, the domain operator may choose to enhance the trustworthiness capabilities of a specific router to achieve a higher Level of Assurance, enabling it to support services with more demanding path profile requirements. In the context of the CASTOR framework, the ability to dynamically provision the capabilities of the TNDE consists of updating the raw traces that need to be reported to the Tracing Hub, reconfiguring the Trust Sources - namely the Attestation Source and the FSM - as well as configuring the Local TAF agent calculations. This allows the latter agent to yield local trustworthiness claims that can be shared with the Global TAF at the Orchestration Layer.
Connected to other requirements	SR.3, SR.4, SR.5, SR.7, OSS.R.1
KPIs	**Description** / **Value**
	Security Control Enforcement Runtime Performance: < 100 **ms** focusing on the verifiable update of the Key Restriction Usage Policy binded to the additional evidence monitored for the verification of the newly activated security controls.
	Granularity of level of assurance that can be achieved by the CASTOR TCB: 5. CASTOR will adopt and build on top of the classification of LoA specified by ETSI [60] in the context of compute continuums for mapping specific LoAs to define path profiles representing varying levels of network and trust metrics.

Table 9.7: SR.6 Onboarding of Network Devices and their TNDIs into CASTOR

SR.6	
Title	**Onboarding of Network Devices and their TNDIs into CASTOR**
Actors Involved	TNDE, TNDI, Service Orchestrator
Type	Security Requirement
Description	**Background**: CASTOR follows a zero-trust model w.r.t. the network devices participating in the trusted path routing. That is, before CASTOR allows the TNDIs of a network device to participate, the CASTOR orchestrator needs to establish trust into the device TCB to ensure that it is equipped with CASTOR's device-side components and provides the necessary security capabilities. Furthermore, the orchestrator needs to configure the TNDE and TNDIs to continuously assess the trust level of TNDIs to check that they satisfy the (minimum) required trust level, and enforce the trusted path routing policies on the TNDIs. That is, CASTOR requires a secure process to let network devices join the CASTOR domain and to onboard their TNDIs. **Description**: CASTOR's device-side TCB components must enable the CASTOR orchestrator to securely join network devices into its network domain (join phase) and perform a trustworthy onboarding of the device TNDIs that are supposed to take part in the trusted path routing (onboarding phase). If a network device provides multiple TNDIs (e.g., vRouter instances), the TNDE needs to allow for an onboarding and configuration of each TNDI independent of the other TNDIs. During the join phase, the CASTOR device-side TNDE must enable the orchestrator to securely attest the device-side TCB components and their protection. Furthermore, the TNDE must enable the orchestrator to securely start the onboarding process of the device TNDIs and allow for the generation of the required attestation and communication keys for the TNDIs (related to SR.3). During the onboarding phase, the CASTOR TNDE must enable the orchestrator to securely assess (attest) a TNDI's TCB and deploy configurations for the evidence tracing and trust assessment process (trust policy) of that TNDI. Furthermore, the TNDE must enable the establishment of cryptographic keys and secure channels for the evidence authentication and reporting (cf. SR.7 and SR.9). **Remarks**: (1) After a device has joined and an associated TNDI has been onboarded to a CASTOR domain, the orchestrator is supposed to be able to securely assess the trust level of that TNDI based on the traces, evidence, and ATL values exposed by the TNDE for that TNDI. (2) A TNDI shall only be actively onboarded to a single CASTOR domain at a time (see TNDE.R.1). (3) The onboarding process and future re-configurations of an onboarded TNDI are supposed to be controlled via a secure TNDI-SP control channel (see SR.7).
Connected to other requirements	SR.1, SR.2, SR.3, SR.7, SR.9, TNDE.R.1, TNDE.R.2

	Description	Value
KPIs	Execution of CASTOR JOIN Protocol	$\leq \mathbf{220ms}$ *Note: This refers to the JOIN phase of the remote attestation process of CASTOR's TNDE, assuming that the TNDE is up and running, and excluding network latency and TNDI-SP control channel establishment.*
	Execution of CASTOR ONBOARDING Protocol.	$< \mathbf{900ms}$ Including the following processes: • Attestation of TNDI, • establishment of CASTOR-related keys, • setup of Tracing Hub (with at least one Trace Unit), $\leq \mathbf{1.3s}$, including the authenticated querying of the appropriate Trust Policy through the CASTOR DLT. *Note: The KPI values refer to the core attestation process of the TNDI to be on-boarded, excluding any network latency. The evaluated assumes a successful JOIN process and the establishment of the necessary TNDI-SP control channel. The specified KPIs do not capture the actions that take place after a successful completion of the ONBOARDING protocol*: (a) the establishment of the relevant TNDI-SP data channels for evidence transmission, (b) the issuance of the Conformity Certificate for the newly on-boarded TNDI in the topology, or (c) the exchange of trust-related evidence between the TNDIs.

Table 9.8: SR.7 Secure E2E CASTOR-to-Device Control and Data Channels

SR.7	
Title	**Secure E2E CASTOR-to-Device Control and Data Channels**
Actors Involved	CASTOR orchestrator services, TNDE, TNDI
Type	Security Requirement
	Background: For the CASTOR orchestrator be able to calculate and provision trust-aware network paths, it must be able to securely communicate with the network devices of its topology, specifically the CASTOR device-side TCB components (e.g., TNDE). The orchestrator requires a control channel to add network devices to its topology (cf. join and onboarding phases in SR.6) and push configurations (e.g., trust policies) to enforce trusted path routing. Furthermore, the CASTOR framework requires data channels to efficiently exchange security claims required for the trust assessment of the onboarded TNDIs (cf. SR.9 for details). However, as CASTOR follows a zero-trust model, CASTOR requires these channels to provide strong mutual authentication and protection of the associated control and data messages against man-in-the-middle attackers (e.g., on-path network attackers, untrusted on-device software). **Description**: CASTOR must support secure E2E control and data channels (TNDI-SP channels, cf. subsection 6.2.8) between the device TNDEs and the CASTOR upper layer services. The channels must provide end-to-end encryption guarantees, providing confidentiality, integrity, authentication, and replay protection. The communication and authentication keys must be securely established between the orchestrator and TNDE components, potentially binding them to the TNDE platform and/or TNDIs associated with the channels (cf. SR.3). The control channel must support strong attestation-based authentication of the device-side TCB components (especially the TNDE) and support secure control messages for configuring and enforcing security policies, e.g., for the runtime tracing of TNDIs, their trust assessment (trust policies), and the enforcement of trusted paths. The control channel might allow the TNDE to authenticate/identify the CASTOR orchestrator which has onboarded a given TNDI (mutual authentication). The data channels must support mutual authentication with optional attestation support where applicable (e.g., CASTOR DLT) and support secure and efficient transportation of runtime traces and trustworthiness evidence (cf. SR.9 for details). They should be configurable to provide different transport and security tradeoffs depending on the destination endpoints and transported data types (Section 6.2.9.2). Both channel types must provide reliable transport, i.e., take care of message reordering or drops.

Description	**Remarks**: (1) This requirement refers to CASTOR's envisioned TNDI-SP protocol which is intended to provide TNDE/TNDI-bound control and data channels. The control channels are intended for communication from the orchestrator to the TNDEs and the data channels for communication between TNDEs and the CASTOR Global TAF or DLT. (2) Attestation support is considered for the TNDEs based on the underlying device HW RoT (SR.1 and SR.2), for the TNDIs as enabled by the TNDEs, and potentially for the secure oracle of the CASTOR DLT (e.g., Intel SGX-based Phala).	
Connected to other requirements	SR.1, SR.2, SR.3, SR.6, SR.9, TNDE.R.2	
KPIs	**Description**	**Value**
	Establishment of a TNDI-SP *control* channel	$<= 800$ **ms**
	Establishment of a TNDI-SP *data* channel	$<= 2$ **sec**

Table 9.9: SR.8 Secure and Efficient Cryptography

SR.8	
Title	**Secure and Efficient Cryptography**
Actors Involved	TNDE, Trust Assessment Framework, CASTOR DLT
Type	Security Requirement
Description	**Background**: A fundamental principle of traffic engineering is to ensure that data is delivered reliably and within acceptable time bounds, thereby optimizing overall network performance and user experience. To maintain optimal Quality of Service (QoS) metrics — such as latency, or throughput — the routing operations shall incur minimal processing and communication overhead. This becomes even more critical in domains involving highly sensitive or real-time applications, such as the ones envisioned in the CASTOR use cases: from UAV operations to V2X communications. In these contexts, routing performance directly influences operational safety, reliability, and user experience. Consequently, while the incorporation of security measures is essential to ensure trusted path routing, it is necessary to prioritise the design of schemes that enhance security without compromising the responsiveness or stability of the routing plane, nor the service overlays that depend upon it. The aforementioned security measures relate to the protection of the management interfaces (i.e., from network elements to the upper control-plane layers) but also on the interactions between network elements. Regarding the former part, this involves the confidentiality and integrity of the communication channels, ensuring the secure management of network elements throughout their lifecycle. Regarding the latter aspect, as we move towards trusted path routing, it is essential that the appropriate guarantees — in the form of cryptographic proofs — are efficiently propagated across the routing plane to establish and maintain a Trusted Network. Consequently, it is imperative to evaluate and adapt existing schemes to meet the diverse requirements of the Compute Continuum, taking into account both intra- and inter-domain scenarios. **Description**: The employed cryptographic mechanisms shall be sound, secure, and lightweight. Through the investigation and proposal of new efficient crypto schemes, CASTOR envisions to unlock all security requirements that enable low-level operations such as authentication of entities, secure collection and reporting of trustworthiness evidence, but also establishing secure and confidential communication channels. At the same time, these operations will help establish core security requirements for the trust-aware service provisioning in CASTOR; from the secure on-boarding of new TNDIs up to the continuous monitoring and evaluation of the operational assurance of a TNDI.

Description	Finally, given the diverse operations across the Compute Continuum—for example, from the establishment of secure and authenticated channels between a TNDE and the Global TAF at the orchestration layer, to the in-router trust evaluation of a TNDI's integrity — it becomes evident that cryptographic mechanisms must satisfy different performance requirements. Therefore, the cryptographic primitives envisioned for design and implementation within the CASTOR framework will be evaluated across various metrics, including their runtime performance in key operations as well as the characteristics of the generated cryptographic outputs (e.g., issued certificates or digital signatures). Based also on the security requirements presented in this Chapter, CASTOR envisions to investigate the design of lightweight crypto for the following operations: • **enable the establishment of secure communication channels** from TNDE entities to the Orchestration Layer. (see SR.7) • **enable the provision of aggregated attestation proofs** that characterize the state of an entire path of router elements. For example, this may allow the secure and efficient collection of trustworthiness evidence, enabling the Global TAF to form trust opinions on path-level propositions (SR.12). • **share trust capabilities** characterizing the infrastructure layer **to external entities**, without disclosing any redundant or sensitive information to external verifiers. In this context, CASTOR will evaluate Order Revealing Encryption (ORE) schemes to protect trustworthiness claims by disclosing e.g., only the minimum trustworthiness value representing a domain-internal trusted segment. This value is paired with zero-knowledge assertions that ensure the hidden trustworthiness values of the other nodes or links within the infrastructure layer maintain their correct relative order, thereby preserving confidentiality while enabling verifiable trust relationships (SR.14). **Remarks**: Adhering to zero-trust principles, all cryptographic mechanisms in CASTOR shall be supported by specific RoT capabilities (SR.1), enabling the formation of the CASTOR TCB (SR.2) — a core security requirement for provisioning the in-router TNDE ecosystem. Given that the envisioned cryptographic schemes will be instantiated across the CC, it is necessary to account for the heterogeneity of the underlying host environments and their diverse RoT capabilities. Therefore, one of CASTOR's key priorities is to ensure cryptographic agility in its designs. Considering the varying capabilities of network elements — routers from different vendors may integrate distinct HW/SW security modules — the proposed schemes will be designed to remain interoperable and HW-agnostic.
Connected to other requirements	SR.1, SR.2, SR.7, SR.9, SR.10, SR.12, SR.14

KPIs	**Description**	**Value**
	Cryptographic Throughput	≥ 100 **operations** per seccond (e.g., sign/verify, encryption/decryption)
	Crypto agility: Number of crypto primitives that can be supported for different router modalities	≥ 2 crypto primitives that can be supported (e.g., vRouter images, HW router equipment)
	Computational overhead	$\leq 30\%$ **overhead** introduced due to the CPU cycles for the crypto operation execution. This will include the detailed benchmarking of all crypto operations, needed to support the CASTOR trust assessment considering the existence (or not) of RoT guarantees.

Table 9.10: SR.9 Secure Reporting of Traces and Trustworthiness Data

SR.9	
Title	**Secure Reporting of Traces and Trustworthiness Data**
Actors Involved	TNDE
Type	Security Requirement

Description	**Background**: CASTOR's trust-aware path provisioning requires the CASTOR framework to continuously assess and monitor the trust levels of all network nodes (or more precisely: the onboarded TNDIs). To do so, CASTOR requires a secure way to share trustworthiness data of TNDIs within the CASTOR domain. That is, CASTOR needs to securely distribute runtime traces, evidence, and ATLs generated by CASTOR's device-side components (esp. the TNDE) locally across the TCB components (e.g., traces passed to Trust Sources) and remotely to CASTOR's upper layer components (especially Gloabl TAF and DLT). That way, CASTOR's upper layer services can securely collect the required data to perform the global trust assessment of the network topology, provision trust-aware paths, and support auditing of suspicious device behaviour on an incident. In addition, CASTOR's TNDIs might need to exchange evidence to attest each other to bootstrap secure communication links (see details in SR.10). In all cases, CASTOR needs to preserve the integrity and authenticity guarantees of the exchanged data to allow for their verification. **Description**: CASTOR must support the secure reporting of runtime traces and trustworthiness data of a TNDI (evidence, ATLs): 1. across CASTOR's device-side TCB components (TNDE, Tracing Units), 2. to upper layer CASTOR services of a TNDI's CASTOR domain (Global TAF), and 3. towards neighbouring TNDIs. CASTOR's device-side TCB components must enable the verification of the integrity and authenticity of their generated TNDI traces and trustworthiness data. The reporting must preserve these integrity and authentication guarantees, allow for freshness guarantees for the trustworthiness evidence, and provide confidentiality against external entities. The sharing shall be configurable to provide different transport and/or security tradeoffs depending on the type of reported data (traces vs. evidence) and destination endpoint (SR.7, SR.10, and section 6.2.9.2). **Remarks**: (1) The reporting to upper-layer CASTOR components is planned via TNDI-SP data channels (see SR.7). (2) Within the TNDE, the traces are passed to the Trust Sources and the TN-DSM, evidence is passed to the Local TAF Agent and the TN-DSM, and the ATLs and trust reports are passed to the TN-DSM. (3) Traces, evidence, ATLs and trust reports are remotely shared with upper layer CASTOR components. (4) TNDIs might exchange their boot-time or even runtime evidence with neighboring TNDIs to establish trustworthy links (see SR.10). If sharing with TNDIs outside a TNDI's CASTOR domain needs to be supported, the orchestrator must configure it to not leak sensitive data (e.g., not share traces, selective evidence sharing).
Connected to other requirements	SR.1, SR.3, SR.7, SR.10, SR.11, SR.16, TAF.R.1, TAF.R.4

	Description	**Value**
KPIs	**Data curation**: pre-processing of traces by Tracing Hub (or its Trace Units) for secure sharing with internal Trust Sources (i.e., FSM and Attestation sources)	$\leq \mathbf{10x(length\_of\_critical\_path)}$ *Note:* This is highly dependent on the complexity of the traced binary which translates to different volumes of data to be curated prior to runtime verification.
	Trace Authentication: Tracing Hub authenticating the traces as part of secure sharing with internal Trust Sources	$\leq \mathbf{20ms}$ *for the TNDE's Tracing Hub to validate the authenticity of the traces captured by a Trace Unit.*
	Secure sharing of trustworthiness claims (including the time needed for the construction and signing of the trustworthiness claims) outside of the TNDE	$\leq \mathbf{800ms}$, out of which 80ms will be needed for the construction of fresh TNDI-SP session keys. Network latency is excluded from this value. $\leq \mathbf{700}$ **Bytes** size, assuming one trust property of interest. *Note: Size of trustworthiness claim is dependent on the number of trust properties that are reported.*

Table 9.11: SR.10 Secure Link Establishment between TNDIs

SR.10	
Title	**Secure Link Establishment between TNDIs**
Actors Involved	TNDE, TNDI, Service Orchestrator
Type	Security Requirement

Description	**Background**: CASTOR's goal is to enable trusted path routing. Therefore, CASTOR continuously assesses the trust level of the network topology in order to provision trust-aware paths that satisfy requested security and performance requirements. To allow secure enforcement of the trusted routing paths, neighbouring TNDIs of the CASTOR domain must be able to securely exchange control and data plane network traffic, i.e., routing information and the actual application traffic. Otherwise, network attackers or rogue TNDIs might be able to leak or tamper with traffic, breaking the required security guarantees. Therefore, CASTOR requires a secure way for its TNDIs to establish trust in each other and exchange traffic. In addition, if trusted inter-domain path routing is required, TNDIs might need to securely communicate to TNDIs of other CASTOR domains. While the IETF TPR [27] proposes incorporating TPM-based measurements into secure link protocols (e.g., MACsec), the current proposal neither considers runtime evidence nor the incorporation into a trusted TNDI onboarding process, unlike CASTOR. **Description**: CASTOR's device-side components need to enable the establishment of trustworthy links between neighbouring TNDIs, allowing the secure exchange of network traffic. The link establishment needs to allow TNDIs to establish trust into each other based on the secure exchange of load-time and runtime evidence information, collected via CASTOR's device-side TCB components (esp. TNDE). The types of exchanged evidence need to be configurable to account for differences in available evidence (e.g., boot-up vs. runtime, different device vendors) and to prevent leakage of sensitive evidence towards TNDIs of different CASTOR domains. The links should support a periodic re-exchange of fresh runtime evidence in order to allow for the continuos re-establishment of a link's trustworthiness if requested by the CASTOR orchestrator(s). The established links are supposed to provide E2E protection for the control and data plane network traffic routed through them, including confidentiality, integrity, (evidence-augmented) authentication, and replay protection guarantees. **Remarks**: (1) The establishment of secure links between neighbouring TNDIs of the same CASTOR domain is supposed to happen as part of the TNDI onboarding process (see SR.6), or following upon it. (2) If secure links need to be established between TNDIs of different CASTOR domains, the orchestrator shall configure sharing policies that prevent leakage of sensitive evidence (see SR.9).
Connected to other requirements	SR.1, SR.3, SR.6, SR.9

	Description	**Value**
KPIs	Issuance of Conformity Certificate during TNDI onboarding	$\leq \mathbf{200ms}$ for the realization of the Local TAF evaluating the onboarding requirements as expressed in the available Trust Policy. $\leq \mathbf{600ms}$ *Note:* This includes the construction of the Conformity Certificate, once the onboarding has been successfully carried out. This includes the execution of the trust enablers to provide the necessary evidence on the static properties of the underlying router element.
	Selective Disclosure of required properties for the E2E TNDI link establishment.	$\leq \mathbf{150ms}$, this includes the construction of the Verifiable Presentation over the superset of evaluated properties/attributes (mix of static properties and runtime configuration and behavioural properties).

