CASTOR

Building Trust
in the Computing Continuum

# D4.2
# Trusted Path Establishment Building Blocks, Optimization Engine & Crypto Structures for Trusted Data Sharing

| | |
|---|---|
| **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/ D4.2 / v1.0 |
| **Workpackage:** | WP4 |
| **Due Date:** | 31st March, 2026 |
| **Actual Submission Date:** | 2nd April, 2026 |
| **Responsible Organisation:** | UvA |
| **Editor:** | Anuj Pathania |
| **Dissemination Level:** | Public |
| **Revision:** | 1.0 |
| **Abstract:** | This document presents the first release of the CASTOR framework's core Trust Engineering pipeline for trusted path provisioning. The Trust Assessment Framework utilizes a federated system of agents; preliminary experimentation validates the critical role of Subjective Logic operators and temporal evolution in ensuring accurate trust propagation. To establish Trust Policies that effectively guide these evaluations, the Risk Assessment Engine introduces a topology-aware risk methodology, identifying the core dimensions required to probabilistically model multi-step threat transitions and state uncertainties. Finally, the Optimization Engine translates these dimensions into actionable routing decisions. By addressing a bi-objective optimization problem, it successfully balances network and trust metrics, benchmarking a quantum-inspired heuristic against an exact Dijkstra-based baseline. |




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** | OTHER* | |
| **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**

Anuj Pathania (UvA)

**Contributors (ordered according to beneficiary numbers)**

Nikos Fotos, Thanassis Giannetsos (UBITECH)
Iasonas Sakellariou, Stelios Kazazis, Symeon Tsintzos (QUBITECH)
Alexandros Fakis, Kostas Maliatsos (FERON)
Anuj Pathania, Andy Pimentel (UvA)
Jamie Pont, Budi Arief, Theo Dimitrakos (UKENT)

**Disclaimer**

# Executive Summary

This document, Deliverable D4.2, details the first version of the core building blocks for establishing a Trust Engineering pipeline that supports trust-aware traffic engineering provisioning. It outlines the three foundational pillars of Work Package 4 (WP4): the Trust Assessment Framework (TAF), the Risk Assessment Engine (RAE), and the Optimization Engine (OE).

**Deliverable Objectives**: D4.2 documents the initial implementation and evaluation of mechanisms that allow CASTOR to achieve the following objectives.

- It allows CASTOR to continuously collect and interpret trustworthiness evidence at a device, but also at a domain-level.
- It allows CASTOR to quantify and propagate trust and risk across complex multi-domain topologies.
- It allows CASTOR to compute network paths that are jointly optimal with respect to both network- and trust-metrics.

**Trust Assessment Framework (TAF)**: The Trust Assessment Framework (TAF) is designed as a federated system of agents called TAFs. TAFs work together to provide a topology-wide trust characterisation. The key features of TAF include the following.

- **Federated Trust Assessment:** The federated design includes a centralized Global TAF at the CASTOR Orchestration layer and distributed Local TAF agents deployed across the network topology. The Local TAFs assess local, in-device trust properties, whereas the Global TAF aggregates all local reports in order to form domain-level trust evaluations.
- **Subjective Logic-Based Modelling:** Trust opinions are expressed using subjective logic, which allows for explicit handling of belief, disbelief, and uncertainty trust metrics. The modelling then allows for fusion and discounting of these metrics using well-defined, standard operations.
- **Trust Policy management:** A trust policy encapsulates crucial information that guides the trust assessment process. This includes the modelling of trust relationships, the relevant evidence sources and the Required Trust Level (RTL) constraints that are used during the final Trust Decision process. In the context of the federated modality, the Trust Policy specifies the appropriate operators that allow the aggregation of trust evaluations from various TAF agents.
- **Historical Assessments Handling:** The TAF is operationally required to weigh past trust evaluations on a trust proposition against fresh evidence. To achieve this, it leverages distinct weighted parameters to capture different behaviours when transitioning between states of varying trustworthiness. Specifically, it dictates how historical characterizations influence new data, ensuring that trust scores increase conservatively following positive evidence but drop radically in response to negative events.

**Risk Assessment Engine (RAE)**: The Risk Assessment Engine (RAE) provides the analytical foundation for deriving the Required Trustworthiness Level (RTL) within the CASTOR architecture. By transitioning

from manual, exhaustive attack path analysis to an automated risk profiling framework, the RAE translates abstract security requirements and organizational risk tolerance into concrete, quantifiable trust thresholds. These design-time RTLs serve as the critical baseline against which runtime evidence (the Actual Trustworthiness Level, or ATL) is continuously evaluated. The key features of the CASTOR RAE include the following:

- **Topology-Aware Risk Profiling:** The RAE elevates risk assessment from isolated, device-level vulnerability calculations to comprehensive, topology-wide evaluations. This approach allows the engine to calculate context-specific RTLs by revealing how different assets and potential attack paths cross-impact one another across complex network structures.
- **Probabilistic Threat Modelling:** The engine defines the core dimensions necessary for the probabilistic modelling of attack path realization. It utilizes Monte Carlo simulation to execute probabilistic "what-if" scenarios, enabling the automatic estimation of the likelihood of discovered single- and multi-step attack paths across the topology.
- **Foundation for Cascading Attack Analysis:** The current architecture lays the groundwork for formally modelling cascading attack paths as a Markov Decision Process (MDP). This establishes the basis for systematically capturing the conditional probabilities of sequential exploits and the degree of belief regarding resulting asset states, paving the way for automated, stochastic decision-making in future releases.

The CASTOR RAE seamlessly bridges theoretical risk mitigation with runtime trust evaluation. By allowing the Security Administrator to indicate which security controls reduce the topology's risk posture to acceptable levels, the engine formulates the final Trust Policy. This policy (apart from the RTL thresholds and the trust models) explicitly defines the set of runtime evidence that the TAF must collect throughout the topology's operational lifecycle to accurately monitor the ATL, ensuring that final trust decisions are semantically aligned.

**Optimization Engine (OE)**: The Optimization Engine (OE) is responsible for calculating optimal trusted paths with respect to both network- and trust-related metrics. In the first release of the OE prototype, it treats network and trust as two distinct objectives. The OE formulates this into a bi-objective problem and considers the following two independent approaches to solve the problem.

- **Bi-Objective Dijkstra Algorithm:** The CASTOR OE uses a multi-objective Dijkstra algorithm, a well-known variant of the single-objective Dijkstra algorithm, to enumerate a full set of Pareto-optimal paths in network and trust dimensions. It provides a complete coverage of the Pareto front at a reasonable computational cost on traditional compute infrastructure.
- **Ballistic Simulated Bifurcation (bSB) Algorithm:** The CASTOR OE also explores the use of a stochastic multi-objective quantum-inspired optimization algorithm called bSB on a limited experimental basis.

The OE employs the bi-objective Dijkstra algorithm as the default operational algorithm to solve the trusted path routing problem within the scope of CASTOR. It regards the bSB algorithm as a research method for the same routing problem.

**Future Outlook:** The building blocks specified in this document position WP4 as the technical core of the broader CASTOR Trust Engineering process. The details in this deliverable represent the initial release of the TAF, RAE, and OE components of the CASTOR framework. They provide the conceptual and technical groundwork for trustworthy path establishment in the computing continuum.

# Contents

# List of Figures

# List of Tables

# Versioning and contribution history

| Version | Date | Author | Notes |
|---|---|---|---|
| v0.1 | 26.1.2026 | Nikos Fotos (UBITECH) | Released Table of Contents |
| v0.2 | 2.2.2026 | Nikos Fotos (UBITECH), Jamie Pont (UKENT) | Input on Trust Assessment System Overview (standalone modality) |
| v0.3 | 9.2.2026 | Alexandros Fakis (FERON), Jamie Pont (UKENT) | Input on Trust Assessment Evaluation Plan and initial draft on Risk Assessment System Overview |
| v0.4 | 16.2.2026 | Iasonas Sakellariou, Symeon Tsintzos (QUBITECH) | First full draft of (exact and heuristic) optimization algorithms. Presentation of first set of evaluation results |
| v0.5 | 23.2.2026 | Alexandros Fakis (FERON) | Description of core properties for cascading attack analysis. |
| v0.6 | 2.3.2026 | Alexandros Fakis (FERON), Nikos Fotos (UBITECH), Jamie Pont (UKENT) | Analysis of Trust Assessment results (Chapter 3), Description of Monte Carlo simulation process based on the core properties for calculations on the cascading attack likelihood |
| v0.7 | 9.3.2026 | Alexandros Fakis (FERON), Iasonas Sakellariou, Symeon Tsintzos (QUBITECH) | First full draft of the Risk Assessment Engine (Chapter 4), including the case study on the RTL methodology results. First full draft on the Optimization Engine results including evaluation scenarios based on realistic network topologies. |
| v0.8 | 16.3.2026 | Anuj Pathania (UvA) | Review of Input on Optimization Problem and Evaluation results (Chapter 5). |
| v0.9.1 | 27.3.2026 | ALL | Final version of evaluation sections (Chapter 3 for TAF, Section 4.5 for Risk Assessment, Section 5.4 for Optimization Engine) |
| v0.9.2 | 1.4.2026 | ALL | Final review and revision of editorial comments |
| v1.0 | 2.4.2026 | Daphne Galani (UBITECH) | Final Review & Submission |

# Chapter 1

# Introduction

## 1.1 Scope and Purpose

The scope of this deliverable is to provide the first integrated description, implementation, and initial evaluation of the trusted-path establishment building blocks developed within CASTOR. Specifically, D4.2 documents the three core pillars that enable trusted communication paths across heterogeneous computing-continuum infrastructures. These pillars are, namely, the Trust Assessment Framework (TAF), the Risk Assessment Engine (RAE), and the Optimization Engine (OE). For each component, this document provides a comprehensive internal system overview, defines the implemented inner interfaces, and details a preliminary standalone evaluation. These initial assessments are critical, as they establish baseline performance and will directly guide the deployment of the artifacts in preparation for the first release of the CASTOR integrated framework and its subsequent evaluation against the CASTOR use cases.