Table 9.12: SR.11 Secure Runtime Tracing Support of TNDIs

SR.11	
Title	**Secure Runtime Tracing Support of TNDIs**
Actors Involved	TNDE, TNDI
Type	Security Requirement

Description	**Background**: CASTOR's trust-aware path provisioning requires a continuous trust assessment of the network topology (i.e., the onboarded TNDIs). CASTOR's global trust assessment is based on evidence and local trust level calculations of each TNDI, provided by CASTOR's TNDE (cf. SR.9). However, to allow for the generation of trustworthiness evidence, CASTOR's device-side components need to provide secure runtime tracing capabilities forming the basis of the evidence. In addition, as the required traces depend on the expected types of evidence that need to be generated by the Trust Sources (TAF.R.4), CASTOR requires flexibility w.r.t. the deployed Trace Units and support for dynamic re-configuration (re-programmability) of the tracing. Such flexibility enables the CASTOR orchestrator to choose and adjust Trace Units based on the required level of granularity, performance, and security (threat model) for the runtime tracing of TNDIs. **Description**: CASTOR's device-side TCB components must enable secure tracing of a TNDI's configurational and behavioural runtime information. The tracing must be (re-)configurable by the CASTOR orchestrator (see SR.6, SR.7, OSS.R.1) to allow taking the requirements for the evidence generation by different Trust Sources into account. That is, the CASTOR orchestrator should be able to change what runtime information is traced, e.g., by asking the TNDE to reconfigure active Trace Units or deploy additional Trace Units with different granularity, performance, or security tradeoffs (see subsection 6.2.9). The tracing must be secure within the boundaries of the Trace Units' threat model(s). That is, attackers within the scope of the threat model must neither be able to disable nor tamper with the runtime tracing mechanisms or the generated traces. Furthermore, a compromised TNDI must not be able to tamper with the tracing configurations or the operation of the TNDE. **Remarks**: (1) The Tracing Hub is part of CASTOR's device-side TNDE and is responsible for managing the trace collection. It is strongly isolated from the traced TNDIs (see SR.2). (2) CASTOR's Tracing Hub supports multiple different Trace Units as part of its multi-level tracing architecture (see subsection 6.2.9). Each Trace Unit can provide different tradeoffs (level of granularity, performance, security), allowing for differences in their threat models. For instance, a memory inspection-based Trace Unit might be strongly isolated from the TNDI, while a kernel-level Trace Unit might or might not be integrated in the TNDI's OS stack depending on the defined threat model (see subsection 6.2.9).
Connected to other requirements	SR.2, SR.6, SR.7, SR.9, SR.13, TAF.R.4, OSS.R.1

	Description	**Value**
KPIs	Support different Trace Units for the verifiable extraction of varying device-state attributes	$\geq \mathbf{2}$ for the verifiable measurement during runtime of the configuration integrity of a TNDI and the verifiable extraction of the execution flow of a critical function path.
	Control flow extraction (with an OS-level Trace Unit)	$\leq \mathbf{5}\%$ overhead on the normal operation of the target binary to be traced. *Note: This is usually in the order of nano seconds unless a complex binary (e.g., a virtual router containerized image) needs to be traced.*
	Configuration measurement extraction performance overhead	$< \mathbf{8} \times (\mathbf{length\_of\_target\_function\_critical\_path})$. *Length of critical path is equivalent to the number of functions, whose configuration needs to be monitored dynamically.*

Table 9.13: SR.12 Composition of Trustworthiness Evidence

	SR.12
Title	**Composition of Trustworthiness Evidence (SURREY)**
Actors Involved	TNDE
Type	Security Requirement

Description	**Background**: In the CASTOR TAF, a key challenge is to extend trust evaluations from the level of individual nodes to entire paths. This can be approached by combining the trust opinions of all nodes along a path, or alternatively, by gathering consolidated trustworthiness evidence directly at the path level. The latter approach enables the construction of atomic trust relationships tied explicitly to path-level evidence. This aligns with the IETF NASR concept [97], where NASR can assist operators in attesting to an orchestrated path and providing verifiable forwarding proofs, thereby allowing clients or authorities to audit the forwarding process. **Description**: The routing plane shall provide verifiable evidence that attests to the validity of a traffic engineering policy, ensuring that such attestation is performed efficiently. Within a segment (domain), each router possesses the capability to autonomously generate a cryptographically signed claim reflecting its current trust level and integrity status. The orchestrator, acting as a centralized trust authority, can then perform a collective attestation of the entire segment of routers. This process yields verifiable evidence attesting to the overall integrity of the constructed path. In different domains, the attested claims are inherently designed to be verifiable by and linked to those from neighbouring segments. Consequently, routers at segment boundaries may selectively verify different subsets of a combined claim set, a process governed by dynamically defined attestation policies that dictate the required scope and depth of verification. CASTOR envisions addressing a challenge. The CASTOR Global TAF must be capable of evaluating composite propositions that enable the elevation of ATL values from low-level (atomic) trust propositions associated with network elements to ATL values at the path or even domain level. This introduces several limitations when attempting to identify suitable logical expressions that can accurately characterize the trustworthiness of complex trust objects. Deliverable D4.1 analyzes the challenge involved in achieving an overarching Trust Assessment Framework. **Remarks**: Based on the descriptions, we can conclude some points (1) there are multiple routers (provers) (2) Cryptographically signed claims (proofs) from different routers (provers) can be aggregated, which can be verified by any entity. (3) In inter-domain scenario, the confidentiality of evidence should be considered. Further, there are some potential solutions: (1) BLS signature is aggregatable, which can be used for composable attestation (2) The Merkle tree is a structure, which can be used to aggregate signatures maintained by a trusted third party
Connected to other requirements	SR.14
KPIs	**Description**: Runtime performance of the verification of a composite attestation report **Value**: $\leq \mathbf{40ms}$, considering one attestation report involving at most 100 signers.

Table 9.14: SR.13 Runtime Operational Assurance and Process Execution Integrity Checks

SR.13	
Title	**Runtime Operational Assurance and Process Execution Integrity Checks**
Actors Involved	Finite State Machine (FSM), TNDE
Type	Security Requirement

Description	**Background**: Capturing runtime behaviour is essential because static verification or pre-deployment testing alone cannot guarantee the ongoing integrity of a device once it is exposed to real-world conditions and potential attacks. By continuously or periodically monitoring the execution flow and operational state of a router, it becomes possible to detect and explain such deviations from expected behaviour that may indicate compromise, misconfiguration, or malicious activity (FSM.R.1, FSM.R.2). Implementing such runtime monitoring, however, presents several challenges, especially if the hardware constraints together with the performance requirements associated to the standard operational services are taken into account. The solution requires tracing and monitoring mechanisms capable of accurately capture and evaluate the execution flow of critical router components with minimal performance impact, as well as ensure an efficient processing of these traces to detect any violations against nominal operation in near real time. **Description**: Deploying runtime behavioural monitoring solutions is a recommended approach to ensure continuous verification of the router's integrity and its operational correctness, even after its deployment. This proactive approach not only strengthens the security of the router itself but also enhances the overall security of the network in which the router is operating by ensuring that itself and its services remain trustworthy throughout their lifecycle. This acts as an additional Trust Source (along with the runtime configuration integrity verification presented in SR.4) coalescing macroscopic behavioural assessment of a router's critical execution path with Finite State Machine models to enable efficient control flow integrity verification. More details about the FSM approach taken in CASTOR can be found in Section 6.2.6. **Remarks**: To tackle all these points, Finite State Machines (FSMs) models have been chosen to represent the nominal operational behaviour of each router. These will be trained taking in consideration the device architecture, its requirements and the specific threat model of its interest (FSM.R.1); will provide augmented results including clarifications on the discrepancies identified (FSM.R.2); and ensure that their evaluation is done in near real time as well as their overhead will not impact the normal operations of the router. For clarification, the KPIs below focus only on the necessary operations to be performed so that the the FSM models can operate effectively. This excludes any integrity, authenticity or validity checks to be done on the traces received by the tracing layer. The KPIs are also designed considering (i) model evaluations based on an optimised set of traced information, i.e., only relevant information are shared to the FSM models, and (ii) the FSM functionalities are considered to be executed outside of a TEE.
Connected to other requirements	FSM.R.1, FSM.R.2
KPIs	**Description** / **Value** Time taken to parse and transform all the collected traces data / < 2 seconds Time taken for the FSM model to perform its analysis on the parsed data / < 1 second Attestation solution leverages the TEE capabilities / Yes/No

Table 9.15: SR.14 Ordering of Attestation Evidence

SR.14	
Title	**Ordering of Trustworthiness Claims**
Actors Involved	Trust Exposure Layer, Trust Assessment Framework
Type	Security Requirement

Description	**Background**: The exchange of trustworthiness evidence from network elements (i.e., TNDIs) constitutes the corner stone for enabling the construction of a trust plane in a single domain. The availability of such evidence - either processed locally in the routers or centrally at the orchestration layer - unlocks the assessment of the trustworthiness of the routing plane. Consequently, this enables trust evaluation to be elevated beyond the node level (e.g., assessing the integrity of a router) to the link, segment, or even path level. However, this path-level trust characterization becomes challenging, considering end-to-end service provisioning -i.e., achieving service connectivity across multiple domains. In this context, it is not as straightforward to construct an overarching trust plane as there is a limitation to the detail (and amount) of trustworthiness evidence that can be exchanged between different domains. This necessitates the exchange of specific trust-related data between domains, while ensuring that no sensitive information regarding the network topology or router characteristics is disclosed. **Description**: Each network operator shall be able to expose its trust capabilities to neighbouring domains without disclosing any sensitive information with respect to the underlying topology. CASTOR envisions to provide these exposure capabilities through the Trust Exposure Layer in the form of Trust Summaries. The Trust Summary shall allow a verifying domain - domain B - to get the necessary (cryptographic) guarantees that the domain in question - say domain A - has not dropped below a minimum ATL value. Therefore, domain B can receive with minimum ATL value of domain A without the latter entity disclosing any redundant information to the former one. On this front, CASTOR envisions addressing two parallel challenges. First, the CASTOR Global TAF must be capable of evaluating composite propositions that enable the elevation of ATL values from low-level (atomic) trust propositions associated with network elements to ATL values at the path or even domain level. This introduces several limitations when attempting to identify suitable logical expressions that can accurately characterize the trustworthiness of complex trust objects. Deliverable D4.1 analyzes the challenges involved in achieving an overarching Trust Assessment Framework. Second, the ATL scores on the composite path- or domain-level trust propositions are recorded on the CASTOR DLT - through CASTOR's Secure Oracle element. These ATL values need to be exposed by the Trust Exposure Layer to other authorized stakeholders (e.g., neighbouring domains) in a way that safeguards any sensitive information from the evaluated domain. In this context, CASTOR will explore the use of Order-Revealing Encryption (ORE) schemes, which enables the encoding of Trust Summaries in a manner that allows external entities to obtain meaningful insights into a domain's trust capabilities without revealing any information that could lead to profiling or the full disclosure of the domain's characteristics. **Remarks**: (1) The minimum ATL for each domain should be achieved (2) The order of different ATLs can be compared while maintaining confidentiality of the domain under evaluation. (3) In existing ORE schemes, we can consider about multi-user settings of ORE scheme as potential solutions.
Connected to other requirements	SR.9

Description	Value
Time to generate comparison tokens	$\leq$ **450 ms** for 100000 users.
Time to compare	$\leq$200 **ms**, referring bit-by-bit comparison of a 64-bit message corresponding to ATL values that characterize a specific property in a domain.

Table 9.16: SR.15 Secure Data Handling and Provenance

SR.15	
Title	**Secure Data Handling and Provenance**
Actors Involved	TNDE, Trust Assessment Framework, CASTOR DLT
Type	Security Requirement

Background: As mentioned in SR.8, one crucial security requirement in the context of traffic engineering policy provisioning, involves the consideration of the trustworthiness of the underlying routing plane. Therefore, it becomes crucial to establishing mechanisms that provide runtime evidence for trust assessment and ensure appropriate data associations. In addition, as per SR.7 and SR.9, these mechanisms shall enable authenticated entities to trace trustworthiness evidence back to their source of origin, allowing their association with the corresponding trust relationships under evaluation. Such authenticated entities may belong to the control-plane of a domain in an effort to analyze the available evidence and pinpoint a possible source of compromise. At the same, it is also possible that external authorized stakeholders may want to access a snippet of evidence that is associated to a specific network element or service as part of an auditing process or in order to identify the root of cause that led to a particular defect. All in all, it becomes evident that the heterogeneous environment of an infrastructure layer and the diversity of the use cases for accessing trustworthiness evidence - or even raw traces - of particular topology regions of interest introduces the need for enhanced stronger guarantees on the overall data provenance and secure data handling.

Description: Only authenticated and authorised entities and components should be able to link the evidence back to the data source. The integration of appropriate cryptographic primitives allows for the deployment of controlled linkability, safeguarding the privacy of entities while maintaining trust mechanisms. Apart from employing cryptographic schemes to assess the trustworthiness of the routing plane, CASTOR envisions to leverage mechanisms for providing verifiable evidence that critical trust-related data have been processed by certified applications. This further enhances the trustworthiness of the data and ensures that processing occurs only through authorised channels, instilling confidence in the network domain and its offered services.

To further illustrate the need for ensuring data provenance and secure handling guarantees, we consider an example stemming from the use cases. In the context of unmanned airspace operations (see Section 8.2), a network operator shall ensure network connectivity that meets stringent network performance and trust requirements in order to support critical workloads. For instance, radar traffic must traverse a highly secure and confidential network path to reach the U-Space Service Supplier connected to the radar zone network. In this context, the CASTOR framework ensures the fulfilment and continuous assurance of the radar application service. If a violation occurs in any of the network elements along the provisioned path, trustworthiness evidence - signalling the detected failure - is reported by the CASTOR TNDE artifacts. Consequently, it becomes essential that authorized entities — both internal (e.g., the Global TAF) and external (e.g., the central airspace authority consuming radar data) — have access to these provenance updates. Such updates must be provided at an appropriate level of abstraction, reflecting the current trustworthiness of the traffic without exposing unnecessary or even sensitive detail.

Based on this example, CASTOR envisions to explore the following cases:

- **Secure processing of trustworthiness claims as a provenance assertion characterizing the trust posture of a topology**: Provide guarantees to an authorized interested party, that the processing of trust-related data related to a particular domain or service is done in a secure and confidential manner. In CASTOR, it is the Trust Exposure Layer which is able to securely process the trust capabilities of a specific domain or service from the CASTOR Blockhain. For this processing CASTOR aims to leverage computational resources that possess the necessary integrity and confidentiality guarantees - e.g., worker nodes running in hardware isolation through the employment of TEEs. This is essential for sharing trust-related provenance about the network guarantees of a service to external authorized stakeholders (see Section 8.2), but also to provide a compliance report ensuring authorized auditors that the established SSLAs are respected throughout the lifespan of a service (see Section 8.5).
- **Accessing historical trustworthiness evidence and raw traces for post processing analysis**: Depending on the domain policies, raw traces and trustworthiness evidence coming from the in-router TNDE artifacts are securely recorded on the CASTOR DLT. In CASTOR, we plan to leverage a novel Attribute-Based Signcryption (ABSC) scheme that enables flexible policy provisioning. This will allow TNDEs to report critical data confidentially, ensuring that only authenticated and authorized entities—those possessing the required attributes—are able to access the data. For example, router vendors may want to access specific set of raw traces as diagnostic information to further analyze any defects of their equipment.

Connected to other requirements	SR.7, SR.8, SR.9	
KPIs	**Description**	**Value**
	Access control policy enforcement on trustworthiness evidence	$< \mathbf{2}$ **sec** This is important in the case where only authorized users can access historical trustworthiness evidence for post-processing analysis in case of an identified incident (e.g., failure in the router firmware which was caught by the CASTOR Attestation enablers).
	Network domain privacy exposure due to transmission of trust capabilities to external requesters	**FALSE** External requesters accessing the trust capabilities of a domain through the Trust Exposure Layer shall not gain any sensitive information that could potentially leak the topology characteristics of the domain's infrastructure layer.