As it pertains to the **CASTOR Trust Assessment Framework**, this deliverable details the first release of the federated architecture, specifying the internal design and deployment of distributed Local TAF agents at the router level and the (centralized) Global TAF at the CASTOR Orchestration Layer. It outlines the expected interfaces necessary for handling Trust Policies, quantifying trust opinions from collected trustworthiness evidence, and consolidating trust evaluations to form the final trust decision. The preliminary evaluation validates Subjective Logic as the probabilistic foundation for aggregating multi-agent trust opinions. Furthermore, it assesses how historical trust characterizations influence the impact fresh evidence, specifically ensuring that trust scores increase conservatively following positive evidence but drop radically in response to negative events. These initial findings will allow us to fine-tune the subjective logic operators and decay factors before the TAF is fully integrated into the CASTOR ecosystem to assess topology-wide trust during the operational use cases.

This deliverable presents the first version of the **CASTOR Risk Assessment Engine**, providing a high-level overview of its internal system architecture, the participating subcomponents, and their interactions toward a continuous and dynamic risk assessment framework. The CASTOR RAE serves as a critical common denominator for the two primary foundational aspects of trust assessment: defining the set of requirements in the form of Required Trust Level (RTL) constraints, and identifying the set of runtime evidence that must be collected to measure the Actual Trust Level (ATL). To provide accurate RTL constraints, the CASTOR RAE employs a topology-aware methodology that adjusts final risk calculations per node by capturing the likelihood of cascading attacks across the network. Accordingly, this deliverable provides a first attempt at the probabilistic modelling of the core dimensions that affect the realization of an attack step within a wider cascading scenario. By employing Monte Carlo simulations to sample possible attack paths, the first version of the engine allows for the specification of ”what-if” scenarios that estimate the likelihood of these paths based on the aforementioned dimensions and the risk knowledge that is available thus far. This probabilistic approach constitutes a crucial stepping stone toward the formulation of a Markov Decision Process (MDP) which is able to support the systematic modelling of state

and transition uncertainty to evaluate the trade-off between the likelihood of an attack taking place and the cost of mitigating it via appropriate security measures. Finally, the preliminary evaluation includes a case study showcasing the impact of cascading analysis and security control considerations on the overall RTL methodology.

Building upon the generic multi-objective problem formulation established in D4.1, this deliverable outlines the first prototype of the **CASTOR Optimization Engine**. In this initial release, the prototype focuses on the bi-objective scenario, providing two implementations that treat the network and trust dimensions as distinct optimization objectives. The system overview specifies how the OE interfaces with these metrics to calculate optimal paths, establishing a critical first step toward the co-enforcement of trust and network requirements in traffic engineering provisioning before progressing to more complex, multi-dimensional cases. The preliminary evaluation then provides a comparative analysis of the CASTOR OE's two internal solvers: the exact Bi-Objective Dijkstra algorithm and the heuristic Ballistic Simulated Bifurcation (bSB) algorithm. Assessing the computational cost and Pareto-front coverage of these algorithms will directly inform the fine-tuning of routing policies in the context of the CASTOR integrated framework in the context of the use case evaluations.

## 1.2 Relation to other WPs and Deliverables

This deliverable is part of CASTOR Work Package 4 (WP4). Figure 1.1 illustrates how this deliverable (D4.2) is positioned in relation to other work packages, as well as to past and upcoming deliverables within its own work package. This document focuses on trusted-path establishment mechanisms and their underlying trust, risk, and optimization capabilities. It provides the first integrated implementation and evaluation of the three core components: the Trust Assessment Framework (TAF), the Risk Assessment Engine (RAE), and the Optimization Engine (OE). D4.2 builds directly upon earlier conceptual and architectural work established in other work packages and deliverables.

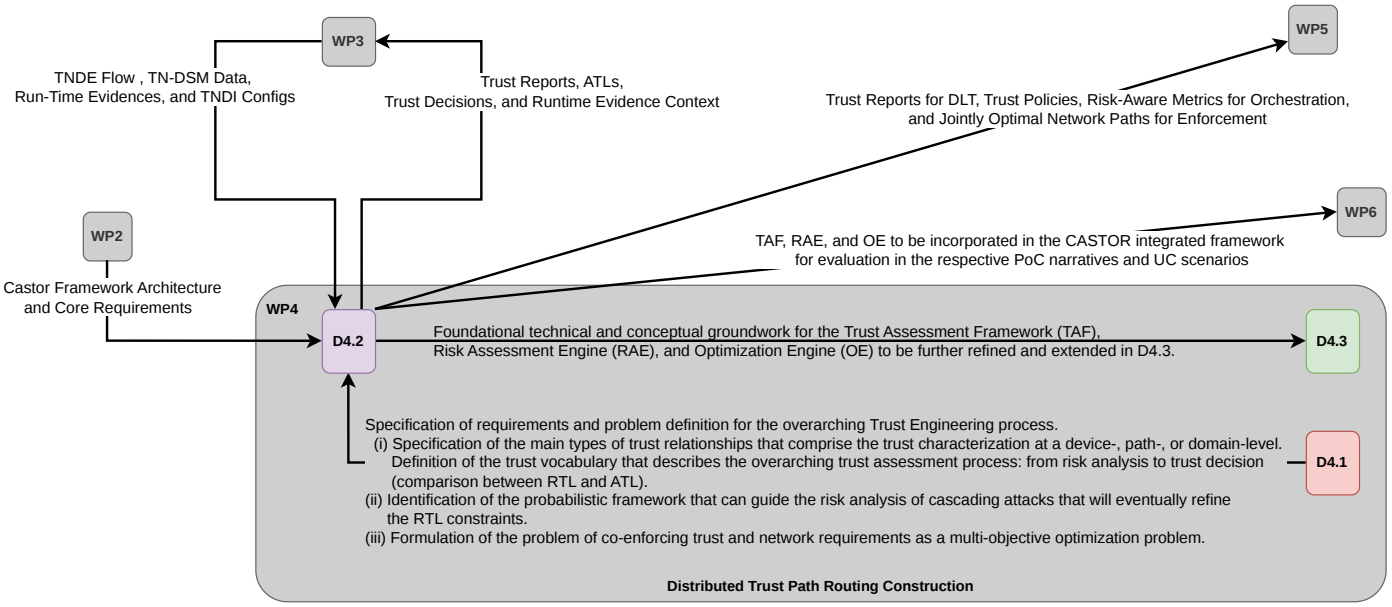



Figure 1.1: Relation of D4.2 with other WPs and Deliverables

WP2 defines the overarching CASTOR architecture and the preparedness phase for trust engineering. It includes the role of the orchestration layer and the forwarding plane. D4.2 concretises these concepts by instantiating the Trust Assessment Framework and its federation of TAF agents, which operate within this architecture to provide topology-wide trust characterisation.

WP3 provides the overall Trust Network Device Extension (TNDE) platform, which consists of the entire in-router architecture for the secure monitoring, collection, and reporting of trustworthiness evidence and claims. This evidence is reported both within the TNDE boundaries (i.e., to the Local TAF agent for the construction of local trustworthiness claims) and beyond (e.g., to the Global TAF through the TNDI-SP protocol defined in WP3). This collected evidence can then be used for improved routing and decision-making.

D4.1 introduced the core terminology and mathematical foundations for the overall Trust Engineering process in CASTOR. Specifically, it established the theoretical baseline for the framework's three core components. For trust modelling, it defined the requirements for the Risk Assessment Engine (RAE) to translate static risks into topology-aware Required Trust Levels. For continuous trust evaluation, it formalized the Trust Assessment Framework mechanisms, detailing the use of Subjective Logic to consolidate runtime evidence. Finally, for trust-aware routing path recommendation, D4.1 formulated the initial multi-objective problem solved by the Optimization Engine to balance competing network and trust metrics. D4.2 builds upon the requirements of D4.1 and significantly advances the objectives of WP4.

D4.3 will serve as the final release of the CASTOR Trust Engineering pipeline, extending the foundational groundwork for the TAF, RAE, and OE laid in D4.2. The scalability analysis and preliminary insights provided in this document will act as a baseline for the large-scale, cross-domain deployment scenarios targeted in D4.3. Moving forward, each core component will be significantly advanced: the TAF will transition from evaluating atomic trust propositions to verifying composite trust propositions across complex path assessments; the RAE will fully integrate cascading attack analysis to systematically derive revised Required Trust Levels (RTLs) and comprehensive Trust Policies; and the OE will elevate its current routing formulation to a higher-dimensional multi-objective problem, supported by further experimentation with diverse algorithmic approaches.

WP5 describes the overarching CASTOR Orchestration layer which focuses on the dynamic enforcement of network- and trust-aware paths in operational network environments. The deliverables in WP5 will rely heavily on the trust and risk metrics, as well as the Pareto-optimal path sets, produced by the TAF, RAE, and OE described in this document.

WP4 provides WP6 with the architectural definition and initial implementations of the Trust Engineering pipeline as part of the overall framework. This connection enables the practical exploitation of trust-aware routing and traffic engineering decisions, ensuring consistency between trust assessment, risk modelling, optimization logic, and domain-level network operations. Specifically, the output of D4.2 (i.e., representing the first version the participating artifacts) will be validated through dedicated Proof-of-Concept narratives (PoC) in deliverable D6.2. Furthermore, these components will serve as foundational elements of the first integrated CASTOR framework, which will be evaluated within the context of the CASTOR Use Cases, also reported in D6.2. The initial definition of these PoC scenarios and the corresponding evaluation planning for the use case scenarios is presented in D6.1.

## 1.3 Deliverable Structure

This deliverable is organized into six chapters. Each chapter address a specific aspect of the CASTOR's trusted-path establishment framework. Together, all chapters provides a full coverage for all components of Work Package 4 (WP4).

- **Chapter 1 - Introduction** defines the scope and objectives of this deliverable. It positions the deliverable within the overall CASTOR workplan and other related deliverables. It describes how this document contributes to the project's goal of establishing trusted communication paths in heterogeneous computing-continuum infrastructures.

- **Chapter 2 – Trust Assessment Framework** presents the Trust Assessment Framework (TAF) within CASTOR. It introduces the overall concept, architecture, and federation of TAF agents. It details the implementation of trust models and the associated Trust Policy. Finally, it describes the core subsystems and internal/external interfaces that enable continuous trust assessment and interaction with the broader CASTOR ecosystem.
- **Chapter 3 – Evaluation of the CASTOR Trust Assessment Framework** describes the evaluation plan and experimental setup for validating the TAF. It focuses on key functionalities such as historical evidence handling, discounting, and trust transitivity. It also reports preliminary evaluation results that demonstrate the correct behaviour of the trust assessment process.
- **Chapter 4 – Risk Assessment Engine** introduces the Risk Assessment Engine (RAE) within CASTOR. It outlines the role of risk analysis within the overall trust engineering process. It also describes the RAE architecture and its main components. Finally, it explains how vulnerabilities, threats, and attack paths are quantitatively modelled to derive risk-aware trust requirements.
- **Chapter 5 – Optimization Engine** presents the Optimization Engine within CASTOR. It formulates trusted-path routing as a multi-objective optimisation problem. The problem jointly considers both network- and trust-related metrics. Thereafter, it introduces both exact classical and heuristic quantum-inspired optimisation techniques. Finally, it reports initial evaluation results on synthetic and real network topologies.
- **Chapter 6 – Summary and Conclusions** concludes the deliverable by summarizing the main outcomes of the deliverable.

# Chapter 2

# Trust Assessment Framework

## 2.1 Overview

The overall CASTOR framework requires a robust and highly modular trust assessment framework for processing the collected trustworthiness evidence from the network elements and deriving continuous, dynamic trust evaluations. These evaluations must accurately characterize the capabilities of the overall forwarding plane as well as the assurance guarantees for the deployed services. In Deliverable D2.1 [8], we presented the overarching problem of the trust engineering process in the context of modern traffic engineering applications. Subsequently, Deliverable D4.1 [6] analyzed the core terminology and the foundational mathematical framework, namely Subjective Logic, capable of tackling the trust assessment requirements in such complex multi-agent systems. Eventually, this culminated in the identification of explicit engineering stories that specify the main functional requirements the framework must satisfy.

The intrinsic challenges of deploying a standalone trust assessment framework are not identified and solved from scratch in CASTOR. Instead, we draw upon experience obtained in the context of previous works [23], building on top of the established, generic principles of the standalone TAF to manifest a highly modular, federated architecture. Driven by this foundation, the core innovation of the CASTOR Trust Assessment Framework (TAF) is the employment of a federation of TAF agents to provide topology-wide trust characterization at the Orchestration Layer (see D2.1 for a high-level overview of the overarching CASTOR framework). This federated approach enables the elevation of atomic, device-level trust evaluations into highly complex trust propositions. Consequently, the framework can seamlessly aggregate local data to characterize the overall trust posture of an individual device, a derived network path, or even an entire administrative domain.

Following these requirements and innovations, this chapter introduces the first release of the CASTOR TAF. The upcoming sections detail the core system overview of a TAF instance capable of carrying out the complete trust engineering lifecycle across all envisioned modalities presented in D4.1 [6] - primarily the Federated Trust Assessment Framework. In addition, we provide a comprehensive walkthrough of the implementation details of a trust model which constitutes one of the core pillars of the Trust Policy that characterizes all aspects of the trust engineering process. As the primary input for a TAF instance, the Trust Policy dictates the end-to-end trust assessment: from the initial trust model representation and evidence collection, through to the quantification of the final ATL value and the derivation of the corresponding trust decision. Finally, the chapter analyzes the set of internal and external interfaces that allow a TAF instance to function and interact seamlessly with the broader CASTOR ecosystem.

## 2.2 Architecture

Internally, the TAF is structured as a modular system comprising several distinct components, as depicted in the high-level architecture in Figure 2.1. The *Trust Model Manager* (TMM) serves as an internal repository for available trust models, instantiating the appropriate model instances upon application request and managing their complete lifecycle by incorporating any necessary modifications. The *Trust Source Manager* (TSM) maintains a catalog of available trust sources utilized as evidence for assessing an entity's trustworthiness. By continuously evaluating evidence from these external sources, the TSM dynamically integrates new trust opinions into the active trust model instances. The specification of the respective interfaces (i.e., NDI, DLT, TAS, TAQI, ECI) is detailed in Section 2.5.

Following this, the *Trustworthiness Level Expression Engine* (TLEE) receives a trust model instance as input and evaluates the trustworthiness level of its underlying propositions. The *Trust Decision Engine* (TDE) then utilizes the output from the TLEE to formulate the final trustworthiness and trust decisions. Overseeing these processes is the *Trust Assessment Manager* (TAM), a central orchestration component that coordinates updates from the TMM and TSM, schedules TLEE computations, and triggers TDE invocations. The TAM also provides the external interfaces required for applications to access the TAF's trust assessment services. Finally, the *Trust Policy Language (TPL) Data Connector* acts as the ingestion point for the system's rules; it is responsible for interpreting the Trust Policy information and forwarding it to the main logic of the TAM to initiate the necessary assessment actions.

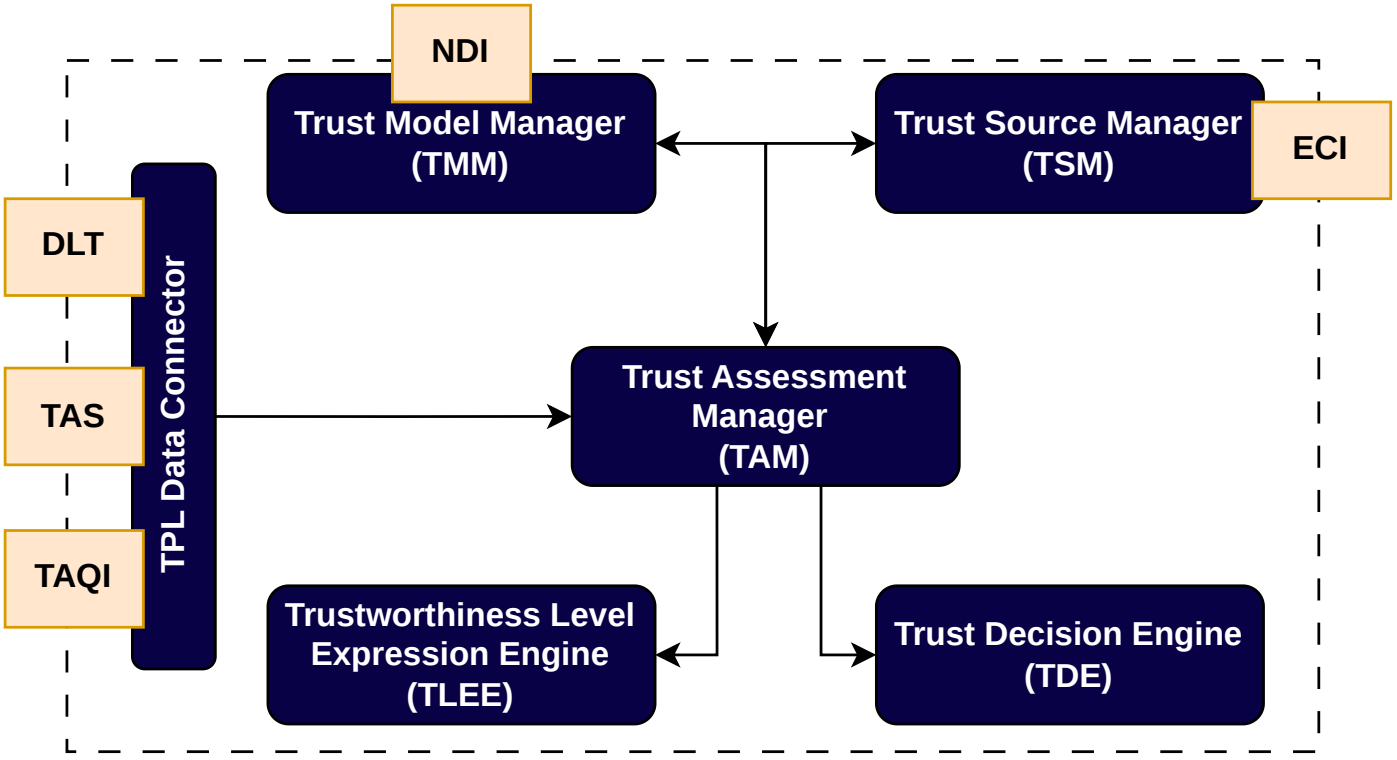



Figure 2.1: High-level architecture of the Trust Assessment Framework (TAF)

The foundational architecture of the TAF follows a concurrent, event-driven paradigm where multiple components continuously await, process, handle, and emit events. Based on the various external services, interfaces, and internal tasks the TAF must integrate, we can conceptually categorize these components into the following groups:

1. Components that process updates from outside the TAF and publish them as internal events (e.g., when TAF agents cooperate in a federated deployment).