9.2 Functional and Non-Functional Requirements

9.2.1 Trust Assessment Requirements

Table 9.17: TAF.R.1 Generalizability

TAF.R.1	
Title	**Generalizability**
Actors Involved	Trust Assessment Framework
Type	Functional Requirement
Description	**Background**: Assessing the trustworthiness across the CC is a difficult task due to the complexity and the heterogeneity of the different layers: from far-edge to cloud infrastructures. In the context of TE, there is a multitude of trustworthiness evaluations that need to be conducted by different entities. Also, adhering to the concept of IETF's TPR, network elements need to assess critical behaviours of adjacent nodes before establishing secure links with each other. At the same time, in the context of a managed network, it is also critical to allow the network controller to evaluate the trustworthiness of the underlying topology. This necessitates the evaluation of the behaviour of each network element individually, setting the path for more complex trust characterization at the link, segment, or even path level. All these requirements introduce the need for a generalizable and comprehensive trust modelling paradigm that allows for the expression of dynamic trust relationships that dictate the necessary calculations for the final trust assessment. The envisioned TAF needs to be applicable in the multitude of the different safety-critical scenarios that need to operate under the zero-trust assumption and also capture the changes in the trustworthiness level that might occur over time. This introduces the need for a generic mechanism, expressing target trust propositions of different levels of atomicity. On one hand, CASTOR TAF needs to express and evaluate atomic trust propositions that are intrinsically linked to the types of evidence that can be collected from the target environment. For example, this is the case of a TAF instance evaluating the runtime configuration integrity of a router's software stack, as this allows the mapping of the derived ATL value with the trustworthiness evidence that can be extracted from the Attestation Source offered by TNDE. On the other hand, it is imperative that the TAF is generic enough to be applicable to the multitude of the different requirements on services with mixed criticality. A generalizable TAF should be widely applicable to different UC and would reduce or eliminate the customization effort for each scenario. Ideally, it could even be updated and extended for use in completely novel UC or scenarios. **Description**: CASTOR TAF shall be capable of assessing the trust level for any given scenario; thus, any arbitrary trust model that includes different types of trust relationships, created among heterogeneous trust objects for different properties. Trust relationships can extend from the router level to the link and path levels. These direct trust relationships enable a Local TAF agent, to assess the trustworthiness of a TNDI's behavior using runtime traces monitored in the target environment. Thus, the TNDE ecosystem relies on the Tracing Hub to collect and process traces from the Trace Units, and on the Trust Sources to interpret these traces and provide meaningful, actionable evidence to the Local TAF agent. Beyond these evaluations, CASTOR envisions to model referral trust relationships allowing other TAF instances to infer the trustworthiness of a router element based on previously established trust evaluations. Towards this direction, the TAF should accommodate assessing trust based both on direct trust relationships but also using referral relationships that enable leveraging trust assessments (or opinions) that have already been made by other entities. Through the adoption of the Subjective Logic paradigm, CASTOR TAF shall be able to aggregate the available trust opinions associated with the various trust relationships and reason - under a non negligible level of uncertainty - about the trustworthiness of the required trust objects. In essence, this involves using general methods to measure and calculate the ATL value for any given trust proposition, while simultaneously identifying the RTL constraints that shape the final trust outcome. **Remarks**: (1) Generalizability refers to the architecture and mode of operation of the TAF. Specifically, CASTOR TAF should be able to be adopted even in the context of new scenarios. However, the appropriate trust models need to be defined for such scenarios, and generalizability should not be confused with the need to have defined trust models for all scenarios to be encountered. (2) When elevating the analysis to link- or path-centric trust relationships, an inherent dependency always exists on the trustworthiness of the participating network elements.

Connected to other requirements	TAF.R.4	
KPIs	**Description**	**Value**
	Number of use cases	**=4 heterogeneous use cases** necessitating the trust evaluation of the underlying paths with different trust requirements and constraints

Table 9.18: TAF.R.2 Correctness

TAF.R.2	
Title	**Correctness**
Actors Involved	Trust Assessment Framework
Type	Functional Requirement
Description	**Background**: It becomes apparent that qualitative, informal trustworthiness assessments are insufficient for making informed decisions on whether an entity can be trusted. Instead, measurable, and quantifiable metrics need to be defined. To this end, such metrics need to be defined to enable the Trust Assessment Framework (TAF) to determine whether an entity can indeed be trusted. The decisions rendered by the TAF must align with the actual trustworthiness of the entity in question. For example, if an entity is malicious and, therefore, deliberately provides false trustworthiness evidence to another entity, the TAF agent of the receiving entity should identify it as not trustworthy. Hence, to make the decision on the trustworthiness of an entity, the level of trustworthiness of a particular trustor towards this entity (i.e., trustee) is necessary. The entity for which the level of trustworthiness, namely the Actual Trustworthiness Level (ATL), is determined is specified in a trust proposition. To derive the ATL, trust sources that provide evidence for a trust object are required. However, calculating the ATL of an entity is not sufficient to decide whether it can be trusted. Therefore, in addition to the ATL, there needs to be a Required Trustworthiness Level (RTL) that reflects the level of trustworthiness of an entity required in order to be characterised as trustworthy. By comparing the ATL and RTL, the TAF can decide whether to trust the corresponding entity. **Description**: CASTOR TAF must be able to produce a correct ATL for a target trust proposition. For complex evaluations, trust models need to be employed in order to capture all the trust relationships between the trust objects in order to capture dynamic changes in the ATL calculations. For each of these trust relationships, CASTOR TAF leverages trust sources to quantify a trust opinion for each trust relationship. Based on the individual trust opinions, the final ATL for a proposition is calculated. When there is a change in any of the relevant trust sources - e.g., a violation has been detected by the FSM source capturing an abnormal behaviour in a critical router operation - this shall be reflected accurately in the revised ATL. In addition, the ATL calculation shall express the time evolution of the trust characterization. One the one hand, this may imply the consideration of the freshness of the collected evidence in the trust quantification process. On the other hand, the ATL trust scores should not only rely on the latest evidence available but also take into consideration the previous trust evaluations in order to accurately reflect the trust posture of the target proposition under evaluation. need to be reflected in the ATL scores. In principle, any change in a trust opinion needs to be reflected in the corresponding ATL values of the (direct) atomic trust propositions (e.g., trust proposition related to the runtime integrity of router A stands) but also in all composite trust propositions that depend on that trust opinion (e.g., trust proposition related to the integrity of a path that includes router A as part of a path profile requirement). **Remarks**: (1) It is assumed that these trust sources are not compromised and provide correct evidence. (2) The ATL and the RTL may evolve dynamically. On the one hand, new evidence may lead to a re-evaluation of the current ATL for a trust proposition, whereas on the other hand a revised risk assessment may lead to a revised set of RTL values.
Connected to other requirements	TAF.R.3
	Description **Value**
	The evaluation of the TAF correctness in terms of trust characterization will be realized on a scenario basis. For each of the CASTOR use cases the following scenarios are to be evaluated.

KPIs	**Scenario 1**: Correctness on benign behaviour: Participating trust objects report positive evidence	In this context, it is expected that all atomic trust propositions and any envisioned high-level composite trust propositions are deemed as trustworthy by the participating TAF instances (i.e., Global TAF or Local TAF agents).
	Scenario 2: Correctness on abnormal behaviour: At least one of the participating entities are considered compromised and report negative trustworthiness evidence	In this case, the violation detected any of the CASTOR Trust Sources (i.e., Attestation or FSM source) shall be reflected in the corresponding ATL calculations of the target trust propositions, yielding a final outcome that characterizes the affected entities as untrustworthy. Example 1: An enforced traffic engineering policy requires that all participating routers exhibit configuration runtime integrity guarantees (i.e., critical router software stack configuration is not tampered with). If one of the routers gets compromised, the associated trustworthiness evidence reported to the corresponding Local TAF agent should lead to an untrustworthy trust decision compare, as opposed to the previous scenario. We expect that in ≥ 70 of successful attacks, leading to a compromised network element in the infrastructure layer based on the underlying threat model and the available evidence extracted from the available Trust Sources, the affecting trust propositions should yield ATL values that do not satisfy the RTL constraints.

Table 9.19: TAF.R.3 Robustness and Resilience

TAF.R.3	
Title	**Robustness and Resilience**
Actors Involved	Trust Assessment Framework
Type	Functional Requirement
Description	**Background**: The continuous evaluation of trust within the CASTOR project must operate within the highly dynamic and heterogeneous Compute Continuum (CC). The CC is inherently susceptible to sudden and unpredictable disruptions, such as physical link failures, performance degradation and the possibility of threat actors aiming to disrupt the network. Inevitably, several of these threats may affect the behaviour of the Trust Assessment enablers and core logic. This relates especially to the Local TAF agents deployed as part of the in-router CASTOR capabilities introduced by the TNDE. Given the criticality of the CASTOR TAF as part of the trust-aware traffic engineering policy provisioning, it is imperative to evaluate the resilience and robustness of the primary aspects that enable the trust characterizations across the CC. As per Chapter 6, Local TAF agents are part of the in-router TNDE artifacts that run in-tandem with a TNDI. Even though this allows a Local TAF agent to be protected by adequate security mechanisms—supported by hardware Root of Trust capabilities — there are still potential threat vectors that could impact the trust calculations. Such attacks may lead to inaccurate ATL results either by affecting the evidence collection process or even by tampering with the trust relationships to compromise the ATL trust opinion calculations. **Description**: The TAF must incorporate efficient mechanisms for robustness against possible attacks, as well as resilience against potential operational failures. Such attacks may affect a TAF instance to report inaccurate trustworthiness claims or even report no claims at all. When considering the CC-wide CASTOR Trust Assessment Framework, it is critical to take into consideration possible attacks on the federation between the different Local TAF agents and the Global TAF centrally located at the Orchestration Layer. Overall, the CASTOR TAF must ensure continuous and reliable trust evaluation, as well as allow recovery and rapid adaptability to manage dynamic trust environments thus mitigating the impact of compromised participating entities. **Remarks**: Whilst a major factor of this requirement is concerned with remediation in the event that a threat actor has compromised the network, it is equally important to consider non-malicious events such as those caused by hardware and/or software failure, connectivity issues and performance degradation.

Connected to other requirements	TAF.R.4	
KPIs	**Description**	**Value**
	Ability of the TAF to withstand attacks targeting the functional operation of the host environment	To ensure that the Local TAF agent operates securely within its host environment, it must provide robust protection mechanisms that do not compromise the router's operational performance. Hence, it shall exhibit an acceptable performance overhead on the overall TNDI performance, ensuring no measurable degradation in the routing operational profile.

Table 9.20: TAF.R.4 Flexibility of Trust Sources

TAF.R.4	
Title	**Flexibility of Trust Sources**
Actors Involved	Trust Assessment Framework
Type	Functional Requirement
Description	**Background**: The Zero-Trust principle is one of the fundamental concepts mentioned in the first TAF functional requirement (TAF.R.1). This implies that no trust is assumed in the infrastructure layer and all trust objects are considered to be possibly untrustworthy at the beginning. Thus, no initial trust between the entities shall be assumed, but the trust between the entities shall be continuously evaluated based on evidence. The multitude of trust propositions involves diverse trust properties (e.g., integrity, resilience, confidentiality) and a wide range of trust objects, enabling trust characterization from the behaviour of a specific path segment up to the evaluation of an entire path. The selection of the target trust propositions to be evaluated is intrinsically linked to the threat model that is taken into consideration. Consequently, this complex and diverse landscape introduces the need for a wide variety of trustworthiness evidence that need to be extracted from the target environment so as to quantify the respective trust opinions and derive the final ATL values. This introduces the need for the overarching CASTOR TAF to support multiple Trust Sources with different inherent characteristics that may affect the trust engineering process (e.g., evidence quantification, deterministic or probabilistic evidence, observation uncertainty, evidence extraction mechanisms). **Description**: CASTOR encompasses a Trust Assessment Framework where multiple Trust Sources share the collected evidence to be considered in the ATL derivation for the target trust propositions. Through an abstract - yet extendable - Trust Source Manager, different Trust Sources may be connected to a TAF instance in depending on the use case and evaluations of interest. Since each Trust Source can rely on different raw traces from the target TNDI environment, it is essential to have a robust and flexible Tracing Hub that can gather and process traces from various Trace Units (SR.11). Adhering to the Zero-Trust paradigm, all types of trustworthiness evidence offered by any Trust Source needs to be collected and reported in a secure and verifiable manner. This allows the TAF's Trust Source Manager to verify the provided evidence and accurately quantify its corresponding trust opinion. Information on the existence and enforcement of security mechanisms constitutes critical trustworthiness evidence that could be reported by multiple Trust Sources. In this context, several hardware-based Trust Sources could be a TPM or a TEE. In parallel, software-related Trust Sources may relate to firewall applications deployed as part of a router's software stack (i.e., as part of the TNDI) or even intrusion detection systems. Such Trust Sources may offer evidence acting as a proof of ownership of a security property (e.g., TNDI has securely booted) or an indication of compromise associated with a specific (probabilistic) confidence level. **Remarks**: Both the core TAF logic and its associated Trust Sources comprise a TAF instance which constitutes a trusted component within the TNDE. This implies that non-authenticated and unauthorized Trust Sources cannot interact nor affect the trust calculations.

Connected to other requirements	SR.4, SR.11, SR.13, TAF.R.1	
KPIs	**Description**	**Value**
	Trust Sources integrated with the TAF framework	$>= 3$ different trust sources are evaluated in the context of the use cases.

Table 9.21: TAF.R.5 Performance of trust evaluations

TAF.R.5		
Title	**Performance of trust evaluations**	
Actors Involved	Trust Assessment Framework	
Type	Non-Functional Requirement	
Description	**Background**: The dynamic nature of the Compute Continuum (CC) necessitates that the Trust Assessment Framework (TAF) operates under strict time requirements to support ongoing network operations that demand low latency, such as high-priority V2X communications to allow trustworthy communication among first responder units as well as the real-time sharing of data across vast and complex networks of Unmanned Aerial Vehicles (UAVs). The nature of the TAF demands continuous and dynamic trust assessment which inherently introduces challenges in relation to computational overhead, energy consumption and network bandwidth due to the need for continuous monitoring and evidence processing. It is therefore vital for the TAF to perform efficient and optimised trust assessments in real-time, ensuring that their integration introduces no perceptible delay and/or overhead to the overall network's communications, nor does it affect the operational performance of all participating entities in the network topology. **Description**: The TAF should be able to assess the trustworthiness of an entity within strict time requirements. In particular, the TAF should be able to calculate an Actual Trustworthiness Level (ATL), and compare it against the Required Trustworthiness Level (RTL), under strict time constraints with a maximum runtime delay of $100ms$ in both standalone and federated contexts. This is achieved by each component of the TAF being responsible and accountable for performing optimised calculations in real-time, introducing no additional overhead when integrated together into the TAF. Various techniques should be incorporated into the TAF to facilitate such performance requirements. For example, Subjective Logic (SL) implemented to form trust opinions should be done so with efficiency in mind, optimising core processes such as fusion and discounting that will need to be frequently conducted in all federated contexts and most standalone contexts. Furthermore, the ability of the TAF to conduct trust assessments in a federated context lessens computational overhead on individual Local TAFs, lessening their potential as bottlenecks. Local TAFs typically run on highly resource-constrained devices and minimising complex trust calculations at this level will drastically improve operational performance. As an example, Local TAFs should exclusively focus on the evaluation of trust in relation to integrity, whereas trust evaluations in relation to other trust properties (although still including integrity) should be handled by the Global TAF. **Remarks**: It should be noted that with timing constraints comes a potential trade-off with the accuracy of trustworthiness calculations. For example, a strict time requirement (i.e. requiring TAF instantiation and execution within $100ms$) may necessitate the evaluation of less trust sources, or the processing of trust decisions at the local level rather than at the Global TAF. Optimising this trade-off is dependent upon the given context, as whereas one scenario may value faster performance over accuracy (such as real-time communication in a low-risk environment), other scenarios (that are safety-critical) may instead prioritise accuracy.	
Connected to other requirements	TAF.R.1	
	Description	**Value**
	Standalone TAF runtime performance	$\leq 100ms$ delay when the TAF is instantiated and executed as part of the application software stack in the target system (i.e., Local TAF agent in TNDI, or Global TAF on the controller plane collecting trustworthiness evidence from TNDE)

KPIs	Federated TAF runtime performance	$\leq 100ms$ delay when the Global TAF is instantiated and interacts with the underlying Local TAF agents
	NOTE: This does not include the time needed for the collection, processing and communication from the entities (Global TAF - Local TAF agents - Trust Sources), while we are further excluding any network latency caused by trust sources. The focus is only on the timing requirements of the TAF operation and calculation of the ATL and evaluation against the specified RTL	

Table 9.22: TAF.R.6 Scalability

TAF.R.6		
Title	**Scalability**	
Actors Involved	Trust Assessment Framework	
Type	Non-Functional Requirement	
Description	**Background**: When it comes to in-router trust assessment, the evaluations characterise the router behaviour of the corresponding TNDI. Adhering to the IETF's trusted path routing concepts, such trust evaluations may also involve the trust posture of the neighbouring TNDIs. Hence, although runtime performance remains a critical factor, there is no explicit requirement concerning the scalability of the trust models. The trust relationships to be managed are inherently dynamic; however, their number remains relatively limited. However, this is not the case for the Global TAF running in the Orchestration Layer. In this context, the trust models need to be able to cope with different sets of trust requirements coming from multiple path profiles. This introduces different trust model instances that need to compose different high-level trust propositions and compare them against different set of RTL constraints. At the same time, the dynamic nature of routing path selection - as part of the traffic engineering process (e.g., automated flow steering within a segment routing TE policy) - necessitates continuous updates to the underlying trust models in order to accurately represent the trust state at the path level. Furthermore, the dynamic reconfiguration of the infrastructure topology, resulting from the enrolment and detachment of router elements, introduces a highly dynamic and volatile environment that must be effectively accommodated by the CASTOR Trust Assessment Framework (TAF). **Description**: The CASTOR TAF shall be scalable in order to assess the trustworthiness levels of all involved TNDI nodes within strict time requirements even if the number of nodes increases dynamically. This also includes calculations at the link, segment, or path level. The TAF should also be able to assess trustworthiness of every new TNDI node as part of the secure onboarding process. This would imply that the TAF instances need to be able to quickly update, analyse, and break down large trust models representing. Irrespective of the number of nodes in the model, the TAF needs to be able to calculate the necessary Actual Trustworthiness Levels (ATLs) in a short period of time. To avoid overloading a Local TAF agent with overwhelming trust computations that could hinder the operational performance of a TNDI, CASTOR leverages the federation of the overarching TAF, allowing the offloading of trust calculations to the Global TAF.	
Connected to other requirements	TAF.R.5	
	Description	**Value**
KPIs	Trust objects/propositions supported in intra-domain	Local vs. Global TAF
	Trust models/sessions	Local vs. Global TAF

9.2.2 Router Operational Assurance

9.2.2.1 TNDE

Table 9.23: TNDE.R.1 Management of Network Device TNDIs

TNDE.R.1		
Title	**Management of Network Device TNDIs**	
Actors Involved	TNDE, TNDI, CASTOR orchestrator	
Type	Functional Requirement	
Description	**Background**: CASTOR provisions trust-aware paths that take the trust level of each node in the network topology into account. By continuously re-assessing the trust levels of each node, CASTOR can dynamically update the paths to satisfy requested security and performance requirements for the routing. The TNDE is CASTOR's core component in the device-side TCB and responsible for exposing the trustworthiness evidence and enforcing the trust-aware paths. In order to support a variety of physical and virtual networking nodes, CASTOR introduces the concept of TNDIs—Trust Network Device Interfaces (see chapter 6)—representing a functional unit capable of taking part in CASTOR's trusted path routing (e.g., a physical router, a vRouter, a routing compartment/partition within a router). CASTOR's device-side TCB needs to be able to manage the TNDIs of a device as a pre-requisite for onboarding each of them into the CASTOR network and enabling trust assessment and path enforcement. **Description**: CASTOR's TNDE needs to be able to manage the TNDI(s) of the underlying network element (e.g., network device). Each of these TNDIs needs to provide the necessary capabilities to take part in CASTOR's trusted path routing. As detailed in SR.6, SR.11, and the related ones, the TNDE needs to allow the CASTOR upper-layer services (especially the orchestrator-related services) to interface with the TNDIs, e.g., to onboard them and enable the trust assessment mechanism for them. The TNDE needs to allow interfacing with each TNDI independently and shall manage their associated security configurations and trust policies (e.g., runtime tracing—SR.11) separately. The TNDE shall be able to associate TNDIs with their assigned CASTOR domain (see SR.7 on TNDI-SP control channel) such that the TNDE can enforce that a TNDI is only actively onboarded to a single CASTOR domain at a time.	
Connected to other requirements	SR.2, SR.6, SR.7, SR.11	
KPIs	**Description**	**Value**
	Topology evaluation	Large scale: ≤ 50 **TNDI nodes**, leveraging vRouter elements. Each host environment spanning between one and 10 TNDIs Small scale: 1 physical router (TPM-enabled Cisco Router, if available)

Table 9.24: TNDE.R.2 Dynamic Setup and Configuration of the TNDE and TNDIs (Re-/Programmability)

TNDE.R.2	
Title	**Dynamic Setup and Configuration of the TNDE and TNDIs (Re-/Programmability)**
Actors Involved	TNDE, TNDI, CASTOR orchestrator
Type	Functional Requirement

Description	**Background**: The CASTOR orchestrator needs to provision trust-aware paths throughout its network topology to enforce trusted path routing. However, this requires the CASTOR orchestrator to be able to configure the network devices, such that they can be onboarded into the CASTOR network and the trustworthiness data of them can be collected. Otherwise, the CASTOR orchestrator cannot calculate the trust-aware paths and enforce them in the network. Furthermore, the CASTOR orchestrator needs to be able to dynamically reconfigure the network nodes to update the paths or enforce a new trust policy if trust levels or path requirements change (i.e., dynamic re-programmability). **Description**: CASTOR's orchestrator needs to be able to remotely configure the TNDE and its TNDIs via control channel messages. Based on that, the CASTOR orchestrator can add (join) a network device to the CASTOR domain and onboard its TNDIs, configuring the TNDE components and TNDIs as required for the trusted path routing. The onboarding allows to set up the tracing and trust assessment for the TNDIs, as well as communication channels with the CASTOR upper layer services (see SR.6 for details). Furthermore, the TNDE shall enable the CASTOR orchestrator to dynamically reconfigure the TNDI security configurations and trust policies to enable adjustments of the trust assessment and the enforcement of different trust-aware routing paths (re-programmability). **Remarks**: (1) As detailed in SR.6 and SR.7, a TNDI-SP control channel can serve as the secure transport for the control messages between the CASTOR orchestrator and the device-side TNDE. (2) The TN-DSM of CASTOR's device-side TNDE can serve as the TNDI-SP control channel endpoint, allowing the CASTOR orchestrator component to configure the TNDE subcomponents and the TNDIs.
Connected to other requirements	TNDE.R.1, SR.6, SR.7, OSS.R.1
KPIs	**Description**: Runtime performance on enforcing of a new policy **Value**: ≤ 1.3 sec, excluding network latency and assuming that the necessary TNDI-SP control channel is in place. In this scenario we envision to measure the time required for a TN-DSM to fetch an updated Trust Policy and enforce it over an existing one. The new Trust Policy may configure the Local TAF operations either directly (e.g., update to the RTL constraints) or indirectly (update to the frequency of trustworthiness evidence collection or reconfiguration of a Trace Unit).

9.2.2.2 Evidence-based Monitoring

Table 9.25: FSM.R.1 Model optimisation and specialisation

FSM.R.1	
Title	**Model optimisation and specialisation**
Actors Involved	Finite State Machine (FSM), TNDE
Type	Functional Requirement

Description	**Background**: Considering the operational functionalities and constraints of the routers under analysis, a flexible, efficient and specifically trained monitoring model is required to achieve a more resilient and secure monitoring solution. This process will take in consideration not only specific runtime information collected by the tracing layer, but also the threat model associated to each router, which could differ depending by the security experts or by the specific needs of each customer. **Description**: The FSM models will be trained and optimised ad-hoc on each router's requirement to have the minimum number of internal states required to perform successful router's behavioural analysis, enabled by the types of traces collected at a lower level. To ensure maximum efficiency and least impact on the system under analysis, the FSM models will be trained and optimised offline based on real runtime traces captured by the tracing layer, and later securely deployed and configured on each device once fully trained. **Remarks**: By leveraging all these information, we aim to train a tailored model which will be more effective than generic monitoring models, while being optimised to not violate the router's operational requirements. Each model generation process will need to find a good balance between efficiency and accuracy, while also avoiding over-fitting models to specific cases. An accurate model might require a bigger set of traced information that will impact its runtime evaluation and, therefore, its efficiency and the overall overhead put on the system, but at the same time there is a need to have an accurate model to provide meaningful attestation evidence.	
Connected to other requirements	SR.13	
KPIs	**Description**	**Value**
	Model optimisations will also depend on specific threat models	Yes/No

Table 9.26: FSM.R.2 Model explainability

	FSM.R.2	
Title	**Model explainability**	
Actors Involved	Finite State Machine (FSM), TNDE	
Type	Functional Requirement	
Description	**Background**: Explainability of the alerts generated by monitoring models is essential because it allows security experts, analysts and local trust agents to gather extra information about the context and the reasoning behind the raised alert. This transparency not only helps in better understanding the current status of the device, and potentially its surrounding infrastructure, but also leads to lower time of alerts assessment and more prompt reaction times. **Description**: The FSM models will be able to identify which traced actions are responsible for behavioural anomalies. **Remarks**: This extra information related to the detected anomalies will be provided by the FSM models in an easily interpretable format, enabling immediate- and post-assessment analysis performed by security experts and local trust agents. When necessary, the alerts will be generated based on the runtime evaluation of the FSM model based on the runtime traces collected by the tracing layer. During the model evaluation, contextual data about the potential current anomaly are also been gathered and used to augment the alert itself to be provide extra evidence about it.	
Connected to other requirements	SR.13	
KPIs	**Description**	**Value**
	Time taken to gather contextual information	< 250 milliseconds

9.2.3 Trust-aware Service Assurance

9.2.3.1 Orchestrator

Table 9.27: OSS.R.1 Secure Remote Asset Management and Reconfiguration Effectiveness

OSS.R.1		
Title	**Secure Remote Asset Management and Reconfiguration Effectiveness**	
Actors Involved	Service Orchestrator, TNDE, TNDI	
Type	Functional Requirement	
Description	**Background**: In distributed, trust-aware network environments such as CASTOR's, remote management and reconfiguration of assets (e.g., routers, virtual network functions, edge nodes) are essential for dynamic orchestration and lifecycle control. However, these operations introduce significant attack surfaces, particularly during remote configuration updates, firmware provisioning, and policy re-enforcement. To maintain security and trustworthiness throughout the asset lifecycle, remote asset management operations must be both secure and effective. This means that all configuration, monitoring, and re-configuration activities must occur over authenticated and integrity-protected communicaton channels, ensuring that no malicious entity can manipulate device states or device configurations. CASTOR envisions this capability as an integral part of its trusted orchestration framework, enabling safe and verifiable reconfiguration of network elements in response to evolving trust and/or performance conditions. **Description**: All configuration and reconfiguration commands must be authenticated, integrity-verified, and logged to ensure accountability and traceability. These operations shall be conducted only on assets securely on-boarded, showing proofs that they are adequately equipped with the proper CASTOR TCB capabilities. Key security and functional objectives include: (i) Dynamic updates to the trust and traffic engineering policies enforced at the TNDI level (SR.7), including example mechanisms such as Segment Routing Traffic Engineering policies enforced through a PCE element. (ii) Secure and verifiable collection of trustworthiness evidence from managed assets towards the orchestration layer, enabling CASTOR TAF to provide an accurate estimation of domain trustworthiness as perceived by the Service Orchestrator. (iii) Enhanced telemetry for near-real-time reporting of trust-related metrics to the orchestrator's event monitoring tools. (iv) Verification and rollback capabilities ensuring recovery from failed or unauthorized configuration attempts. **Remarks**: (1) The implementation should be agnostic to the underlying management interface employed, acknowledging that no single technology is uniformly supported across all vendors (e.g., the Cisco IOS XRv 9000 router does not support RESTCONF interfacing). The approach will therefore focus on presenting well-defined network controller APIs compatible with well-established orchestration tools, ensuring interoperability across heterogeneous environments. (2) Reconfiguration actions should be latency-aware and designed not to disrupt ongoing services or established trusted paths.	
Connected to other requirements	SR.7, OSS.R.2	
KPIs	**Description**	**Value**
	Container lifecycle management (LCM) Operations [controlled and uncontrolled element configuration]. Instantiation and termination time of virtualized and hardware network elements.	$\leq 1sec$ for the instantiation of network element ≤ 10 sec for the reconfiguration of network element ≤ 20 sec for restarting network element ≤ 30 sec for the reconfiguration of network element which requires restart of the element

Table 9.28: OSS.R.2 Trust- and Policy-driven Orchestration and Service Placement

OSS.R.2		
Title	**Trust- and Policy-driven Orchestration and Service Placement**	
Actors Involved	Service Orchestrator, Facility Layer	
Type	Functional Requirement	
Description	**Background**: In distributed, multi-domain network environments such as those envisioned in CASTOR, the orchestration of services must optimize not only performance, scalability, and resource utilization, but also the trustworthiness of the underlying infrastructure. The compute continuum, from far-edge to cloud, introduces high heterogeneity in device capabilities, ownership domains, and security postures. To ensure trustworthy end-to-end service delivery, CASTOR augments existing Traffic Engineering (TE) approaches with trust metrics. Traditional TE objectives (e.g., latency, bandwidth, load balancing) are extended with trust-related indicators derived from the TAF. At the same time, orchestration decisions are guided by formally defined policies and intents that encode business, security, and trust requirements, enabling CASTOR's policy-driven orchestration model. This approach ensures that service placement, routing, and lifecycle operations remain trust- and policy-compliant, dynamically adapting to changing conditions while maintaining alignment with pre-defined SSLAs. **Description**: The service orchestrator shall expose and consume a policy-driven orchestration interface that translates high-level intents and SSLA-derived trust/security policies into actionable orchestration directives. These directives drive trust- and network-aware service placement, ensuring that orchestration decisions are simultaneously policy-compliant and trust-optimized. Key functional capabilities include: (i) The orchestrator shall deploy services only on nodes that meet the required trust thresholds as assessed by the TAF (actual trust level $\geq$ required trust level). (ii) The orchestrator shall synchronize with the Optimization Engine to obtain trust- and performance-aware placement recommendations, combining network state, TE metrics, and trust evidence. (iii) The resulting placement and routing policies shall be enforced through the PCE for automated configuration, ensuring consistency between orchestration intent and network-level realization (see TE.R.1). (iv) The orchestrator shall support continuous monitoring and automatic adaptation—re-placement or path re-optimization, when trust degradation or SSLA non-compliance is detected. (v) The orchestration interface shall expose endpoints to upper-layer management or exposure functions to, submit or modify intents/policies, query trust/risk posture and SSLA compliance, subscribe to policy violation or trust-score change events, and retrieve service lifecycle and compliance status. Through these mechanisms, CASTOR enables end-to-end intent translation and enforcement, ensuring that trust, security, and performance considerations are jointly reflected in orchestration decisions across intra- and inter-domain environments. **Remarks**:(1) Continuous synchronization between the service orchestrator, the TAF, and the Optimization Engine is required to ensure that service placements and TE policies always reflect the latest trust and network state. (2) The interface supports both automated policy enforcement (via PCE) and manual configuration (via Facility Layer APIs or CNI layer), depending on deployment capabilities. (3) Cross-domain service provisioning is supported under the assumption of pre-established SSLAs between cooperating domain operators, guaranteeing mutual trust and policy compatibility. (4) Together, the trust- and policy-driven orchestration functions form a key enabler for CASTOR's intent-based automation vision, linking trust assurance, security compliance, and operational efficiency.	
Connected to other requirements	OSS.R.3, TE.R.1	
KPIs	**Description**	**Value**
	Synchronization latency to enforce latest TE policy recommendations by the Optimization Engine.	Near real time < 5 sec for automated policy configuration via PCE. No fixed threshold (measured value depends on API communication latency) for manual policy configuration via Facility Layer, CNI layer etc.

Table 9.29: OSS.R.3 Accurate and fresh synchronization of the network topology attributes and trust assurance reports

OSS.R.3		
Title	**Accurate and fresh synchronization of the network topology attributes and trust assurance reports**	
Actors Involved	Service Orchestrator, Telemetry API	
Type	Functional Requirement	
Description	**Background**:In CASTOR's distributed and trust-aware orchestration framework, the accuracy and freshness of network topology information and trust levels are crucial for reliable service placement, path computation, and enforcement. The orchestration layer relies on continuous updates about topology state, including link metrics, node status, routing capabilities, and trust levels, to make informed decisions. In traditional MANO or SDN environments, topology synchronization mechanisms often operate in a best-effort manner and may not include trust- or risk-related metadata. However, CASTOR's architecture requires real-time, trust-enriched topology synchronization that integrates both operational (e.g., link latency, throughput) and security/trust parameters (e.g., trust scores, attestation state, data confidentiality). Furthermore, CASTOR's orchestrator must not only deploy and manage services but also continuously verify that active services remain compliant with their SSLA-defined trust requirements, triggering appropriate countermeasures when violations occur. **Description**: The service orchestrator shall maintain an accurate and continuously updated view of the network topology by synchronizing with all relevant components across intra- and inter-domain boundaries and ensure continuous trust assurance for all deployed services by verifying compliance with SSLA policies and dynamic trust conditions. These operations must include both traditional network attributes (e.g., utilization, routing state) and CASTOR-specific trust parameters provided by the TAF and the risk assessment engine. This requirement enables the orchestrator and the optimization engine to make consistent, context-aware, and trust-compliant orchestration decisions. The synchronization process shall: (a) Support near real-time updates of topology and trust information; (b) Detect and propagate topology or trust changes (e.g., node isolation, trust degradation); (c) Ensure that all orchestration and policy decisions are based on the latest verified network state. Regarding trust degradation, attestation failure, or SSLA non-compliance detection, the orchestrator shall: (a) Initiate automated actions, such as service re-placement, isolation, or re-orchestration. (b) Notify relevant CASTOR components (e.g., Policy Manager, Optimization Engine) to adapt service configurations or paths accordingly. By these functions, CASTOR guarantees trust-consistent orchestration, where both functional and trust dimensions of the network are continuously aligned. **Remarks**: (1) The freshness of topology data directly affects the reliability of trust-aware orchestration, risk assessment, and optimization decisions. (2) Integration with telemetry and tracing components ensures evidence-backed validation of topology states. (3) The orchestrator's trust assurance mechanisms must integrate both proactive (predictive trust monitoring) and reactive (event-triggered remediation) approaches. (4) All of the trust-related information (e.g., Local TAF agent trustworthiness claims or trustworthiness evidence from in-router Trust Sources) are available at the Global TAF residing in the Orchestration Layer. This communication is realized through appropriate TNDI-SP channels that are provisioned as part of the CASTOR framework. This allows the Global TAF to properly discount the evaluations - that are provided by each router - appropriately and derive the trust capabilities of the entire topology enabling the trust-aware service provisioning. Using the discounting operator from subjective logic, the Global TAF refines its opinion by taking into account not only the evidence provided by a router element but also the degree of trust placed in that router. On a parallel note, as part of the enhancement of the real-time insights that are collected at the orchestration layer, CASTOR envisions to extend well-established telemetry mechanisms (e.g., Prometheus interfaces) in order to share the local trustworthiness claims that are computed by the Local TAF agents. This aims to provide enriched insights on the trustworthiness levels as perceived by the Local TAF agents. (5) Evidence of trust compliance and service behavior should be securely logged within the DLT infrastructure for transparency and auditability.	
Connected to other requirements	OSS.R.1	
KPIs	**Description**	**Value**
	Secure telemetry freshness interval	In the order of seconds

9.2.3.2 Optimization

Table 9.30: OPT.R.1 Trusted Path Optimization

OPT.R.1		
Title	**Trusted Path Optimization**	
Actors Involved	Optimization Engine	
Type	Functional Requirement	
Description	**Background**: This requirement defines the ability of CASTOR's Optimization Engine to compute trusted, efficient trusted end-to-end paths across network segments that combine trust-level constraints and network-level performance metrics. The requirement operationalizes the outcome Trust Assesment Framework (TAF) and Service Security Level Agreement (SSLA) enforcement into the actual path computation. The goal is to ultimately guarantee that every path used within the CASTOR frameworks respect the Required Trust Level (RTL) while maximizing resource efficiency. **Description**: The Optimization Engine shall compute near-optimial end-to-end (ingress to egress) trust paths for network services. It will jointly consider (a) node/link trust scores (ATLs), (b) user SSLA constraints, (c) network metrics. It then outputs the ranked segment routing path candidates annotated with compliance and cost indicators. It must provide inherent support for multi-objective optimization balance trust with performance and other extra-functional metrics. **Remarks**: (1) Supports inter/intra-domain usage. (2) Configurable weighting between trust and performance. (3) Supports explicit SR path encoding. (4) Supports identification of the optimal set of rules to enforce a dynamic trust-aware TE policy. **Note**: Optimality gap is defined as the difference in minimal value of the objective function for an optimisation problem, versus the value of objective function returned by heuristic search in the Optimization Engine.	
Connected to other requirements	TAF.R.1, OSS.R.3	
KPIs	**Description**	**Value**
	Optimality Gap (vs. Exhaustive Oracle) on small examples.	$\leq 10\%$ (median)

Table 9.31: OPT.R.2 Network and attestation information

OPT.R.2	
Title	**Network and Trust Objective Variables**
Actors Involved	Optimization Engine
Type	Functional Requirement
Description	**Background**: This requirement ensures that the optimization engine can work with live trust and network telemetry. It bridges the gap between the trust layer and optimization process. It ensures that routing decision remains consistent with current trust and network states. **Description**: The Optimization Engine shall consume network performance metrics and attestation-based trust data from TNDE/TNDI. It will integrate theses values into the path cost function. It will penalize the segments with low or stale trust values, as described in explicit path identification formulation . It will also allow configurable attenuation of trust decay over time. The Optimization Engine will overcome challenge of integrating these diverse values in a cost function using weighted average of the cost functions into a single overarching one or potential alternatives. **Remarks**: (1) Ensures continuous synchronization with TNDE/TNDI and TAF. (2) Supports freshness thresholds. (3) Supports configurable penalty functions.