2. Components that process internal updates by triggering changes to internal representations (e.g., the Trust Model Manager).
3. Components that monitor changes to internal representations and schedule downstream computations (e.g., the Trust Assessment Manager).
4. Components dedicated to executing specific computations upon invocation (e.g., the TLEE).
5. Components that provide services to external entities, enabling access to internal representations and computational results (e.g., the Trust Assessment Manager).

This functional separation largely corresponds to a unidirectional execution flow. External updates (e.g., opinions arriving at the Global TAF from connected Local TAF agents) are captured and subsequently reflected in internal representations, such as trust model instances. These internal representations then serve as inputs for computations (via the TLEE) and result storage (e.g., caching recent ATLs). Orthogonally, the results of these recent computations are utilized to provide application services, such as responding to Trust Assessment Requests (TARs) or sending notifications. This strict separation of concerns among components enables the system to achieve several critical design goals:

- **Concurrent Execution Model** To ensure scalability and performance under heavy load, the TAF heavily utilizes multi-threading to take full advantage of modern CPU architectures. This is achieved through dedicated event processors and, depending on data consistency requirements, thread pools that handle events concurrently. Rather than relying on external message queues or brokers—which introduce network latency, unrealistic messaging abstractions, and unnecessary thread overhead—the TAF uses highly efficient language-internal mechanisms (i.e., shared memory) for event dissemination.

  The system conceptually acts as a highly concurrent in-memory database maintaining numerous trust model instances with high update rates, and so mitigating write contention is critical. To address this, the architecture minimizes shared state and strictly applies the Single Writer Principle. Under this principle, only a single execution unit is permitted to modify a specific data structure, while all others retain read-only access. This eliminates write contention and aligns seamlessly with the caching optimizations of modern CPUs. Furthermore, the TAM and TLEE utilize strict data ownership models: the TAM generates an immutable copy of a trust model instance's latest state and passes it to the TLEE. The TLEE performs its computations on this isolated copy and returns a strictly immutable result, ensuring thread safety throughout the assessment pipeline.

- **Decoupled Architecture** The architecture enforces loose coupling across the majority of its components, with tighter coupling (i.e., direct function calls) reserved exclusively for interactions between the TAM and the TLEE/TDE to schedule computationally expensive operations. This generalized loose coupling provides two distinct advantages. First, given that components interact primarily by processing and emitting events, they operate without needing deep knowledge of each other's internal mechanics. They simply react to defined event types. This allows new events to be introduced or component implementations to be swapped transparently, provided the necessary event handlers are in place.

  Second, separating components into independent event processors backed by threads facilitates "bulk-heading" during load peaks. In systems utilizing back-pressure, an overloaded component throttles upstream processes, eventually stalling the entire system's ability to process new external messages. Since the TAF cannot control the volume of external messages, it relies instead on load-shedding; that is, prioritizing newer, more relevant updates over older ones to maintain reasonable latency. Bulk-heading isolates these components, ensuring that an overloaded module (e.g., a surging Trust Source Manager) does not impede the TAM's ability to invoke the TLEE or halt critical trust computations.

- **Extendable Framework** Components that interface with external systems (such as the TAF Federation Listener or the CASTOR DLT) are designed to be easily modularized and swapped. This extendability not only simplifies development and testing through the use of mock modules, but also allows for the seamless integration of specific adapters when running complex emulations or workload generators.

## 2.3 Trust Model Implementation

A trust model captures the specific set of trust relationships and trust opinions required to execute a dedicated trust evaluation. Ultimately, it encapsulates all the necessary information to calculate an aggregated, consolidated view of trustworthiness from the perspective of a single assessor or agent.

In its simplest form, a trust model could be implemented purely through a symbolic representation of various opinions alongside the mathematical operations (e.g., Subjective Logic) required to derive the final ATL value. However, an enriched alternative would be to provide a structured approach and implement the trust model as a graph. As we will see below, this allows the TLEE to automate the process of filtering the appropriate trust relationships for an evaluation and selecting the corresponding Subjective Logic operators for aggregating all opinions into a single ATL value for a given evaluation (i.e., trust proposition).

This structured approach aligns with our initial conceptualization in Deliverable D4.1 [6], where we defined a trust model as follows: “A trust model is a graph-based model which represents all components and data needed to perform a certain function. It consists of trust objects and directional trust relationships between trust objects (i.e., the trust network). It also stores trust sources used to quantify trust relationships”. Although this definition provides a solid baseline, it remains relatively generic and broad. In the following sections, we extend and refine this definition to focus explicitly on the model’s operational functionality. This refined perspective allows us to better capture the specific implementation details and dynamic requirements of trust models within the CASTOR framework.

At its core, a Trust Model is a directed acyclic graph data structure whose nodes we refer to as **Trust Objects**. A visual concept of a Trust Model - as discussed in D4.1 [6] - is shown in Figure 2.2.

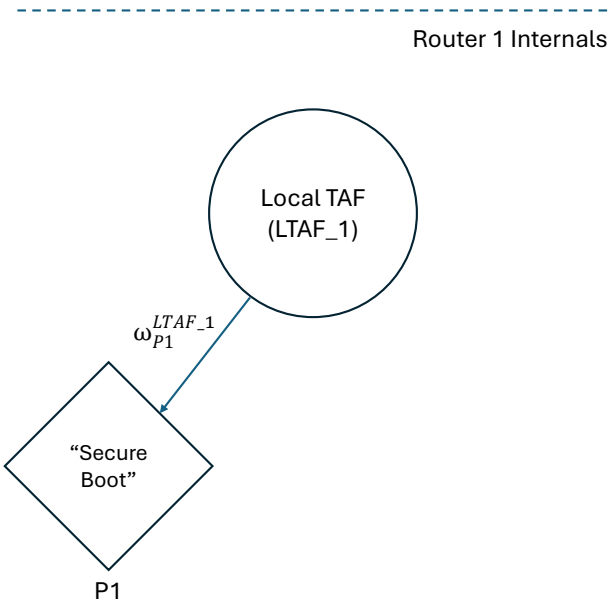



Figure 2.2: A visual concept of a Trust Model.

The Trust Model operates as a core component of the overarching Trust Policy, which dictates the complete trust evaluation process. Structurally, the Trust Model is represented as a directed graph. It features a single node with no incoming edges, designated as the root node. This root node represents the entity conducting the assessment, referred to as the Agent (i.e., Trust Object represented as circle) in Figure 2.2). Conversely, the model contains at least one leaf node representing the entity whose trustworthiness is being assessed, referred to as the Proposition (depicted as the grey Trust Objects in Figure 2.2). Depending on the complexity of the evaluation, intermediary nodes (white Trust Objects) may also exist between the root and the leaf nodes.

The directed edges shown in Figure 2.2 represent the Trust Relationships between these trust objects. In this context, the source node acts as the Trustor and the target node acts as the Trustee. As established in Deliverable D4.1 [6], a Trust Relationship is mathematically expressed as a subjective logic binomial opinion, $\omega_{Trustee}^{Trustor}$.

The overarching Trust Model constitutes an integral part of the Trust Policy that is enforced in a TAF instance. In addition to the Trust Model, a Trust Policy specifies the types of Trust Sources that need to invoked in order to continuously update the Trust Model state. The Trust Sources represent the specific external entities from which the Trust Assessment Framework must collect evidence to assess and quantify a given Trust Relationship. Finally, the Trust Policy contains the additional algorithmic information and rules required to process this collected evidence. Specifically, it dictates how to quantify a set of evidence from a single Trust Source into a Trust Opinion, and how to aggregate multiple Trust Opinions to derive the final Actual Trustworthiness Level (ATL).

### 2.3.1 Trust Model Templates and Instances

To be able to distinguish between a design-time trust model and a run-time trust model, we defined the following two manifestations of trust models – templates and instances:

**Trust Model Template (TMT)** is a design-time manifestation of a trust model. A Trust Model Template is created at design-time with the goal of capturing all of information necessary for the Trust Assessment Framework to assess trustworthiness for a specific application upon the receipt of a Trustworthiness Assessment Request. A Trust Model Template specifies all of the relevant Trust Objects, Trust Model Instantiation Policies, Trust Relationships, Trust Sources, and Trust Methods which are to be instantiated as part of a Trust Model Instance. As such, a uniquely identifiable, application-specific Trust Model Template is used by the Trust Model Manager to instantiate a corresponding Trust Model Instance at run-time, making a TMI completely dependent on the design of the TMT.

**Trust Model Instance (TMI)** is a run-time manifestation of a trust model. A Trust Model Instance reflects the TAF's current and latest view on the world and is thus a single, internal source of truth for any computations to be done during trust assessment. The TAF uses the information inside a Trust Model Instance to know which Trust Sources to instantiate, which Trust Relationships need to be quantified in form of a Trust Opinion, and which Trust Opinions form an ATL. However, a TMI is also designed to serve as a data structure to store Atomic Trust Opinions, Trust Opinions, and Actual Trustworthiness Levels during the lifetime of an instance. As such, a TMI will be continually updated with recalculated opinions and levels.

Trust model templates can specify varying levels of dynamicity regarding the topology used by the model at runtime. Static trust model instances have a fixed topology that is already fully determined at design time in their template. In turn, there are also trust model instances whose graph structure can change over time based on the policies inside their corresponding template **and** external input to the TAF. We refer to the prior as **Static Trust Model Instances** and the latter as **Dynamic Trust Model Instances**.