Connected to other requirements	OPT.R.1, SR.9, TAF.R.4	
KPIs	**Description**	**Value**
	Selected path compliance with ATL	100%

Table 9.32: OPT.R.3 Multi-path computation

OPT.R.3		
Title	**Multi-path computation**	
Actors Involved	Optimization Engine	
Type	Functional Requirement	
Description	**Background**: CASTOR requires not just a single ”optimal” path but a portfolio of diverse, compliant paths to provide high availability and resilience (reliability). This requirement ensures redundancy and potential load balancing but requiring Optimization Engine to compute multiple viable paths per service. This requirement ensures rapid recovery and continuous SSLA compliance even under failures or trust degradation. **Description**: The Optimization Engine shall compute multiple alternative paths per service. It will support K-best disjoint path algorithms while providing performance and trust compliance metrics for reach. **Remarks**: (1) Enables resilience (2) Enables load balancing (3) Enables fast failover	
Connected to other requirements	OPT.R.1 , TE.R.4, OSS.R.3	
KPIs	**Description**	**Value**
	Alternative Paths per Request	$>= 2$
KPIs	Average Optimization Engine Service Time	$<=$ Average Optimization Engine Invocation Arrival Time

Table 9.33: OPT.R.4 Re-optimization

OPT.R.4	
Title	**Re-optimization**
Actors Involved	Optimization Engine
Type	Functional Requirement
Description	**Background**: CASTOR is designed for dynamic networks that evolve with trust levels, topologies, or SLAs in a constant state of Flux. This requirement mandates that the Optimization Engine adapts to the changing network. Thereby, optimization engine must be capable of re-evaluating existing paths and recomputing alternatives when circumstances shift. These re-evaluations prevent static trust violations and ensure service continuity when the nodes lost trust or new evidence become available. **Description**: The Optimization Engine shall support re-optimization. This reoptimization could be triggered by changes in trust attributes, network parameters, or SLA requirements (or violations). It shall enable checkpointing to allow search rollbacks for faster response time. Furthermore, it will provide incremental updates for efficiency. It will provide tunable persistence depth to provide quality vs computation trade-off, choosing wisely between re-calculating from scratch or starting from a snapshot/checkpoint based on the complexity. **Remarks**: (1) Improves reliability. (2) Improves Efficiency. (3) Supports partial re-computation use. (4) Superioir solution than state-of-the-art IGP shortest path solution.

Connected to other requirements	OPT.R.1	
KPIs	**Description**	**Value**
	Re-optimization Success Rate	$\geq 80\%$

Table 9.34: OPT.R.5 Optimization Scalability

OPT.R.5		
Title	**Optimization Scalability**	
Actors Involved	Optimization Engine	
Type	Non-Functional Requirement	
Description	**Background**: CASTOR framework is designed to be applied to large networks. Therefore, to be practical, the Optimization Engine must also scale to large network topologies and workloads. This requirement defines the performance expectations and computation design goals that ensure the optimization process remains responsive even network sizes grow by an order of magnitude. It provides a trade-off between accuracy and runtime by allowing heuristic and distributed execution models. **Description**: The optimization engine is able to solve the problem of trust path optimization problem in practical time frame for real-world representative size instances. **Remarks**: (1) Supports CPU/GPU acceleration and distributed execution. (2) exposes performance scaling metrics to operators.	
Connected to other requirements	OPT.R.1	
KPIs	**Description**	**Value**
	Problem Scaling Factor	$\leq 2^x$ (subexponential)

9.2.3.3 Risk Assessment

Table 9.35: RA.R.1 RTL Derivation and Management

RA.R.1		
Title	**RTL Derivation and Management**	
Actors Involved	Risk Assessment	
Type	Functional Requirement	
Description	**Background**: CASTOR operates under Zero-Trust principles where no implicit trust is granted to any infrastructure element. The Risk Assessment Engine must quantify security requirements by deriving RTL values that define the minimum trust thresholds infrastructure elements must achieve. RTL values must capture varying trust requirements across three critical dimensions: first, establishing minimum trust requirements for router onboarding to ensure only adequately secure infrastructure elements can join the topology; second, expressing the diverse trust requirements of different services in the service catalogue as specified in their respective SSLAs; and third, enabling the Global TAF to determine the trust guarantees each node, link, and path can provide, thereby allowing the Optimization Engine to match infrastructure capabilities with service requirements when computing paths for specific path profiles.	

Description	Additionally, RTL derivation must account for router-specific factors that influence individual trust requirements. Different router types from various vendors have distinct hardware security capabilities that establish baseline trust levels, requiring vendor-specific RTL thresholds. Furthermore, topological placement affects risk exposure, as routers in critical network positions require elevated RTL values due to increased cascading attack risks. Finally, security control configurations vary between individual routers, where identical hardware may have different RTL requirements based on the specific security mechanisms actively enforced on each device. These RTL values serve as fundamental constraints for trusted path selection, ensuring that only paths that meet the required security posture are utilized for service provisioning. The derivation of RTLs must be based on comprehensive threat and vulnerability analysis across the heterogeneous Compute Continuum, accounting for the dynamic nature of emerging threats and evolving attack landscapes. **Description**: The Risk Assessment Engine shall analyze threats and vulnerabilities across all layers of the Compute Continuum to generate RTL values tailored to different service criticality levels as expressed in SSLAs. The engine shall perform comprehensive threat modeling and risk assessment (e.g., leveraging methodologies such as TARA), categorizing vulnerabilities from device-level (memory-related, firmware exploits) to cross-layer threats across infrastructure software/hardware stacks and network-level threats (routing manipulation, traffic analysis). Based on this analysis, the engine shall construct risk graphs identifying vulnerability paths and dependencies, then map identified attack types to necessary security controls, determining the minimum RTL required for determining the minimum RTL required to satisfy each path profile in the service catalogue. The engine shall obtain topology status via the Topology Graph in the Facility Layer, with asset topology information provided by the Service Orchestrator. Additionally, failed attestation evidence and raw traces may be shared to the CASTOR DLT for auditability purposes, enabling Security Administrators to post-analyze and identify vulnerabilities or update risk assessments based on these risk indicators, which can subsequently be reflected in the Risk Assessment Engine to update risk graphs and affected RTL values. These RTL values serve as thresholds for the Trust Assessment Framework, enabling the Optimization Engine to select paths where node/link ATLs exceed the required RTL, ensuring SSLA compliance. **Remarks**: (1) Methodology & Profiles: It uses systematic threat and risk analysis to construct Trustworthiness Profiles and determine the minimum RTL. (2) Dynamic Adjustment: It dynamically adjusts the RTL based on topology updates from the Facility Layer and post-analysis of security incidents to detect emerging threats. (3) Role: It provides the Required Trust Level thresholds to the TAF. The TAF identifies nodes and links where ATL exceeds RTL, and the Optimization Engine uses these trusted elements for path selection. (4) Routing Plane Validation: The Risk Analysis within a CASTOR-enabled domain will be validated at the routing plane.
Connected to other requirements	SR.4, TAF.R.1, TAF.R.2
KPIs	**Description** / **Value**
	RTL Calculation Methodology exhibits expected behaviour degradation based on the defined threat model — TRUE - Evaluation scenario: Initially run a baseline risk assessment to establish a specific set of RTL values for network infrastructure. Subsequently introduce new and more severe vulnerabilities into the Risk Assessment Engine and verify that RTL values increase proportionally to reflect the heightened risk exposure. The methodology should demonstrate measurable RTL adjustments that correlate with vulnerability severity and network topology impact
	Time to update RTL values upon detection of new threats — ≤ 1 minutes, including the updated risk analysis, the construction and enforcement of the Trust Policy through the CASTOR DLT.

Table 9.36: RA.R.2 Cascading Attack Detection

RA.R.2	
Title	**Cascading Attack Detection**
Actors Involved	Risk Assessment
Type	Functional Requirement

Description	**Background**: In the heterogeneous Compute Continuum, attacks often cascade across multiple components through lateral propagation between nodes or vertical escalation across software/hardware layers within nodes. These cascading attacks create compound risks that exceed individual vulnerabilities, as each compromised element enables further exploitation. Traditional risk assessment approaches analyze threats in isolation, failing to capture propagation dynamics and dependency chains. In CASTOR's zero-trust environment, detecting and preventing cascading attack patterns is essential for trusted path routing, as a single compromised path element could compromise the entire end-to-end security assurance. Furthermore, cascading attack analysis enables more accurate RTL calculations by accounting for topology-aware risks that network placement introduces to router assets, ensuring that path profile requirements reflect the true risk exposure of infrastructure elements within their specific network context. **Description**: The Risk Assessment Engine shall identify and analyze cascading attack scenarios by constructing dependency graphs that map both node-to-node relationships across the network topology and cross-layer dependencies within individual infrastructure elements. The engine shall analyze how the exploitation of vulnerabilities at one node can provide attack vectors to compromise neighboring nodes along routing paths, as well as how vulnerabilities at one layer (e.g., device firmware, hypervisor, network OS) can enable attacks on other layers within the same node. By modeling these multi-dimensional propagation paths, the engine shall identify critical dependency chains where a single point of failure can trigger widespread security degradation across the network. Based on this analysis, the engine shall address the complex challenge of modeling the threat likelihood of attack paths by assessing the compound risk posed by multi-stage attack scenarios. To accomplish this, CASTOR aims to explore advanced modeling approaches such as fault tree analysis to identify failure points and Markov chain modeling to quantify attack progression probabilities, enabling the engine to dynamically recommend RTL adjustments that accurately account for cascading threats in both dimensions. These recommendations shall prioritize mitigation of critical propagation paths by increasing RTL thresholds for nodes serving as potential stepping stones in network-wide attack chains or exhibiting cross-layer vulnerabilities that could enable privilege escalation. The cascading attack detection mechanism shall integrate with real-time monitoring data from TNDE to identify early indicators of multi-stage attacks in progress and trigger immediate RTL recalibration to prevent further propagation. **Remarks**: (1) Multi-Dimensional Propagation: The engine analyzes cascading attacks in both network topology (lateral node-to-node) and infrastructure stack (vertical cross-layer) dimensions, as well as combined attack scenarios that exploit both. (2) Dependency Chain Analysis: Risk graphs capture how compromise of one element enables attacks on others, identifying critical junction points where targeted RTL increases can effectively break attack propagation chains. (3) Proactive Defense: By identifying potential cascading scenarios before exploitation occurs, the engine enables preemptive RTL adjustments that prevent attackers from using compromised components as stepping stones toward more critical network elements. This detailed risk analysis produces accurate RTL calculations that enable precise trust-aware path provisioning decisions.
Connected to other requirements	SR.4, TAF.R.2

KPIs

Description	Value
RTL calculation methodologies	$\geq \mathbf{2}$ considering (static) tree-based attack paths (capturing a binary-based propagation model) vs. probabilistic analysis leveraging "what-if" scenarios (e.g., Markov-chain logic) embedding uncertainty in the threat propagation
Maximum time to generate and recommend RTL adjustments after detecting cascading attack patterns	≤ 2 seconds, focusing on the attack path calculation aspect.
Percentage of correctly identified critical junction points where RTL increases can break attack propagation chains	$\geq 90\%$

Table 9.37: RA.R.3 Black-box risk analysis

RA.R.3		
Title	**Black-box risk analysis**	
Actors Involved	Risk Assessment	
Type	Functional Requirement	
Description	**Background**: CASTOR's trusted path routing frequently spans multiple administrative domains, each managed by different organizations with distinct security policies, infrastructure configurations, and trust assessment methodologies. This capability is exemplified in UC4 Scenario 2, where CASTOR-enabled domains must incorporate risk indices from MNOs using internal risk assessment processes agnostic to CASTOR, enabling CASTOR to influence path selection based on external risk information while maintaining domain privacy. In inter-domain scenarios, a domain's Risk Assessment Engine cannot access detailed internal information about external domains' infrastructures due to privacy requirements, commercial confidentiality, and operational security constraints. Instead, external domains provide abstract trust summaries or risk indices, reflecting their overall security posture without revealing sensitive topology details, specific vulnerabilities, or security controls. Nonetheless, to compute accurate RTL values for end-to-end paths across multiple domains, the Risk Assessment Engine must still incorporate risk information from these external domains into its assessment process. This requires processing abstract risk representations for inter-domain path selection without compromising domain privacy. **Description**: The Risk Assessment Engine shall support black-box risk analysis by processing unverified security assertions and risk indices from external domains whose internal infrastructure characteristics are not fully disclosed. The engine shall critically evaluate these abstracted risk values, which can be expressed as aggregated trust scores, minimum trustworthiness thresholds, or domain-level risk ratings, without requiring access to detailed vulnerability assessments, topology information, or specific security control implementations of external domains. The engine shall incorporate these external risk indices into its RTL derivation process for inter-domain paths, while explicitly modeling the uncertainty and potential unreliability of externally-provided information. Following zero-trust principles, the engine shall never implicitly trust external assertions but rather treat them as additional risk factors that increase the required RTL for paths traversing those domains. The black-box analysis capability shall protect sensitive trustworthiness values during the risk assessment of external domains. The engine shall translate external risk formats to CASTOR's RTL scale and apply conservative adjustments when external information cannot be verified, flagging cases where insufficient information prevents reliable assessment. **Remarks**: (1) Privacy Preservation: The engine processes external risk indices without requiring disclosure of sensitive internal topology, vulnerability details, or security control implementations from external domains. (2) Modular Risk Assessment Framework: The Risk Assessment Engine shall be modular and extendable to explore harmonization approaches in which domain operators can follow their own risk methodologies while still derive required trust level (RTL) values compatible with the CASTOR framework. The engine shall investigate generic methodologies for translating diverse external risk representation formats to CASTOR's RTL scale, enabling interoperability without requiring domains to adopt CASTOR-specific trust assessment approaches. (3) Zero-Trust Uncertainty Handling: RTL derivation for inter-domain paths applies conservative trust requirements for all external domain segments, with additional penalties when operating with limited visibility or unverifiable information.	
Connected to other requirements	SR.16, RA.R.2, TAF.R.2, INTER-DOM.2	
KPIs	**Description**	**Value**
	Increase the accuracy of RTL value by incorporating external risk indices	$\geq 50\%$ Note: Assuming the existence of a baseline set of RTL values provided by a Security Analyst, a router vendor, or an MNO operator.

9.2.3.4 Policy Enforcement

Table 9.38: POLICY.R.1 Intra-domain translation to SLA (SSLA) and policies

POLICY.R.1		
Title	**Intra-domain translation to SLA (SSLA) and policies**	
Actors Involved	Intent Translation & Decomposition Service	
Type	Functional Requirement	
Description	**Background**: In highly dynamic and heterogeneous environments, there is a growing need to bridge the gap between what users or services want—expressed as high-level intents—and what the infrastructure can deliver in technical, enforceable terms. Translating these intents into Service Level Agreements (SLAs) and subsequently into domain-specific policies is crucial to ensure that user expectations around performance, security, and trust are met in a verifiable and automated manner. This translation mechanism enables intent-driven orchestration, where abstract objectives (e.g., "maintain secure low-latency communication") are decomposed into concrete, measurable guarantees and mapped to actionable enforcement rules. Such a capability not only supports interoperability across diverse domains but also enhances resilience, adaptability, and compliance, allowing the network to autonomously maintain desired service levels even under changing conditions or disruptions. **Description**: The Orchestration layer shall be capable of translating high-level user intents into measurable (Secure) Service Level Agreements ((S)SLAs) and subsequently into technical, enforceable policies within each administrative domain through the Service Intent Translation & Decomposition Service (SITDS). SITDS interprets abstract objectives (e.g., performance, security, and trust targets), decomposes them into (S)SLA terms and metrics, and then derives domain-specific enforcement rules by matching requirements to predefined path profiles in a path profile catalogue. By doing so, SITDS defines how (S)SLA trust requirements are translated to actionable domain-specific intents through the path profile catalogue (e.g., selecting isolation, crypto, telemetry, and remediation behaviors aligned with the chosen profile). The service produces policies consumable by policy engines and controllers in the domain, supports intradomain translation for heterogeneous capabilities, and maintains traceability from intent to (S)SLA. Out of scope are commercial or contractual negotiations of intents with service providers, brokering or arbitrating user objectives, and any pre-(S)SLA activities required to refine or reconcile intents until they can be formally expressed as (S)SLAs. **Remarks**: In order to perform the translation, the following information is required: the status of the network topology thanks to the Topology Graph and the complete catalog of path profiles.	
Connected to other requirements	OSS.R.2	
KPIs	**Description**	**Value**
	SSLA encoding interoperability.	Encoding SSLA clauses to be represented in a machine-readable format by extending well-adopted SLA schemas, to incorporate trust-related objectives.

Table 9.39: POLICY.R.2 Cross-domain translation to SLA (SSLA) and policies

POLICY.R.2	
Title	**Cross-domain translation to SLA (SSLA) and policies**
Actors Involved	Intent Translation & Decomposition Service
Type	Functional Requirement

Description	**Background**: In the context of the Compute Continuum, where services span from cloud to edge and across multiple administrative domains, the ability to enable trust-aware end-to-end (E2E) service provisioning is essential to ensure reliability, security, and accountability across heterogeneous infrastructures. This requirement becomes particularly important in CASTOR's use case scenarios, such as UAV inspection missions or V2X communications, where real-time decisions rely on dynamically composed network paths that must guarantee trusted behaviour under strict latency and security constraints. Without trust-aware coordination between domains, a single untrusted or misconfigured segment could compromise the integrity of the entire service chain, leading to operational risks or mission failure. Therefore, establishing a mechanism to maintain E2E trust and verifiable assurance across domains is fundamental to delivering resilient, predictable, and safe service performance in these mission-critical environments. **Description**: To extend cross-domain protocols to include trust-related information for cross-domain path establishment, CASTOR must address two distinct scenarios for parameter mapping between domains. In the first scenario, an SLA already exists between two domains, meaning that the mapping of attributes corresponding to each domain has been predefined as part of the bilateral agreement. This pre-established mapping enables direct definition of the End-to-End (E2E) trust requirements by leveraging the existing attribute correlations. In the second scenario, no predefined agreements or mappings exist between domains. In this case, CASTOR must dynamically create the parameter mapping by utilizing information that, for example, BGP-LS (Border Gateway Protocol - Link State) or other protocol extensions can provide regarding how trust is being measured and represented in each domain. This enables CASTOR to perform the necessary mapping between SLAs by understanding the trust measurement semantics and capabilities of each participating domain.
Connected to other requirements	OSS.R.2, INTER-DOM.TE.R.1
KPIs	**Description**: Enforcement of traffic engineering policies across multiple administrative domains; satisfying the established SSLA **Value**: Cross-domain TE policy enforcement across a minimum of (3) distinct administrative domains adhering to diverse trust models and semantics; i.e., different scalar measures of what a trustworthiness value represents.

9.2.3.5 Distributed Ledger Technologies

Table 9.40: DLT.R.1 Secure Storage of Security Claims

DLT.R.1	
Title	**Secure Storage of Security Claims**
Actors Involved	CASTOR DLT
Type	Functional Requirement
Description	**Background**: The foundational goal of the CASTOR system is to ensure secure path routing by continuously monitoring and establishing the trustworthiness of all nodes and links of a network. To achieve this, the system relies on collecting various types of security claims, such as attestation evidence (e.g., raw network traces) and trust reports from the Global TAF and the Local TAF agents. The introduction of Distributed Ledger Technologies (DLT) in CASTOR is a strategic decision to ensure the immutability and auditability of this sensitive security data. Therefore, a primary need is a mechanism that not only stores these claims but does so in a way that is verifiable, tamper-proof, and accessible only to authorized entities, thus creating a reliable historical trust record for the entire network's operation. This storage is crucial for post-mortem analysis (router vendors pinpointing issues) and continuous compliance monitoring (checking against trustworthiness thresholds defined in SLAs/SSLAs).

Description	**Description**: The CASTOR Blockchain shall enable the storage of security claims for secure path routing in a safe manner. These claims primarily include: (a) Attestation Evidence: Raw traces and claims detected by the CASTOR Trust Network Device Extension (TNDE), which indicate a failed runtime attestation task for a node or link. This data is recorded through dedicated TNDI-SP channels and is verified by the Secure Oracle before being stored, (b) Trust Reports: Periodic or event-driven reports generated by the Global TAF and Local TAF agents, which characterize the security posture of the infrastructure layer based on assessed trust propositions, and (c) (S)SLAs: These comprise the trustworthiness threshold where the network nodes and links should adhere to continuously, and the overarching trust framework should adapt to, based on the latest changes in the compute continuum network by enforcing new policies, if such threshold is not achieved. By leveraging the multi-layer access control framework presented in Section 6.2.3.1, CASTOR Secure Oracle shall enforce strict authorization checks before any security claim undergoes the final veracity check. Only upon successful verification by the Secure Oracle will the claims be recorded in the Private Ledger of the CASTOR DLT, ensuring their immutability and providing a reliable data source for auditable trustworthiness assessments. **Remarks**: (1) The Secure Oracle should leverage ABAC capabilities for enabling secure authorization of entities, complementarily to the verification of incoming data.
Connected to other requirements	SR.9, SR.16
KPIs	**Description**: Storage operation time **Value**: < 2 **sec** (regarding end-to-end measurement of sending/storage between the providing entity and the on-chain/off-chain storage facility)

Table 9.41: DLT.R.2 Authentication & Authorization in CASTOR Blockchain

	DLT.R.2
Title	**Authentication & Authorization in CASTOR Blockchain**
Actors Involved	CASTOR DLT
Type	Functional Requirement
Description	**Background**: The CASTOR Blockchain Infrastructure stores highly sensitive security and policy data, including attestation evidence, trust policies, and contractual obligations (SLA/SSLAs). To maintain the security, integrity, and privacy of the network operation, it is essential that only authenticated and authorized entities can interact with the DLT, whether for storing new data (e.g., trust policies, evidence) or retrieving existing data (e.g., diagnostic data, abstracted trust capabilities). In this sense, the use of Verifiable Credentials (VCs) and Verifiable Presentations (VPs), which contain attributes defining the entity's role and permissions, is crucial. Therefore, a robust mechanism for Attribute-Based Access Control (ABAC) is necessary to check these attributes against defined security policies, ensuring the principle of least privilege and maintaining the trustworthiness of the DLT as the central source of trust information. **Description**: The CASTOR Blockchain shall leverage the necessary ABAC mechanisms for granting access only to entities acquiring the appropriate attributes as part of their Verifiable Credentials. The CASTOR Blockchain Infrastructure shall implement strict authentication and authorization controls for all entities attempting to interact with the DLT functionalities (storage, retrieval, notification). The Security Context Broker (SCB) shall serve as an important trusted intermediary, responsible for vetting a critical part of incoming requests that are not received directly by the Secure Oracle. Every external CASTOR entity (such as local TAF agents, or router vendors) must present its Verifiable Presentations (VPs), which are derived from its Verifiable Credentials (VCs) issued by the CASTOR Trusted Computing Base.

Description	The ABAC service, integrated within the SCB workflow, shall process these VPs to perform a mandatory authorization check. This check involves verifying that the attributes contained in the VPs are correct and align with the necessary access rights for the requested action. For example, only authenticated router vendors, possessing the necessary access control attributes, can retrieve diagnostic data (e.g., failed attestation evidence) from the DLT for vulnerability discovery and auditing. This process ensures that access is granted exclusively to entities acquiring the appropriate attributes, thus securing the DLT's indispensable role in the CASTOR system. **Remarks**: The CASTOR Blockchain will introduce specifically three different levels of authorization-validation as elaborated in Section 6.2.3.1.
Connected to other requirements	SR.9, SR.16

KPIs	**Description**	**Value**
	Authorization time	$<$**15 ms** for ABAC operation This time duration focuses on the enforcement of the attribute-based access control policy and the validation result - i.e., whether the sender of a request is authorized or not. It does not include any VC-related processing or validation.

Table 9.42: DLT.R.3 Secure Oracle

DLT.R.3	
Title	**Secure Oracle**
Actors Involved	CASTOR DLT
Type	Functional Requirement
Description	**Background**: While the DLT ensures the immutability of stored data and ABAC ensures authorization, there is a critical need to guarantee the veracity and integrity of data before it is committed to the blockchain. Simply storing data from an authorized source is not sufficient; the data itself must be verified as authentic and correct, especially since the entire system's decision-making (e.g., Global TAF's trust calculations) hinges on this information. A Secure Oracle comprises a "fundamental innovation" specifically implemented to check the veracity of incoming data (like SLAs, SSLAs, and Trust Policies). This component is necessary to ensure the DLT is populated only with trustworthy information, thereby securing the foundation upon which the secure path routing decisions are made. **Description**: The CASTOR Blockchain shall leverage secure oracles capabilities towards ensuring data privacy protection, secure execution guarantees, and computational verifiability. In this sense, the CASTOR Blockchain Infrastructure shall incorporate the Secure Oracle component responsible for performing mandatory veracity checks on critical data before its final storage in the DLT's Private Ledger. This function is performed immediately after the Security Context Broker (SCB) has successfully processed the requesting entity, when necessary. The Secure Oracle shall specifically: (a) Verify Data Authenticity and Correctness: It must check the veracity of SLAs and SSLAs coming from the Facility Layer during the preparedness phase, and the authenticity of Trust Policies defined at the network enrolment stage. This ensures that the foundational security thresholds and rules stored on the DLT are accurate, (b) Enforce Secure Channel Integrity: For trust reports and trustworthiness evidence pushed from the Local TAF/TNDE, the Secure Oracle shall ensure that the recording of this information is strictly performed only through the provisioned, secure Trust Network Device Interface - Security Protocol (TNDI-SP) channels, maintaining data integrity from the network edge. By acting as a gatekeeper that performs crucial checks on data integrity and secure source channelling, the Secure Oracle provides computational verifiability and protection against the recording of malicious or erroneous data, thereby upholding the DLT's role as a trusted source for the CASTOR system.
Connected to other requirements	SR.9, SR.16

KPIs	**Description**	**Value**
	Veracity of incoming data	100%

Table 9.43: DLT.R.4 Network Trust Exposure Capability

DLT.R.4		
Title	**Network Trust Exposure Capability**	
Actors Involved	CASTOR DLT	
Type	Functional Requirement	
Description	**Background**: While the core DLT functionality focuses on internal security and auditability, the overall CASTOR project must provide value to its external stakeholders, such as potential Service Providers, and Federated Orchestration Services from other domains. These external entities need to assess the trustworthiness of the CASTOR network paths before or during the deployment of their services. Simply storing the trust data internally is not enough; the information must be made available in a controlled and secure manner. However, providing raw network details would compromise sensitive operational security. Therefore, there is a clear need for a dedicated function, the Trust Exposure Layer, that can securely and selectively provide trust capabilities to external stakeholders while simultaneously abstracting the data to prevent the revelation of any sensitive network details. **Description**: The CASTOR Blockchain shall expose the necessary interfaces for service providers to get access to the status of (S)SLA compliance with respect to their deployed services. In addition, based on the rich trust-related telemetry that is collected throughout the operational lifecycle of a domain, there is also the need to provide information about the trust capabilities of a domain without disclosing any privacy-sensitive (e.g., topology) information. In this sense, the Trust Exposure Layer shall expose the necessary abstraction functions to securely provision abstracted trust capabilities to authorized external stakeholders. In brief, The Trust Exposure Layer shall expose query capabilities which are able to securely process and return trust-related data (e.g., trust summaries of a domain or SSLA violation information) with the necessary level of abstraction. This ensures that the external stakeholders receive valuable, actionable information necessary for assessing network trustworthiness without revealing any sensitive, low-level network details about the compute continuum elements. Finally, access to the Trust Exposure Layer must still be subject to the existing CASTOR DLT authorization mechanisms, ensuring that only authorized external stakeholders can retrieve this trust-related information.	
Connected to other requirements	SR.9, SR.16	
	Description	**Value**
	Abstraction of topological data	100%

9.2.4 Traffic Engineering Requirements

Table 9.44: TE.R.1 Automated Traffic Engineering TE capabilities (to cope with network conditions dynamicity)

TE.R.1		
Title	**Automated Traffic Engineering TE capabilities (to cope with network conditions dynamicity)**	
Actors Involved	Traffic Engineering	
Type	Functional Requirement	
Description	**Background**: In the CASTOR architecture, automated TE capabilities are introduced to enable the dynamic adaptation of traffic in view of changing network conditions. Traditional routing approaches may be capable to identify (shortest-path) routes but lack the flexibility to establish trust-driven paths that at the same time meet network requirements, across the (AS-level) domain. The intended integration of trust information to TE primitives, will enable the network to automatically compute and steer traffic along (trust- and network-optimal paths) by including or excluding specific links or devices based on predefined constraints. The expected steering mechanism of increased automation aims to minimize manual configuration and operational overhead but also seeks to enable fine-grained, policy-compliant routing decisions. By integrating trust information exchange with automated path computation, CASTOR pursue continuous alignment between service intents, trust levels, and performance objectives across distributed domains, enabling secure and optimized routing in heterogeneous (i.e., equipment- and capabilities-wise) network environments. **Description**: The CASTOR Service Orchestration layer shall provide the mechanisms to manage the dynamic configuration of network elements such as routers by creating, modifying, or deleting their settings in response to topology changes, while simultaneously handling mixed-criticality requirements through distinct overlay topologies. For example, a prominent solution is to encode network and trust requirements as Flex Algo configuration, where each Flex Algo instance defines a virtual topology containing only the nodes, links, and paths that satisfy a specific set of trust requirements established in the SSLA. This approach enables service providers to create multiple such topologies corresponding to different trust profiles within the service catalogue, allowing dynamic path selection based on changing trust conditions. This solution combined with link coloring, based on trust profiles derived by the CASTOR Trust Assessment Framework (TAF), ensuring the required automated TE capabilities will ensure that the network remains correctly configured during unexpected changes, establishing paths that aggregate or discriminate nodes according to specific requirements such as latency, bandwidth, reliability, and trust levels.	
Connected to other requirements	OSS.R.1, OSS.R.2, OSS.R.3	
KPIs	**Description**	**Value**
	Time to adapt to trust and topology changes	≤ 5 sec, Time to identify and utilise an alternative path. *NOTE: The time that elapses between the PCE obtaining the new policies and the data flow being modified.*

Table 9.45: TE.R.2 Constraint-based path computation capabilities (for domain-wide traffic)

TE.R.2	
Title	**Constraint-based path computation capabilities (for domain-wide traffic)**
Actors Involved	Traffic Engineering and Path Computation Element
Type	Functional Requirement

Description	**Background**: Network topologies undergo constant changes in performance based on the irregular and unpredictable needs of users and services. Furthermore, in the CASTOR context, trust values of the underlying network elements may attain time-varying values. Addressing such kind of dynamicity beyond manual (re)configuration, CASTOR leverages centralised TE intelligence for efficient network configuration and global (i.e., domain-wide) optimization. The PCE (Path Computation Element) functionalities enables dynamic, trust-aware traffic engineering within the domain scope, focusing specifically on minimizing the configuration footprint at the data and forwarding plane. Traditional routing and policy enforcement systems require extensive configurations at multiple network points, leading to redundancy, inconsistent updates, and operational fragility. **Description**: To significantly reduce the actions required by the network operator and streamline deployment at the infrastructure layer, we adopt the Path Computation Element (PCE) solution—a centralized intelligence responsible for computing optimal, policy-compliant paths based on network state, intent requirements, and trust constraints. By delegating path computation to a PCE, each domain node can operate with a minimal configuration footprint, reducing management overhead while ensuring that routes adapt automatically to performance variations or trust degradations in CASTOR. This engineering choice streamlines control, offers increased optimization capabilities by centrally gathering the required inputs and overall establishes a foundation for resilient, low-maintenance TE engineering (facilitated by the CASTOR orchestration) across the (AS) domain.
Connected to other requirements	OSS.R.2

KPIs	**Description**	**Value**
	PCE computation latency overhead	$\leq 25\%$ overhead imposed by CASTOR framework to the PCE computation latency. This overhead includes the time for the PCE to return a computed path (including any optimization computations), proportional to the time required for the SSLA function that is related to it. *NOTE: The absolute latency of the PCE functionality is heavily dependent on the software and hardware capabilities on top of which the PCE logic is running.*

Table 9.46: INTER-DOM.TE.R.1 Trust summary exchange

INTER-DOM.TE.R.1	
Title	**Interdomain exchange of trust summaries**
Actors Involved	Trust Exposure Layer, Trust Assessment Framework
Type	Functional Requirement
Description	**Background**: In order for CASTOR to establish a trusted end-to-end path traversing several domains, trust related information should be exchanged between involved domains. The already exchanged reachability and routing information (see Chapter 3) between (two) domains should be complemented with trust related information Therefore, in addition to the network performance quantities (e.g. latency, throughput), the cross-domain information exchange should include trust related assurances. In detail, the expected trust related part should prescribe the different levels of trust supported by the domain, together with a mechanism that allows for the meaningful comparison of the trust summaries (of different domains). **Description**: Each domain exchanges trust related information through a mechanism that needs to gather a relevant synopsis of the domain trust and make it available to the adjacent domain. The communication of the synopsis can rely on (the extension of) well-established inter-domain protocols or utilise direct links between the corresponding domain orchestrators.