#### 2.3.1.1 Static Trust Models Instances

The structure of Static Trust Model Instances are solely determined by their Trust Model Templates. The graph structure of the static Trust Model Instance should stay the same throughout the lifetime of the instance and it is not supposed to change based on any input received during run-time. In the case of the in-router trust evaluations, the Local TAF agent may be aware of the capabilities of the corresponding TNDE - i.e., its tracing, processing and reporting mechanisms. Hence, the overal Trust Policy that dictates the trust assessments that need to be conducted may rely on a static trust model that can be pre-configured in the Local TAF agent's TMM in advance. An example of such a trust model is shown in Figure 2.2.

In this base scenario, the network operator configures the Local TAF agent of Router 1 to assess its own integrity. The exact aspects evaluated are intrinsically tied to the evidence available within the target environment and the specific requirements (e.g., "high integrity"). As illustrated in Figure 2.2, this static model focuses on evaluating a single atomic trust proposition (P1): "Secure boot in Router 1 stands." Within the current scope of the CASTOR project, all trust propositions are strictly Boolean, evaluating only to True or False. To execute this static model, the Local TAF calculates its Actual Trust Level (ATL) regarding its integrity using direct evidence from internal trust sources (specifically, secure boot attestation).

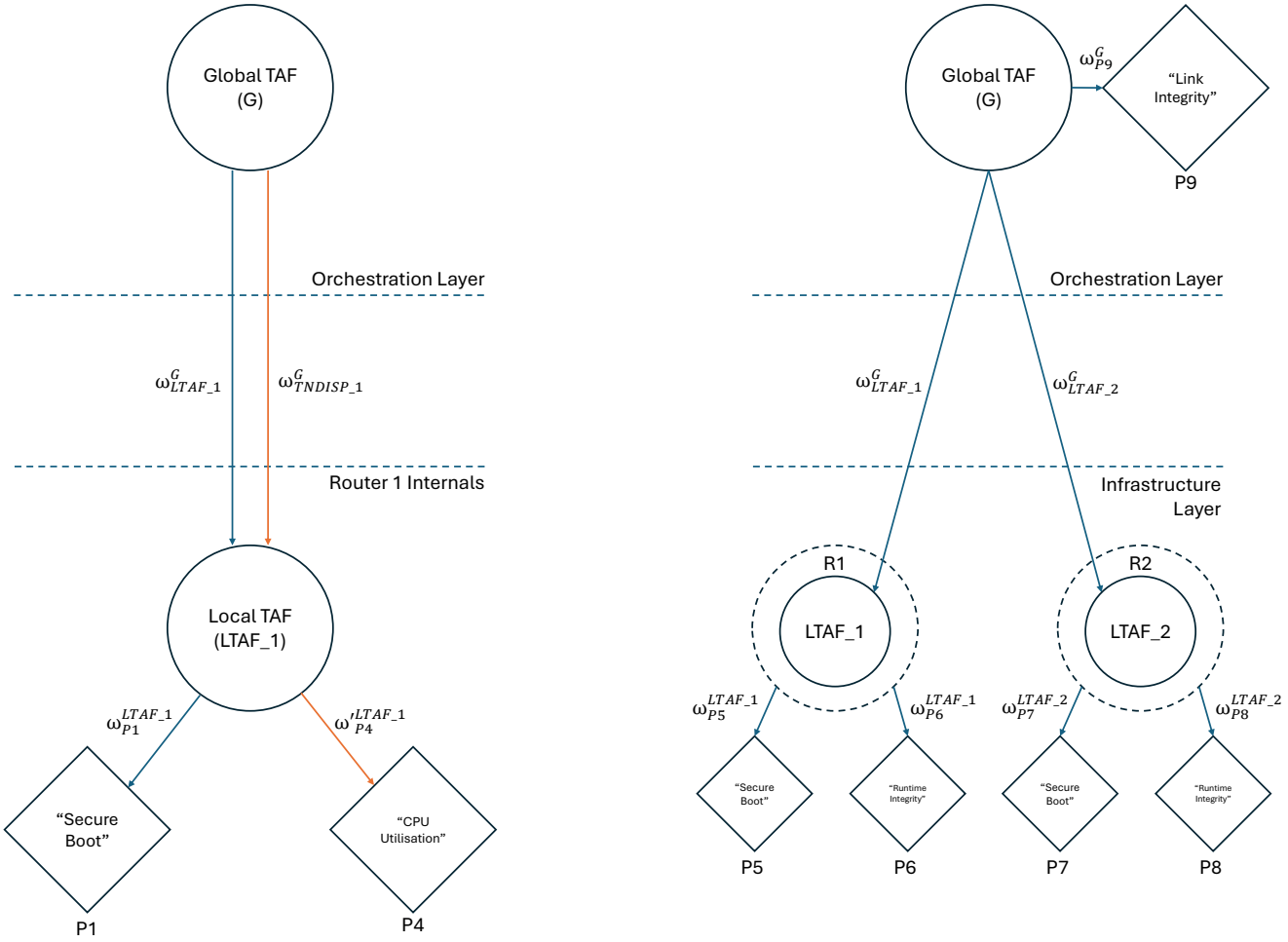
Note that for applications which are represented by Static Trust Models, a single Trust Model Instance is needed to assess the trustworthiness of all relevant trust propositions.

#### 2.3.1.2 Dynamic Trust Models Instances

Dynamic Trust Model Instances (TMIs) are instantiated using their Trust Model Templates (TMTs) and external input received by the TAF instance. Unlike in the case of Static TMIs, there can be multiple Dynamic TMIs per single TMT. In the case of Global TAF trust evaluations, dynamic TMIs can be managed (i.e., created, updated, or removed) according to the lifecycle of the onboarded vRouter topology. For instance, a dynamic TMI is created when a new vRouter gets onboarded. Figure 2.3a illustrates a single dynamic TMI that is created to capture the Global TAF evaluations pertaining to the trustworthiness of vRouter 1. As more vRouters are onboarded into the topology, additional dynamic TMIs are spawned, as shown in Figure 2.3b. In its simplest form, a dynamic TMI expresses device-level trust evaluations per vRouter (i.e., TNDI). This allows the Orchestration Layer presented in D5.1 [7] to evaluate composite trust propositions at the device level by specifying logical expressions that combine the atomic trust evaluations produced by the respective Local TAF agent. For example, by inspecting the Dynamic TMIs in Figure 2.3b, a path profile that characterizes a deployed application service may dictate that the device-level trust characterization for vRouter 1 requires the conjunction of atomic trust propositions $P5$ and $P6$.

However, what happens when a path profile requires the evaluation of more complex trust propositions that go beyond device characterization (e.g., how could the Global TAF evaluate the link-level trust proposition $P9$ that involves the two adjacent routers R1 and R2 in Figure 2.3b)? The previously presented dynamic TMIs do not model the broader scope necessary for deriving trustworthiness claims at a link or path level. This requires introducing a new type of dynamic TMI. Instead of spawning one TMI per vRouter instance, we consider a single, consolidated TMI that is dynamically updated with additional trust objects as the vRouter topology evolves. This allows the Global TAF to maintain a holistic overview of all trust relationships that characterize a link (or a path), thereby enabling the specification of composite trust propositions that expand beyond the boundaries of a single vRouter. An example of such a trust model instance is discussed in the evaluation of Section 3.4.2. The dynamic addition and deletion of Trust Objects within a single TMI can result in frequent structural changes over time. Hence, it is crucial to distinguish between different TMIs at different points in time. Moreover, we anticipate that even the TMTs may need to be updated, albeit significantly less frequently than the TMIs. To address this, we propose a Trust Model Versioning scheme, as explained in the following subsection.

Overall, the first version of the TAF instance's core logic supports both static and dynamic TMI management. The evaluation of the derived ATL values' behaviour is detailed in Chapter 3. Specifically, in this initial round of evaluations, the static TMI is evaluated in the context of in-router trust evaluations within the Local TAF agent, while the variations among the different types of dynamic TMIs are examined in the context of Global TAF evaluations.




(a) An example of a Dynamic Trust Model Instance with one router

(b) An example of a Dynamic Trust Model Instance with a second router added lately

Figure 2.3: This is the main caption for the entire group of three subfigures.

## 2.4 TAF Core Subsystems and Internal Interfaces

This section outlines the internal architecture of the TAF, detailing its core subsystems and their intra-component interfaces. To enforce strict separation of concerns, these communication pathways are fully encapsulated within the TAF boundary and are inaccessible to external consumers. They are detailed here to provide a comprehensive architectural blueprint and clarify the internal operational mechanics of the prototype.

Table 2.1: List of internal and external interface exposure in a TAF instance

| TAF Interface | TAM | TMM | TSM | TLEE | TDE |
|---|---|---|---|---|---|
| *External Interfaces* | | | | | |
| Trust Assessment Service | ✓ | | | | |
| Distributed Ledger Technology Interface | ✓ | | | | |
| Trust Assessment Query Interface | ✓ | | | | |
| Evidence Collection Interface | | | ✓ | | |
| Node Discovery Interface | | ✓ | ✓ | | |
| *Internal Interfaces* | | | | | |
| Trust Model Instance API | ✓ | ✓ | ✓ | | |
| TLEE Invocation | ✓ | | | ✓ | |
| TDE Invocation | ✓ | | | | ✓ |

### 2.4.1 Trust Assessment Manager

The Trust Assessment Manager (TAM) serves as the central orchestration component within a TAF instance (i.e., Global TAF or Local TAF agent). Through its interfaces that form the Trust Policy Language (TPL) Data Connector capabilities of the TAM, it is possible to process any CASTOR Trust Policy and provision the necessary sessions so as to enable the continuous and dynamic trust assessment of the requested trust propositions. Its primary responsibilities include providing trust assessment service functionalities - such as application session management — directly to external applications. Additionally, the TAM processes and incorporates changes to trust model instances whenever indicated by the Trust Model Manager or the Trust Source Manager, while actively scheduling computations for both the Trust Level Expression Engine (TLEE) and the Trust Decision Engine (TDE).

By taking on these responsibilities, the TAM effectively decouples and compartmentalizes several independent functions of the TAF. Specifically, it enforces a strict architectural separation between the following processes:

- **Event Observation vs. Model Updates**: Separating the detection of external network changes from the actual updating of internal trust model instances.
- **Model Updates vs. Trust Computations**: Decoupling the updates made to internal representations from the recalculation of ATLs (via the TLEE) and final trust decisions (via the TDE).
- **State Management vs. External Requests**: Isolating the engine's internal state management from the processing of external Trust Assessment Services (TAS) requests.

This compartmentalization allows the TAM to intelligently schedule TLEE and TDE executions rather than reacting synchronously to every trigger. When determining execution schedules, the TAM evaluates multiple operational factors within a single cohesive workflow. It considers the volume of pending change events, the accumulation of model updates not yet reflected in previous calculations, and the staleness of cached results. It also factors in the immediate demand for updated decisions from active sessions or non-cached Trust Assessment Requests (TARs), alongside the current volume of concurrent executions across other trust model instances.

Furthermore, this decoupled design enables the TAM to apply selective load-shedding during bursts of heavy external updates. Ultimately, this architecture provides the high level of flexibility required to configure and scale the TAF across diverse load scenarios and runtime environments, easily accommodating varying degrees of parallelism, caching, and load management.

#### 2.4.1.1 Triggering Trust Model Updates

The TAF provides an internal, event-based API in which the Trust Model Manager and the Trust Source Manager components can emit events about updates in their observations. These changes are then processed by the TAM in order to update the trust model instances. This asynchronous API approach decouples the components and follows the single writer principle in a way that only the TAM has write access on all trust model instances. At the same time, this prevents write contention and concurrent writes between the trust model manager and the trust source manager. Instead, the trust assessment manager is expected to handle updates for the same trust model instance sequentially, although updates for different trust model instances can be executed in parallel.

### 2.4.2 Trust Model Manager

The Trust Model Manager is the TAF component responsible for overseeing both Trust Model Templates and Trust Model Instances. As introduced in Section 2.3, Trust Model Templates act as application-specific blueprints created at design-time. Each template is mapped to a specific network function or application and explicitly defines all the structural elements required for instantiation, including Trust Objects, Trust Model Instantiation Policies, Trust Relationships, Trust Sources, and Trust Methods. The TAF utilizes these blueprints to provision the necessary sources and map the required trust relationships, establishing the foundation for computing the Actual Trustworthiness Level and reaching a final trust decision. At run-time, the Trust Model Manager generates active Trust Model Instances based on these templates. These live instances act as stateful representations of the trust assessment, storing Trust Opinions and Actual Trustworthiness Levels which are dynamically evaluated during operation.

To maintain these design-time blueprints, the Trust Model Manager utilizes a dedicated Trust Model Template Database (TMT-DB). Within this database, each template is assigned a universally unique Trust Model Template ID (TMT-ID). These distinct identifiers are primarily defined during the CASTOR preparedness phase. By leveraging these TMT-IDs, the Orchestration Layer can unambiguously reference and trigger the exact trust evaluations required at both the router and orchestration levels.

In the process of initializing a session with the TAF, an application (e.g., the TN-DSM in the case of a Local TAF agent) sends the TMT-ID as part of the **initialization message**. The Trust Assessment Manager then forwards the TMT-ID to the Trust Model Manager which then either:

1. locates the corresponding Trust Model Template in the local TMT repository or, if not,
2. queries the Distributed Ledger Technology (DLT) where additional Trust Model Templates are for that specific TMT-ID.

If the specific TMT-ID is not found either in the local TMT-DB or in the DLT, then the Trust Model Manager returns a null value to the Trust Assessment Manager.

If, however, a corresponding Trust Model Template is found, then the Trust Model Manager will either:

1. create a Trust Model Instance based on this template and return a pointer to the instance to the Trust Assessment Manager, or
2. return a pointer to the Trust Model Template which has the corresponding TMT-ID to the Trust Assessment Manager.

The Trust Model Manager will only be able to instantiate *static* Trust Model Instances during the initialization phase. This is because the structure of a static Trust Model Instance is based only on the fixed

structure inside its Trust Model Template. The Trust Model Manager will not be able to instantiate *dynamic* Trust Model Instances during the initialization phase because these types of instances are also dependent on the relevant received messages which the TAF has not yet started listening to during the initialization phase.

During the run-time phase, an application submits a Trustworthiness Assessment Request (TAR) to the TAF. This request includes a TMT-ID, explicitly dictating the Trust Model Template required for the assessment. Upon receiving the TAR, the Trust Assessment Manager (TAM) evaluates the requested TMT-ID against currently active sessions to determine the next step. If a corresponding Trust Model Instance already exists and its Actual Trustworthiness Levels have been previously assessed and stored, the TAM immediately returns these cached results to the requesting application. However, if no Trust Model Instance exists for that TMT-ID, the TAM forwards the identifier to the Trust Model Manager to initiate the instantiation of a new, active Trust Model Instance.

In general, the Trust Model Manager is responsible for:

1. creating appropriate Trust Model Instances during run-time based on the input it receives from the Trust Assessment Manager,
2. adding new trust objects to already existing Trust Model Instances based on appropriate input,
3. deleting expired trust objects,
4. deleting Trust Model Instances when a network element for which the instance was instantiated is no longer sending messages after a certain period of time, and
5. managing and updating the Trust Model Template Database.

In a standalone mode, the Trust Model Manager encapsulates a local TAF instance's perception, which is based solely on the trustworthiness evidence it directly observes. However, transitioning to a federated TAF modality—one of the core innovations of the CASTOR Trust Assessment Framework—significantly expands this scope. To support this federation, the Trust Model Manager has been enhanced to broaden its perception by incorporating trust evaluations reported by external TAF agents. This architectural upgrade enables the Global TAF at the Orchestration Layer to collect and aggregate trust reports generated at the TNDE level by Local TAF agents.

Eventually, this allows the Global TAF at the Orchestration Layer to collect trust reports that are produced at a TNDE level by Local TAF agents and aggregate them in order to derive collective device-level trust evaluations or even provide more complex trust propositions that characterize entire paths or domains. To support this federated approach, the Trust Model Manager introduces a Federation Listener component. This element actively consumes Trust Reports via the Node Discovery Interface, seamlessly enriching the overall trust perception of the TAF instance. The technical evaluation of this initial push-based federation mechanism is detailed in Chapter 3.

### 2.4.3 Trust Source Manager

The Trust Source Manager (TSM) is a component of the TAF responsible for processing the collected trustworthiness evidence from the available Trust Sources. Once a Trust Policy is enforced in a TAF instance, the respective Trust Model Template specifies the Trust Sources that the TSM needs to interact with in order for the overall TAF instance to form its final ATL opinion.

In principle, the TSM spawns a dedicated **Trust Source Quantifier Instance (TSQ-I)** per Trust Source. A TSQ-I is created at run-time based on the corresponding Trust Source Quantifier Template (TSQ-T)

which is created at design time. The TSQ-I receives evidence from the corresponding Trust Source and calculates an atomic trust opinion based on this evidence.

When integrating a new Trust Source Quantifier, the corresponding TSQ-T describes how the evidence provided by that Trust Source should be interpreted. In addition, in the TSQ-T the corresponding formulas and calculation rules for calculating the belief, disbelief and uncertainty of the atomic trust opinion are defined.

There are two options for where the TAF obtains the TSQ-T:

1. The TSQ-Ts could be stored in the TAF. They are created during design time and are installed together with a TAF instance when the corresponding node is set up. Updates of the TSQ-Ts would be possible by an Over-The-Air update of the software of the TAF.
2. The TSQ-Ts could be stored as part of the overall Trust Policy on the CASTOR DLT. If a TSQ-T is required for the calculation of an atomic trust opinion and this template is not stored locally in the TAF, the TAF would request the Trust Policy from the CASTOR DLT and then extract and store the corresponding TSQ-Ts in its local memory. Details on the interaction between the CASTOR DLT and any TAF instance is described in D5.2 [11].

As highlighted in D4.1 [6], a primary challenge in the trust engineering process is breaking down target trust propositions to an atomic level, ensuring that each trust opinion can be quantified by a single type of evidence. Under this approach, applications can leverage logical expressions to query complex trust propositions that encompass multiple facets of a trustee object.

Alternatively, as demonstrated in the standalone TAF prototype [23], it is often necessary to aggregate multiple “atomic” opinions to derive a final trust opinion for a specific trust relationship. From a risk analysis perspective, this implies that assessing trustworthiness for a given proposition requires consuming multiple types of evidence, potentially gathered from various distinct Trust Sources.

Delineating exactly when one strategy is preferable over the other remains an open, domain-specific research problem that must be fine-tuned within the corresponding Trust Policy. This need for flexibility also extends to the evidence quantification process. While a common quantification approach relies on standard Jøsang equations—which form the basis of our evaluation in Section 3.2.1 [6]—the Trust Source Manager (TSM) is designed to be highly modular. It can seamlessly accommodate arbitrary evidence quantification functions to support the intrinsic characteristics of any Trust Source or specific Trust Policy requirements.

As already described in the previous section, an application sends the TMT-ID as part of the initialization message when a session is initialised between the application and the TAF. At this initialization phase, the TSM will process through the trust model instance and extract the corresponding TSQ-T IDs for each trust relationship. In addition to the TSQ-T IDs, further metadata is also provided in the trust model for each trust source. Based on this metadata, a TSQ-I is created out of the TSQ-T. This metadata contains information for the calculation of the trust opinions. These weights are used as an input for the calculation rules and formulas described in the TSQ-T. In addition to the metadata necessary for the calculation rules and formulas, the trust model instance also includes information from where the TSM should request the evidence. Based on the TSQ-T ID and the corresponding metadata provided in the trust model instance, the TSM creates an instance based on TSQ-T, which is the TSQ-I.