Description	**Remarks**: A Trust Summary does not contain any domain-sensitive information about the infrastructure, topology, or network (in)stability. The concept of a trust synopsis — expressed as abstract trust claims — is to convey the minimal trust declarations, adequately abstracted, to enable the establishment of trustful service unions across multiple domains. This may include, for example, the minimum trust level exhibited by a domain-specific segment. However, this raises a second question: to meaningfully evaluate the end-to-end common trust level, there must be a shared scalar reference that defines what each trust value represents — for instance, "medium integrity" should correspond to the same type of requirements across all domains. The definition of this common semantics is already agreed upon at the MNO level. Through these functional requirements, CASTOR fills the gap by extending mutual AS agreements with the appropriate encoding.	
Connected to other requirements	TAF.R.2, TE.R.1	
KPIs	**Description**	**Value**
	Trust summaries shall be interpretable by neighbouring domains as part of the E2E service provisioning.	TRUE
	Establish mechanisms for exchange of trust summaries across domains.	TRUE.

Chapter 10

Summary and Conclusions

The current deliverable deals on who to engrain trust as part of the traffic engineering process, which is not a new concept, but now is becoming the focal point of the industry, serving as the cornerstone for the design of the CASTOR framework. Several proposals exist, such as the SCION, with different maturity levels. CASTOR builds on top of these works, by identifying several foundational security, trust, operational and traffic engineering requirements and the corresponding KPIs. CASTOR KPI values are not just target or threshold values that characterize the success and impracticality of a solution in the Compute Continuum landscape, since there is not such a baseline. CASTOR's strategy with the definition of the architecture and the KPI values is to identify the operational boundaries regarding the complexity and the minimal performance footprint (operational profiling) and not the applicability. Thus, the reasoning behind its selection is also documented in the deliverable.

In addition to the requirements, D2.1 delivered a comprehensive view of the overall CASTOR complex architecture, its individual in-router artifacts and their interactions. CASTOR's conceptual architecture aims to satisfy the requirements that have been formulated during the requirements analysis phase. It has also defined in more detail the four use cases of the project, going down into the level of user stories that will be used to validate the overall CASTOR framework. The deliverable also fleshes out the questions that need to be answered as part of the technical deliverables. More details regarding the internal functionalities, interactions and interfaces of the individual components will be analysed in the upcoming deliverables of WP3, WP4 and WP5. Last but not least, D2.1 is the predecessor of the final version of the deliverable (D2.2) that will consider the cross-domain end-to-end service provisioning.

List of Abbreviations

Abbreviation	Translation
ACC	Assisted Cruise Control
AIC	Attestation Integrity Verification
ZKP	Zero-Knowledge Proof
5GC	5G core
ABAC	Attribute-Based Access Control
ACO	Ant Colony Optimization
ALTO	Application-Layer Traffic Optimization
AMF	Access and Mobility Management Function
AS	Application Server
AS	Autonomous System
ASD	Aftermarket Safety Device
ATL	Actual Level of Trust
AUSF	Authentication Server Function
BBL	Basic Block Level
BGP	Border Gateway Protocol
BGP-LS	Border Gateway Protocol Link-State
BLS	Boneh–Lynn–Shacham
CA	Certificate Authority
CAM	Cooperative Awareness Message
CER	Central emergency responder
CFA	Control Flow Attestation
CFSM	Communicating FSM
CIM	Coherent Ising Machine
CN	Core Network
CNI	Container Network Interfaces
CRL	Certificate Revocation List
CRL-CA	Certificate Revocation CA
CSP	Communication Service Provider
CSP	Cloud Service Provider
CSC	Communication Service Customer
CTI	Cyber Threat Intelligence
CXP	Control Exchange Points
DENM	Decentralized Environmental Notification Message
DMA	Direct Memory Access
DLT	Distributed Ledger Technology
DoS	Denial of Service
DRL	Deep Reinforcement Learning
eBPF	extended Berkeley Packet Filter

ECA	Enrolment Certification Authority
ETSI	European Telecommunications Standards Institute
Flex-Algo	Flexible Algorithm
FSM	Finite-State Machines
E2E	End-to-End
E2EE	End-to-End Encrypted
GCS	Ground Control Station
GSP	GEO Service Provider
I-UPF	Intermediate User Plane Functions
IBN	Intent-Based Networking
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
ITS	Intelligent Transportation System
L1	Layer 1
LDP	Label Distribution Protocol
LER	Local Emergency Responder
LSA	Link State Advertisements
LTS	labelled Transition Systems
MANET	Mobile Ad Hoc Networks
MORP	Multi-Objective Routing Optimization Problem
MPLS	Multi-Protocol Label Switching
MVP	Minimum Viable Product
NEP	Network Equipment Provider
NFV	Network Functions Virtualization
NISQ	Noisy Intermediate-Scale Quantum
NOP	Network Operator
OBU	Onboard Unit
ORE	Order-Revealing Encryption
OSPF	Open Shortest Path First
OTA	Over-the-Air
OSS/BSS	Operational Support System/Business Support System
PCA	Pseudonym Certification Authority
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	Path Computation Element Communication Protocol
PKI	Public Key Infrastructure
PRF	Pseudo-Random Function
QA	Quantum Annealing
QAOA	Quantum Approximate Optimization Algorithm
QUBO	Quadratic Unconstrained Binary Optimization
QoS	Quality of Service
RAT	Remote Attestation Procedure
RCA	Root Certificate Authority
RIB	Routing Information Base
RIP	Routing Information Protocol
RoT	Root of Trust
RSU	Roadside Unit
RSVP-TE	Resource Reservation Protocol for Traffic Engineering
RTL	Required Trust Level

SB	Simulated Bifurcation
SCB	Security Context Broker
SCMS	Security Credential Management System
SDN	Software-Defined Networking
SITDS	Service Intent Translation & Decomposition Service
SLA	Service Level Agreement
SMD	Security Management Domain
SMF	Session Management Function
SotA	State-of-the-Art
SR	Segment Routing
SRv6	Segment Routing over IPv6
SSI	Self-Sovereign Identity
SSLA	Security Service Level Agreements
SSLO	Security and Service Level Objective
TAF	Trust Assessment Framework
TDE	Trust Decision Engine
TDI	TEE Device Interface
TE	Traffic Engineering
TEE	Trusted Execution Environment
TLEE	Trustworthiness Level Expression Engine
TMM	Trust Model Manager
TN-DSM	Trust Network Device Security Monitor
TNDE	Trust Network Device Extension
TNDI	Trust Network Device Interfaces
TNDISP	TNDI security protocol
TO	Traffic Operator
TSM	Trust Source Manager
TPL	Trust Policy Language
TPM	Trusted Platform Module
TPR	Trusted Path Routing
TVM	Trusted Execution Environment Virtual Machine
UAV	Unmanned Aerial Vehicle
UE	User Equipment
UPF	User Plane Function
UTM	Unmanned Traffic Management
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VNF	Virtual Network Function
VPN	Virtual Private Network
VQE	Variational Quantum Eigensolver
VRU	Vulnerable Road Users
ZK	Zero knowledge
ZKP	Zero knowledge proof
ZSM	Zero-touch network and Service Management

Bibliography

[1] Falco: Real-time threat detection solution for containers, hosts, Kubernetes and the cloud. https://github.com/falcosecurity/falco.

[2] TEE Device Interface Security Protocol (TDISP). https://pcisig.com/tee-device-interface-security-protocol-tdisp.

[3] Tetragon: eBPF-based Security Observability and Runtime Enforcement. https://github.com/cilium/tetragon.

[4] Intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service. Technical Report ETSI TS 102 637-3, European Telecommunications Standards Institute, September 2010.

[5] Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. Technical Report ETSI TS 102 637-2, European Telecommunications Standards Institute, March 2011.

[6] Network Functions Virtualisation (NFV); Management and Orchestration. Etsi gs nfv-mano 001 v1.1.1, ETSI, Sophia Antipolis, France, 2014.

[7] Autonomics; Enablers for autonomous management. Etsi gr nfv-ifa 041 v4.1.1, ETSI, Sophia Antipolis, France, 2021.

[8] Intelligent transport systems (its); security; its communications security architecture and security management;. Technical Report ETSI TS 102 940-2.1.1, European Telecommunications Standards Institute, July 2021.

[9] 5G Automotive Association (5GAA). 5gaa white paper on c-v2x use cases. White Paper P-1801062, Version 1.0, 5G Automotive Association, Munich, Germany, 2019.

[10] 5G Automotive Association (5GAA). Road operator use case modelling and analysis. Technical Report Report 2, 5G Automotive Association, Munich, Germany, 2023.

[11] 5G Automotive Association (5GAA). 5gaa – connected mobility for people, vehicles, and transport infrastructure. https://5gaa.org/, 2025. Accessed: 2025-11-21.

[12] 5G Automotive Association (5GAA). C-v2x use cases and service level requirements — volume III. Technical Report TR, 5G Automotive Association, Munich, Germany, 2025.

[13] A. Abbas et al. Challenges and opportunities in quantum optimization. Nature Reviews Physics, pages 1–18, 2024.

[14] Tigist Abera, N. Asokan, Lucas Davi, Jan-Erik Ekberg, Thomas Nyman, Andrew Paverd, Ahmad-Reza Sadeghi, and Gene Tsudik. C-FLAT: Control-Flow Attestation for Embedded Systems Software. In ACM SIGSAC Conference on Computer and Communications Security, CCS '16, 2016.

[15] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In Proceedings of the 2004 ACM SIGMOD international conference on Management of data, pages 563–574, 2004.

[16] Esteban Aguilera, Jins de Jong, Frank Phillipson, Skander Taamallah, and Mischa Vos. Multi-objective portfolio optimization using a quantum annealer. Mathematics, 12(9):1291, 2024.

[17] H. B. Akande, O. C. Abikoye, U. A. Jauro, and S. A. Salihu. Meta-heuristic optimization algorithms for network routing: A survey. Journal of Computer Science and Control Systems, 12(1):5–8, 2019.

[18] Rajeev Alur and David L. Dill. A theory of timed automata. Theoretical Computer Science, 126(2):183–235, 1994.

[19] Mahmoud Ammar, Adam Caulfield, and Ivan De Oliveira Nunes. SoK: Integrity, Attestation, and Auditing of Program Execution. In IEEE Symposium on Security and Privacy (SP). IEEE Computer Society, 2025.

[20] Dana Angluin. Learning regular sets from queries and counterexamples. Information and Computation, 75(2):87–106, 1987.

[21] Daniele Enrico Asoni, Takayuki Sasaki, and Adrian Perrig. Alcatraz: Data Exfiltration-Resilient Corporate Network Architecture. In IEEE 4th International Conference on Collaboration and Internet Computing (CIC), 2018.

[22] SCION Association. SCION — scalability, control, and isolation on next-generation networks. https://scion-architecture.net/, 2023.

[23] D. Basile, V. Goretti, C. Di Ciccio, and S. Kirrane. Enhancing blockchain-based processes with decentralized oracles. In Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2021 Blockchain and RPA Forum, Rome, Italy, September 6–10, 2021, Proceedings, pages 102–118, Cham, 2021. Springer International Publishing.

[24] Carsten Baum, Bernardo David, Elena Pagnin, and Akira Takahashi. Universally composable interactive and ordered multi-signatures. In IACR International Conference on Public-Key Cryptography, pages 3–31, 2025.

[25] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on uppaal. In Marco Bernardo and Flavio Corradini, editors, Formal Methods for the Design of Real-Time Systems, volume 3185 of Lecture Notes in Computer Science, pages 200–236. Springer, 2004.

[26] H. Birkholz, T. Fossati, W. Pan, and C Bormann. Epoch Markers. https://www.ietf.org/archive/id/draft-ietf-rats-epoch-markers-01.txt, April 2025.

[27] H. Birkholz, E. Voit, C. Liu, D. Lopez, and M. Chen. Trusted Path Routing. https://www.ietf.org/archive/id/draft-voit-rats-trustworthy-path-routing-11.html, January 2025.

[28] Henk Birkholz, Dave Thaler, Michael Richardson, Ned Smith, and Wei Pan. Remote ATtestation procedureS (RATS) Architecture. RFC 9334, January 2023.

[29] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. Order-preserving symmetric encryption. In Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28, pages 224–241. Springer, 2009.

[30] Alexandra Boldyreva, Craig Gentry, Adam O'Neill, and Dae Hyun Yum. Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In ACM CCS, pages 276–285, 2007.

[31] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 563–594. Springer, 2015.

[32] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In International conference on the theory and application of cryptology and information security, pages 514–532. Springer, 2001.

[33] Maria C Borges, Joshua Bauer, and Sebastian Werner. Oxn-automated observability assessments for cloud-native applications. In 2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C), pages 167–170. IEEE, 2024.

[34] Daniel Brand and Pitro Zafiropulo. On communicating finite-state machines. Journal of the ACM, 30(2):323–342, April 1983.

[35] Marcus Brandenburger, Christian Cachin, Rüdiger Kapitza, and Alessandro Sorniotti. Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric. Technical report, arXiv, 2018. Preprint — describes the design of Fabric Private Chaincode (FPC).

[36] A. Callison and N. Chancellor. Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond. Physical Review A, 106:010101, 2022.

[37] Brian E. Carpenter, Bing Liu, Mohamed Boucadair, Yu Fu, and Qin Wu. Autonomic control plane. RFC 8994, May 2021.

[38] Emiliano Casalicchio. Container Orchestration: A Survey, pages 221–235. 01 2019.

[39] Marco Casazza, Mathieu Bouet, and Stefano Secci. Availability-driven nfv orchestration. Computer Networks, 155, 03 2019.

[40] CASTOR. Architectural specification of castor continuum-wide trust assessment framework. Deliverable 4.1, The CASTOR Consortium, 15 2026.

[41] CASTOR. Architectural specification of dynamic enforcement of trust-/network-aware path establishments. Deliverable 5.1, The CASTOR Consortium, 15 2026.

[42] Lin Chen, Rong Yuan, and Yufei Xia. Tora: A trusted blockchain oracle based on a decentralized tee network. In 2021 IEEE International Conference on Joint Cloud Computing (JCC), pages 28–33. IEEE, August 2021.

[43] Nathan Chenette, Kevin Lewi, Stephen A Weis, and David J Wu. Practical order-revealing encryption with limited leakage. In Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers 23, pages 474–493. Springer, 2016.

[44] Mingxi Cheng, Chenzhong Yin, Junyao Zhang, Shahin Nazarian, Jyotirmoy Deshmukh, and Paul Bogdan. A general trust framework for multi-agent systems. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '21, page 332–340, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

[45] Laurent Chuat, Markus Legner, David Basin, David Hausheer, Samuel Hitz, Peter Müller, and Adrian Perrig. The Complete Guide to SCION: From Design Principles to Formal Verification. Information Security and Cryptography. Springer Cham, 1 edition, 2022.

[46] Cisco Systems, Inc. MPLS Quality of Service (QoS) — Cisco IOS XE Multiprotocol Label Switching (MPLS) Configuration Guide, 2009. Accessed: 2025-11-24.

[47] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. Model Checking. MIT Press, Cambridge, MA, 1999.

[48] Alexander Clemm, Laurent Ciavaglia, Lisandro Z. Granville, and Jeff Tantsura. Intent-Based Networking - Concepts and Definitions. RFC 9315 (Informational), October 2022.

[49] Mike Cohn. User stories applied: For agile software development. Addison-Wesley Professional, 2004.

[50] E. Davies, D. Banfield, V. Cărare, B. Weaver, C. White, and N. Walker. Optical routing with binary optimisation and quantum annealing. In 2024 International Conference on Optical Network Design and Modeling (ONDM), pages 1–6, 2024.

[51] Corine de Kater, Nicola Rustignoli, and Samuel Hitz. SCION Control Plane. Internet-Draft draft-dekater-scion-controlplane-12, Internet Engineering Task Force, November 2025. Work in Progress.

[52] Corine de Kater, Nicola Rustignoli, Jean-Christophe Hugly, and Samuel Hitz. SCION Data Plane. Internet-Draft draft-dekater-scion-dataplane-08, Internet Engineering Task Force, November 2025. Work in Progress.

[53] Joeri de Ruiter and Erik Poll. Protocol state fuzzing of TLS implementations. In Proceedings of the 24th USENIX Security Symposium, pages 193–206, Washington, D.C., 2015. USENIX Association.

[54] Theo Dimitrakos, Tezcan Dilshener, Alexander Kravtsov, Antonio La Marra, Fabio Martinelli, Athanasios Rizos, Alessandro Rosetti, and Andrea Saracino. Trust aware continuous authorization for zero trust in consumer internet of things. In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pages 1801–1812, 2020.

[55] M. Doole, J. Ellerbroek, and J. Hoekstra. Drone delivery: Urban airspace traffic density estimation. In Proceedings of the 8th SESAR Innovation Days, Salzburg, Austria, December 2018. SESAR. 3–7 December 2018.

[56] Mirna El Rajab, Li Yang, and Abdallah Shami. Zero-touch networks: Towards next-generation network automation. Computer Networks, 243:110294, 2024.

[57] Reese Enghardt and Cyrill Krähenbühl. A Vocabulary of Path Properties. RFC 9473, September 2023.

[58] Samaneh Eskandari, Mohammad Salehi, Wen-Chen Gu, and Jeremy Clark. Sok: Oracles from the ground truth to market manipulation. In Proceedings of the 3rd ACM Conference on Advances in Financial Technologies, pages 127–141, September 2021.

[59] ETSI, Dec 2024.

[60] ETSI Industry Specification Group NFV-SEC. Network functions virtualisation (nfv); trust; report on attestation technologies and practices for secure deployments. Technical Report GR NFV-SEC 007 V1.2.1, European Telecommunications Standards Institute, November 2024.

[61] EUROCONTROL and SESAR 3 Joint Undertaking (CORUS-XUAM). U-space concept of operations (conops) — fourth edition. Technical report, Publications Office of the European Union, Luxembourg, 2023. Official publication of the CORUS-XUAM project, Deliverable D4.2, Edition 01.00.02, July 2023. Accessed November 2025.

[62] European Commission, DG Communications Networks Content and Technology. White paper – how to master europe's digital infrastructure needs? White Paper COM(2024) 81 final, European Commission, February 2024. Accessed: 2025-11-30.

[63] European Telecommunications Standards Institute (ETSI). Etsi gs nfv-sec 003 v1.1.1 (2014-12): Network functions virtualisation (nfv) whitepaper. Standard GS NFV-SEC 003 V1.1.1, ETSI, Sophia Antipolis, France, 2014.

[64] C. Filsfils, K. Talaulikar, D. Voyer, A. Bogdanov, and P. Mattes. Segment routing policy architecture. RFC 9256, July 2022. Standards Track, updates RFC 8402.

[65] Filsfils, C. and Previdi, S. and Ginsberg, L. and Decraene, B. and Litkowski, S. and Bashandy, A. Segment routing architecture. RFC 8402, Internet Engineering Task Force (IETF), july 2018. RFC 8402.

[66] D. Gambetta and P.S.O.F.D. Gambetta. Trust: Making and Breaking Cooperative Relations. B. Blackwell, 1988.

[67] Tal Garfinkel and Mendel Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In Proceedings of the NDSS Symposium 2003, NDSS, 2003.

[68] Keno Garlichs, Alexander Willecke, Martin Wegner, and Lars C. Wolf. Trip: Misbehavior detection for dynamic platoons using trust. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pages 455–460, 2019.

[69] Kuntal Gaur, Anshuman Kalla, Jyoti Grover, Mohammad Borhani, Andrei Gurtov, and Madhusanka Liyanage. A survey of virtual private lan services (vpls): Past, present and future. Computer Networks, 196:108245, 2021.

[70] Xinyang Ge, Weidong Cui, and Trent Jaeger. GRIFFIN: Guarding Control Flows Using Intel Processor Trace. In Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '17. ACM, 2017.

[71] Nitesh Ghodichor, Dinesh Sahu, Gautam Borkar, and Ankush Sawarkar. Secure routing protocol to mitigate attacks by using blockchain technology in manet. International journal of Computer Networks & Communications, 15:127–146, 03 2023.

[72] Georgios Giantamidis, Stylianos Basagiannis, and Stavros Tripakis. Learning Moore machines from input–output traces. International Journal on Software Tools for Technology Transfer (STTT), 23:85–104, 2021. First online 2019.

[73] A. Y. Hamed, M. H. Alkinani, and M. Hassan. A genetic algorithm optimization for multi-objective multicast routing. Constraints, 6:10, 2020.

[74] P. Hauke, G. Mattiotti, and P. Faccioli. Dominant reaction pathways by quantum computing. Physical Review Letters, 126:028104, 2021.

[75] Gerard J. Holzmann. The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston, MA, 2003.

[76] Krontiris I, Giannetsos T., and Birkholz H. Extending Trusted Path Routing: Issues in Runtime Trust Assessment and monitoring. https://www.ietf.org/archive/id/draft-rats-runtime-tpr-00.html, 2025.

[77] International Civil Aviation Organization (ICAO). Unmanned aircraft systems traffic management (utm) – a common framework with core principles for global harmonization, edition 4. https://www.icao.int/sites/default/files/left-menu-pdfs/UTM%20Framework%20Edition%204.pdf, 2025. Accessed November 2025.

[78] Malte Isberner, Falk Howar, and Bernhard Steffen. The open-source learnlib: A framework for active automata learning. In Computer Aided Verification (CAV 2015), volume 9206 of Lecture Notes in Computer Science, pages 487–495. Springer, 2015.

[79] Bhushan Jain, Mirza Basim Baig, Dongli Zhang, Donald E. Porter, and Radu Sion. SoK: Introspections on Trust and the Semantic Gap. In IEEE Symposium on Security and Privacy, 2014.

[80] A. T. Jawad, R. Maaloul, and L. Chaari. A comprehensive survey on 6g and beyond: Enabling technologies, opportunities of machine learning and challenges. Computer Networks, 237:110085, 2023.

[81] Hanwei Jia, Fengjun Zhao, Zhong Li, Yuhao Liu, Fangsen Liu, Kang Wang, and Pei Zhang. MEC Data Offloading Strategy for UPF Sinking in 5G Core Network, page 63–71. Springer Nature Singapore, 2024.

[82] Audun Jøsang. Subjective logic, volume 3. Springer, 2016.

[83] Audun Josang, Dongxia Wang, and Jie Zhang. Multi-source fusion in subjective logic. In 2017 20th International Conference on Information Fusion (Fusion), pages 1–8, 2017.

[84] Hu Junping, Jin Yuhui, and Dou Liang. A time-based cluster-head selection algorithm for leach. In 2008 IEEE Symposium on Computers and Communications, pages 1172–1176, 2008.

[85] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu. Ten challenges in advancing machine learning technologies toward 6g. IEEE Wireless Communications, 27(3):96–103, 2020.

[86] Henna Kokkonen, Lauri Lovén, Naser Hossein Motlagh, Abhishek Kumar, Juha Partala, Tri Nguyen, Víctor Casamayor Pujol, Panos Kostakos, Teemu Leppänen, Alfonso González-Gil, et al. Autonomy and intelligence in the computing continuum: Challenges, enablers, and future directions for orchestration. arXiv preprint arXiv:2205.01423, 2022.

[87] Cyrill Krähenbühl, Marc Wyss, David Basin, Vincent Lenders, Adrian Perrig, and Martin Strohmeier. Fabrid: Flexible attestation-based routing for inter-domain networks. In USENIX Security Symposium, 2023.

[88] Heba A. Kurdi, Bushra Alshayban, Lina Altoaimy, and Shada Alsalamah. Trustyfeer: A subjective logic trust model for smart city peer-to-peer federated clouds. Wirel. Commun. Mob. Comput., 2018, 2018.

[89] James F. Kurose and Keith W. Ross. Computer Networking: A Top-Down Approach (6th Edition). Pearson, 6th edition, 2012.

[90] Andrea Lanzi, Davide Balzarotti, Christopher Kruegel, Mihai Christodorescu, and Engin Kirda. AccessMiner: using system-centric models for malware protection. In 17th ACM Conference on Computer and Communications Security, CCS '10. ACM, 2010.

[91] Dayeol Lee, Kevin Cheang, Alexander Thomas, Catherine Lu, Pranav Gaddamadugu, Anjo Vahldiek-Oberwagner, Mona Vij, Dawn Song, Sanjit A Seshia, and Krste Asanovic. Cerberus: A formal approach to secure and efficient enclave memory sharing. In ACM SIGSAC Conference on Computer and Communications Security, pages 1871–1885, 2022.

[92] Young Lee, Dhruv Dhody, Giuseppe Fioccola, Qin Wu, Daniele Ceccarelli, and Jeff Tantsura. Traffic engineering (te) and service mapping yang data model. Internet-Draft, IETF, October 2025. Expires April 15, 2026.

[93] Kevin Lewi and David J Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 1167–1178, 2016.

[94] Irian Leyva-Pupo, Cristina Cervelló-Pastor, Christos Anagnostopoulos, and Dimitrios P. Pezaros. Dynamic upf placement and chaining reconfiguration in 5g networks. Computer Networks, 215:109200, October 2022.

[95] Yuan Li, Hongbing Wang, and Yunlei Zhao. Delegatable order-revealing encryption. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pages 134–147, 2019.

[96] Michael Hale Ligh, Andrew Case, Jamie Levy, and Aaron Walters. The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory. Wiley Publishing, 1st edition, 2014.

[97] C. Liu, L. Iannone, D. Lopez, A. Pastor, M. Chen, and L. Su. Nasr use case and requirements. Technical Report draft-liu-nasr-requirements-03, Internet Engineering Task Force (IETF), October 2024. Internet-Draft, intended status Informational, expires 24 April 2025.

[98] Francesco Lombardo, Stefano Salsano, Ahmed Abdelsalam, Daniel Bernier, and Clarence Filsfils. Extending kubernetes networking to make use of segment routing over ipv6 (srv6). arXiv preprint arXiv:2301.01178, 2023.

[99] Dominik Lorych and Lukas Jäger. Design space exploration of dice. In Proceedings of the 17th International Conference on Availability, Reliability and Security, pages 1–10, 2022.

[100] Andrew Lucas. Ising formulations of many np problems. Frontiers in physics, 2:5, 2014.

[101] Chunyang Lv, Jianfeng Wang, Shi-Feng Sun, Yunling Wang, Saiyu Qi, and Xiaofeng Chen. Efficient multi-client order-revealing encryption and its applications. In European Symposium on Research in Computer Security, pages 44–63. Springer, 2021.

[102] Bruno Martini, Daniele Borsatti, Julio García Sáez, and Salvatore Privitera. Intent-based network slicing for sdn vertical services with assurance: Context, design and preliminary experiments. Future Generation Computer Systems, 142:101–116, 2023.

[103] A. Maskooki, K. Deb, and M. Kallio. A customized genetic algorithm for biobjective routing in a dynamic network. European Journal of Operational Research, 297(2):615–629, 2022.

[104] Barry M McCoy and Tai Tsun Wu. The two-dimensional Ising model. Harvard University Press, 1973.

[105] Rob McManus, Mark Longwell, Hanen Garcia, Charlie Ashton, and O'Toole Rory. Unlocking the potential of 5g at the edge, *Red Hat, Inc.*, 2025. [Online] Available: https://www.redhat.com/en/blog/unlocking-potential-5g-edge. Accessed: 2025-10-16.

[106] Ralph C Merkle. A certified digital signature. In Conference on the Theory and Application of Cryptology, pages 218–238. Springer, 1989.

[107] Jesper Møller and Brian Carpenter. Autonomic networking: Definitions and design goals. RFC 7575, June 2015.

[108] Jad Naous, Michael Walfish, Antonio Nicolosi, David Mazières, Michael Miller, and Arun Seehra. Verifying and enforcing network paths with icing. In Proceedings of the 7th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT 2011). ACM, 2011.

[109] S. J. Nawaz, S. K. Sharma, S. Wyne, M. N. Patwary, and M. Asaduzzaman. Quantum machine learning for 6g communication networks: State-of-the-art and vision for the future. IEEE Access, 7:46317–46350, 2019.

[110] Phala Network. Phala network introduces phat contract into saas3 to build a highly scalable oracle. Medium, January 2023.

[111] Thanh Nguyen, Meni Orenbach, and Ahmad Atamli. Live system call trace reconstruction on Linux. Forensic Science International: Digital Investigation, 2022. Proceedings of the Twenty-Second Annual DFRWS USA.

[112] Jose Oncina and Pedro García. Inferring regular languages in polynomial update time. Pattern Recognition and Image Analysis, 1:49–61, 1992.

[113] Meni Orenbach, Rami Ailabouni, Nael Masalha, Thanh Nguyen, Amhad Saleh, Frank Block, Fritz Alder, Ofir Arkin, and Ahmad Atamli. BlueGuard: Accelerated Host and Guest Introspection Using DPUs. In USENIX Security Symposium (USENIX Security 25). USENIX Association, 2025.

[114] Koffi Ismael Ouattara, Ana Petrovska, Artur Hermann, Nataša Trkulja, Theo Dimitrakos, and Frank Kargl. On subjective logic trust discount for referral paths. In 2024 27th International Conference on Information Fusion (FUSION), pages 1–8, 2024.

[115] Amir Pasdar, Y. C. Lee, and Zhe Dong. Connect api with blockchain: A survey on blockchain oracle implementation. ACM Computing Surveys, 55(10):1–39, 2023.

[116] E. Pelofske, A. Bärtschi, and S. Eidenbenz. Quantum annealing vs. qaoa: 127 qubit higher-order ising problems on nisq computers. In High Performance Computing, pages 240–258. Springer Nature Switzerland, 2023.

[117] Cong Peng, Rongmao Chen, Yi Wang, Debiao He, and Xinyi Huang. Parameter-hiding order-revealing encryption without pairings. In IACR International Conference on Public-Key Cryptography, pages 227–256, 2024.

[118] D. J. Persis and T. P. Robert. Ant based multi-objective routing optimization in mobile ad-hoc network. Indian Journal of Science and Technology, 8(9):875, 2015.

[119] Amir Pnueli. The temporal logic of programs. In Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science (FOCS'77), pages 46–57. IEEE, 1977.

[120] Precedence Research. Unmanned aerial vehicle market size, share, and trends 2025 to 2034. https://www.precedenceresearch.com/unmanned-aerial-vehicle-market, 2025. Accessed June 2025.

[121] Ivan Puddu, Moritz Schneider, Daniele Lain, Stefano Boschetto, and Srdjan Čapkun. On (the lack of) code confidentiality in trusted execution environments. In IEEE Symposium on Security and Privacy (SP), pages 4125–4142, 2024.

[122] S. Randriamasy, Y. Lee, J. Seedorf, and R. Yang. ALTO Performance Cost Metrics. RFC 8896, November 2020.

[123] Yakov Rekhter, Tony Li, and Susan Hares. A border gateway protocol 4 (bgp-4). RFC 4271, January 2006. Standards Track.

[124] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, January 2001. Standards Track.

[125] Paul D Rowe. Principles of layered attestation. arXiv preprint arXiv:1603.01244, 2016.

[126] K. Ryu and W. Kim. Multi-objective optimization of energy saving and throughput in heterogeneous networks using deep reinforcement learning. Sensors, 21(23), 2021.

[127] Nazih Salhab, Rana Rahim-Amoud, and Rami Langar. Nfv orchestration platform for 5g over on-the-fly provisioned infrastructure. 04 2019.

[128] V. Sastry, T. Janakiraman, and S. I. Mohideen. New algorithms for multi objective shortest path problem. Opsearch, 40:278–298, 2003.

[129] Fabian Schwarz. TrustedGateway: TEE-Assisted Routing and Firewall Enforcement Using ARM TrustZone. In 25th International Symposium on Research in Attacks, Intrusions and Defenses, RAID '22. ACM, 2022.

[130] Fabian Schwarz and Christian Rossow. 00SEVen – re-enabling virtual machine forensics: Introspecting confidential VMs using privileged in-VM agents. In USENIX Security Symposium (USENIX Security 24). USENIX Association, August 2024.

[131] ChangXiang Shen, HuanGuo Zhang, HuaiMin Wang, Ji Wang, Bo Zhao, Fei Yan, FaJiang Yu, LiQiang Zhang, and MingDi Xu. Research on trusted computing and its development. Science China Information Sciences, 53(3):405–433, 2010.

[132] Kiran K. Somasundaram and John S. Baras. Path optimization and trusted routing in manet: An interplay between ordered semirings. In Natarajan Meghanathan, Brajesh Kumar Kaushik, and Dhinaharan Nagamalai, editors, Advances in Networks and Communications, pages 88–98, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[133] J. Song, Y. Lee, A. G. D. Raymond, and J. Seedorf. Application-Layer Traffic Optimization (ALTO) Protocol. RFC 7285, September 2013.

[134] Syh-Yuan Tan, Tiong-Sik Ng, and Swee-Huay Heng. Efficient fork-free bls multi-signature scheme with incremental signing. In International Conference on Provable Security, pages 250–268, 2024.

[135] The CONNECT Consortium. CONNECT Trust & Risk Assessment and CAD Twinning Framework (InitialVersion) , 2024. Accessed: 2025-11-16.

[136] Jan Tretmans. Test generation with inputs, outputs and quiescence. Software—Concepts and Tools, 17(3):103–120, 1996.

[137] Muhammad Usman, Simone Ferlin, Anna Brunstrom, and Javid Taheri. A survey on observability of distributed edge & container-based microservices. IEEE Access, 10:86904–86919, 2022.

[138] JP Vasseur, Adrian Farrel, and Gerald Ash. A Path Computation Element (PCE)-Based Architecture. RFC 4655, August 2006.

[139] Eric Voit, Henk Birkholz, Thomas Hardjono, Thomas Fossati, and Vincent Scarlata. Attestation Results for Secure Interactions. Internet-Draft draft-ietf-rats-ar4si-09, Internet Engineering Task Force, August 2025. Work in Progress.

[140] C.-X. Wang and et al. On the road to 6g: Visions, requirements, key technologies, and testbeds. IEEE Communications Surveys & Tutorials, 25(2):905–974, 2023.

[141] Lisa Wernet, Sebastian Rust, Silas Gerock, Tobias Meuser, and Björn Scheuermann. Quicup: Secure user plane tunneling for cellular networks. In 2025 IEEE 50th Conference on Local Computer Networks (LCN), page 1–9. IEEE, October 2025.

[142] Qin Wu, Will Liu, and Adrian Farrel. Service models explained. RFC 8309, January 2018.

[143] Kai Xiong, Tao Chen, Zheng Yang, Weizhi Wang, and Ke Li. Intent-driven nfv service orchestration for edge computing. IEEE Transactions on Network and Service Management, 17(4):2200–2214, 2020.

[144] Haoxuan Xu, Jia Xiang, Zhen Huang, Guoxing Chen, Yan Meng, and Haojin Zhu. Latte: Layered attestation for portable enclaved applications. In 2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P), pages 339–354, 2025.

[145] Ganxiang Yang, Chenyang Liu, Zhen Huang, Guoxing Chen, Hongfei Fu, Yuanyuan Zhang, and Haojin Zhu. A formal approach to multi-layered privileges for enclaves. In NDSS, 2025.

[146] S. Yang, L. Zhuang, J. Zhang, J. Lan, and B. Li. A multipolicy deep reinforcement learning approach for multiobjective joint routing and scheduling in deterministic networks. IEEE Internet of Things Journal, 11(10):17402–17418, 2024.

[147] Siqi Zhao, Xuhua Ding, Wen Xu, and Dawu Gu. Seeing Through The Same Lens: Introspecting Guest Address Space At Native Speed. In USENIX Security Symposium (USENIX Security 17). USENIX Association, 2017.

[148] Lei Zhou, Xuhua Ding, and Fengwei Zhang. Smile: Secure memory introspection for live enclave. In 2022 IEEE Symposium on Security and Privacy (SP), pages 386–401, 2022.