As the TSM manages the TSQ-Is, it is aware of which instance needs evidence from which trust source (see Figure 2.4). Therefore, the TSM handles the communication between the TAF and the trust sources. It either requests evidence or subscribes to the Trust Source so that it receives evidence as soon as the evidence has changed. The TSM also manages received evidence and forwards it to the corresponding TSQ-I. For this purpose, a unique identifier is necessary in each received evidence and which can be used by the TSM to map the evidence to the corresponding TSQ-I.

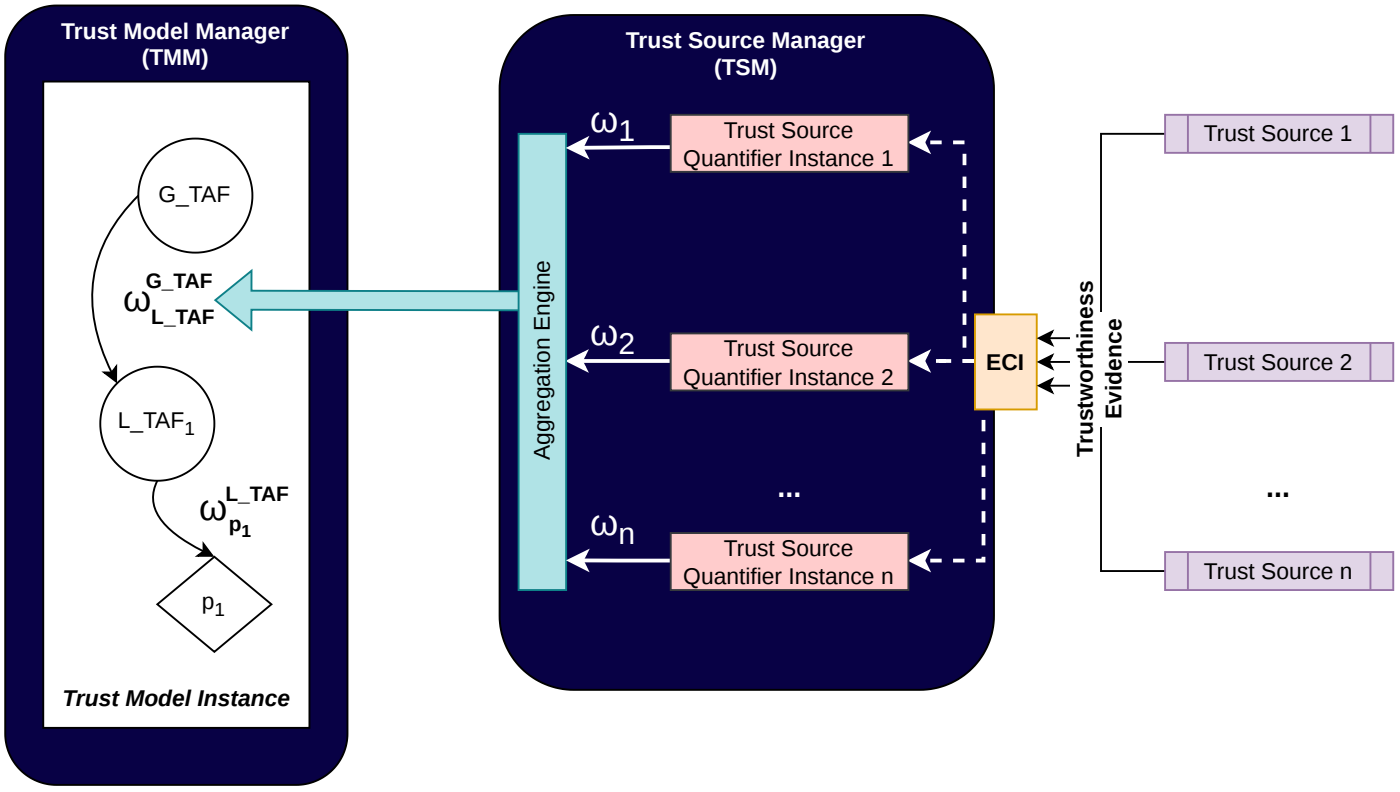



Figure 2.4: High level overview of the trust source manager handling receiving evidence and calculating a trust opinion for a single trust relationship.

Based on the received evidence, each TSQ-I creates an atomic trust opinion. This is a subjective trust opinion calculated by a TSQ-I that has not been discounted or fused with trust opinions provided by other TSQ-Is. For one trust relationship in the trust model instance, several trust sources can be specified. The TSM is aware of which trust sources should be used for each trust relationship. After the corresponding TSQ-Is have calculated the atomic trust opinions for the specified trust sources of a trust relationship, the TSM aggregates the atomic trust opinions to a trust opinion which is assigned to the corresponding trust relationship. The aggregation of the trust opinions is user-specific and is specified as part of the overarching Trust Policy that characterizes the trust engineering process. An overview of the described approach is shown in Figure 2.4.

### 2.4.4 Trustworthiness Level Expression Engine

As explained in the previous section, the TSM is responsible for creating, or quantifying, Trust Opinions on the level of Trust Relationships (TO-TR). As defined in D4.1[6], the Trust Model combines various trust relationships among different trust objects. In response, the TLEE is responsible for assessing the trustworthiness on the level of the trust model (i.e., the trust network), abbreviated as TO-TM. To calculate the trust opinion for the whole trust network, a set of functionalities are necessary that are provided by different modules in the TLEE. In the following section, we first explain the high-level architecture and the functional specifications of the different components of the TLEE. In the second half of the section, we describe the internal API of the TLEE with the TAM (as shown in Figure 2.1), where we detail the interface of the TLEE with the rest of the TAF components.

#### 2.4.4.1 Architecture and functional specifications

In Figure 2.5, we show the high-level TLEE architecture. The TLEE receives input from the TAM ($I\_TA\_TLEE$, through the `runTLEE()` function call) that triggers the execution of the four main components of the TLEE: *DSPG transformer*, *Expression Synthesizer*, *Meta-to-Concrete Expression Converter* and *Evaluator*. We explain the `runTLEE()` function call in more detail later on. Besides the four main TLEE components, TLEE has an *Opinion Information Point* that caches the trust opinions for the trust relationships (TO-TRs), calculated by the TSM. Additionally, it is important to emphasize that the first three components (DSPG transformer, Expression Synthesizer and Meta-to-Concrete Expression Converter) operate symbolically by rewrite expressions on trust opinion variables; whereas, the last component calculates the final numerical trust opinion for the trust network.

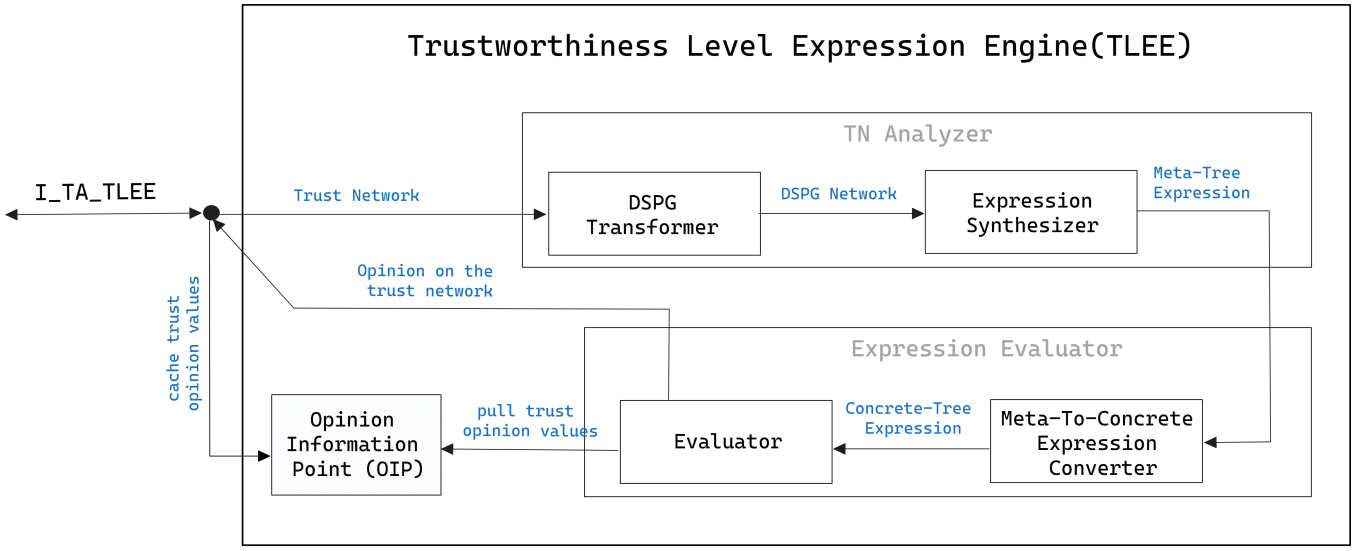



Figure 2.5: TLEE Architecture

As part of the TLEE, by employing trust discounting and fusion operations, the trust network (per proposition) can be effectively transformed into an equivalent single edge per trust proposition. For instance, the trust model instance of Figure 2.4 consists of a chain with two trust relationships under a single trust proposition. As it can be seen in the figure, all the trust opinions that are depicted as part of this trust model instance are on a level of trust relationship (TO-TRs), and are therefore calculated by the TSM. Consequently, the TLEE is responsible for aggregating the trust opinions on the trust relationships (TO-TRs) that have been previously calculated by the TSM, based on which the final trust opinions on the level of trust network (per proposition) is derived (i.e., a single edge from the assessor, namely the Global TAF, to the target trust proposition $p_1$). This consolidated edge encapsulates the comprehensive trust flow from the root of the trust model to the proposition (e.g., one of the leaves of the trust model), while considering the various trust relationships and their corresponding trust opinions. The calculated opinion, captured as $\omega_{p_1}^{G\_TAF}$, represents the ATL value calculated by the TLEE.

In what follows, we explain in more detail the necessary processes to calculate the ATL values and the TLEE's functional specifications: the inputs, the outputs and a short summary of the functionality of the four main TLEE components.

*DSPG Transformer.*

```
Input: trust network
Output: trust network in a DSPG format
```

Figure 2.6 shows the input and the output of the DSPG transformer. DSPG stands for Directed Series-Parallel Graph (DSPG) [15]. As presented in the Trust Assessment Principles in D4.1 [6], a Subjective

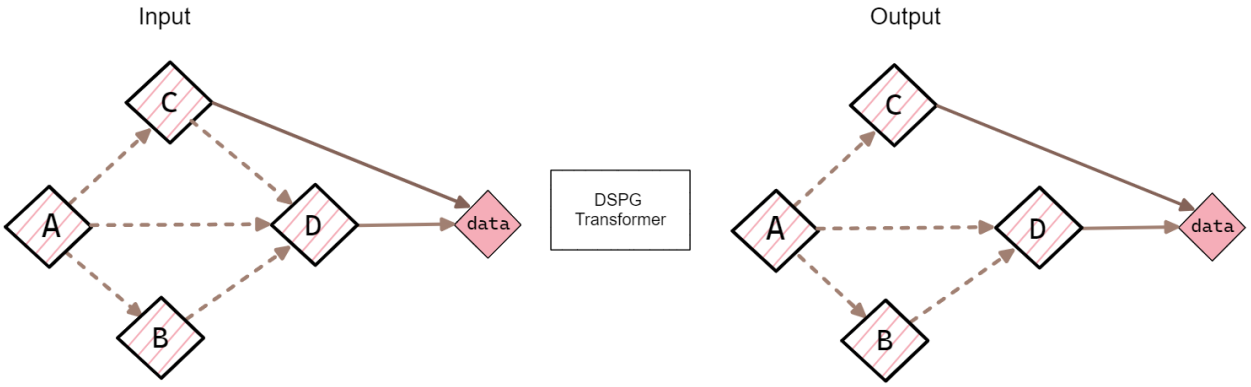



Figure 2.6: DSPG Transformer.

Trust Network (STN) is a graph representation of trust and belief relationships from agents, via other agents and agents to target entities/properties, where each trust and belief relationship is expressed as a subjective opinion [15]. On the other side, for the trust network[1] to be able to be analysed, i.e., for the operators for fusion and trust discounting to be applied to referral and functional trust relationships, the trust network needs to be represented as a Directed Series-Parallel Graph (DSPG). DSPGs have a fundamental role in the TLEE implementation.

**Definition 1** (Directed Series-Parallel Graph (DSPG) [15])**.** A graph is called a Directed Series-Parallel Graph (DSPG) if it can be decomposed as a combination of Series and Parallel graphs and it only consists of directed edges that form paths without loops from the source to the target.

By employing trust discounting and fusion operations, a Subjective Trust Network (STN) structured as a DSPG can be systematically reduced to a single equivalent edge. Because these mathematical operations strictly require a DSPG format to function correctly, the system relies on a dedicated DSPG Transformer to process all incoming trust networks prior to evaluation.

The DSPG Transformer accepts any STN as input and evaluates whether it natively adheres to the required DSPG properties. If the network is already compliant, no modifications are necessary. However, if the input is a complex, non-DSPG network, the component must synthesize it into a valid DSPG structure. The simplest method for this transformation involves systematically pruning the specific edges that violate the DSPG property. To minimize information loss and preserve the overall accuracy of the trust evaluation, the Transformer can also be extended to employ more sophisticated approaches, potentially leveraging additional context — such as the existing trust opinions provided by the TSM — to guide the transformation process.

*Expression Synthesizer.*

```
Input: trust network in a DSPG format
Output: meta-tree expression
```

Figure 2.7 shows the input and the output of the *Expression Synthesizer*. As input this component receives a trust network in a DSPG format, and outputs a Meta-Tree Expression. The functionality of this module is to build the Meta-Tree Expression from the trust network. This includes the calculation of the nesting levels and graph analysis based on which the expression is built.

To synthesize the DSPG into a computational expression, the Expression Synthesizer module executes a systematic graph reduction process driven by edge nesting levels. First, the module analyzes the trust network to identify Parallel-Path Subnetworks (PPS) and calculates a nesting level for each edge, which

[1] Note that the terms 1) trust model, 2) trust network and 3) subjective trust network are used interchangeably throughout this section

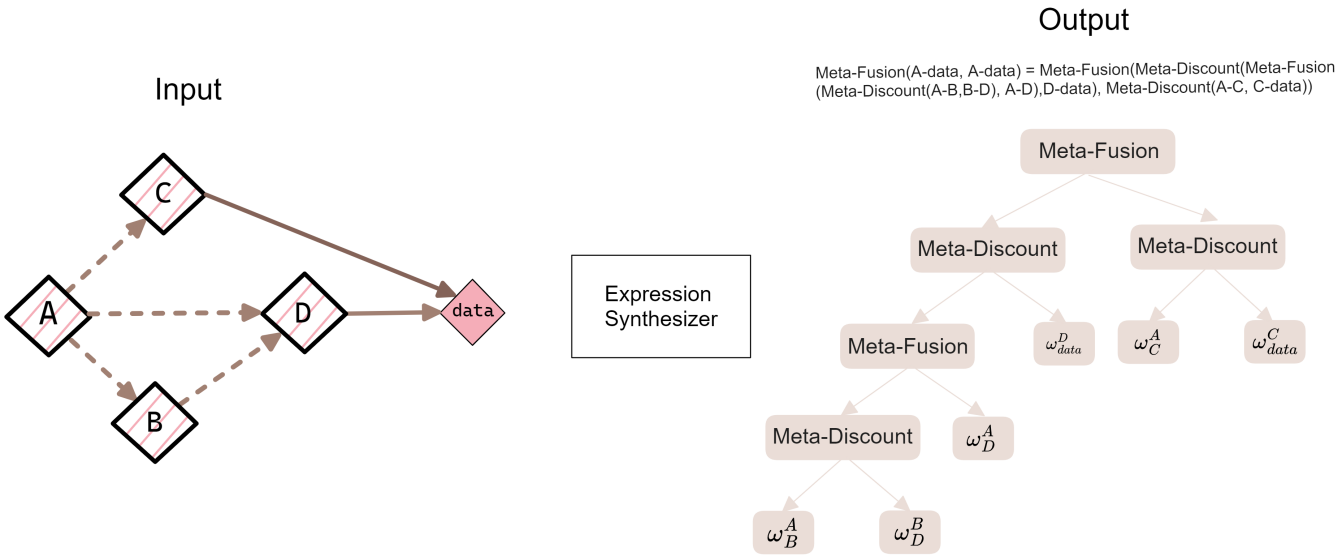



Figure 2.7: Expression synthesizer.

essentially dictates the correct order of mathematical operations. Based on these levels, the synthesizer iteratively collapses the graph. It applies discount operations to sequential trust edges (e.g., combining $\omega^A_B$ and $\omega^B_D$ into $\omega^A_D$ in Figure 2.7) and fusion operations to parallel edges, progressively building the expression until the network is reduced to a single, consolidated trust opinion connecting the source to the target (e.g., $\omega^A_{data}$).

Crucially, to ensure the TLEE remains flexible and math-model agnostic, this entire reduction process operates at a symbolic level. Rather than hardcoding specific Subjective Logic equations, the module generates a meta-tree expression composed of abstract meta-operators (meta-fusion and meta-discount). This design choice decouples the graph traversal logic from the underlying mathematics, seamlessly paving the way for future extensions that might utilize alternative frameworks for trust assessment under uncertainty, such as Epistemic SL, Evidence-Based Subjective Logic (EBSL), or other homomorphic approaches. These abstract meta-operators are only instantiated into concrete mathematical functions in the subsequent processing module.

*Meta-to-Concrete Expression Converter.*

```
Input: Meta-Tree Expression and (optionally) Math Model & Fusion Operators
Output: Concrete-Tree Expression (it could be in tree of String format)
```

We introduced a separate component *Meta-to-Concrete Expression Converter*, with a very simple functionality: take 1) the meta-tree expression that is output in the previous component and 2) optionally a mathematical model (e.g., SL, EBSL, etc.) and return a concrete tree expression. In other words, the functionality of this component is to map the meta operators with concrete operators based on the specific mathematical model that can be given as an input parameter. As highlighted in Chapter 4 of D4.1 [6], in CASTOR we focus on the use of Subjective logic as the main underlying theory for evaluating the trustworthiness of a trust proposition. Hence, the Meta-to-Concrete Expression converter always substitutes the Meta-Operators to the appropriate Subjective Logic operators. Figure 2.8 shows the input and the output of the Meta-to-Concrete Expression Converter. As input the Meta-to-Concrete Expression Converter receives the Meta-Tree Expression from the previous module and, optionally, the mathematical model according to which the operators will be instantiated. The Meta-to-Concrete Expression Converter outputs a Concrete-Tree Expression in tree or String format.

If the fusion operators are not already indicated in the trust model (i.e., the rewritten expression), then the component can also receive a parameter that specifies the concrete fusion operator (e.g., cumulative,

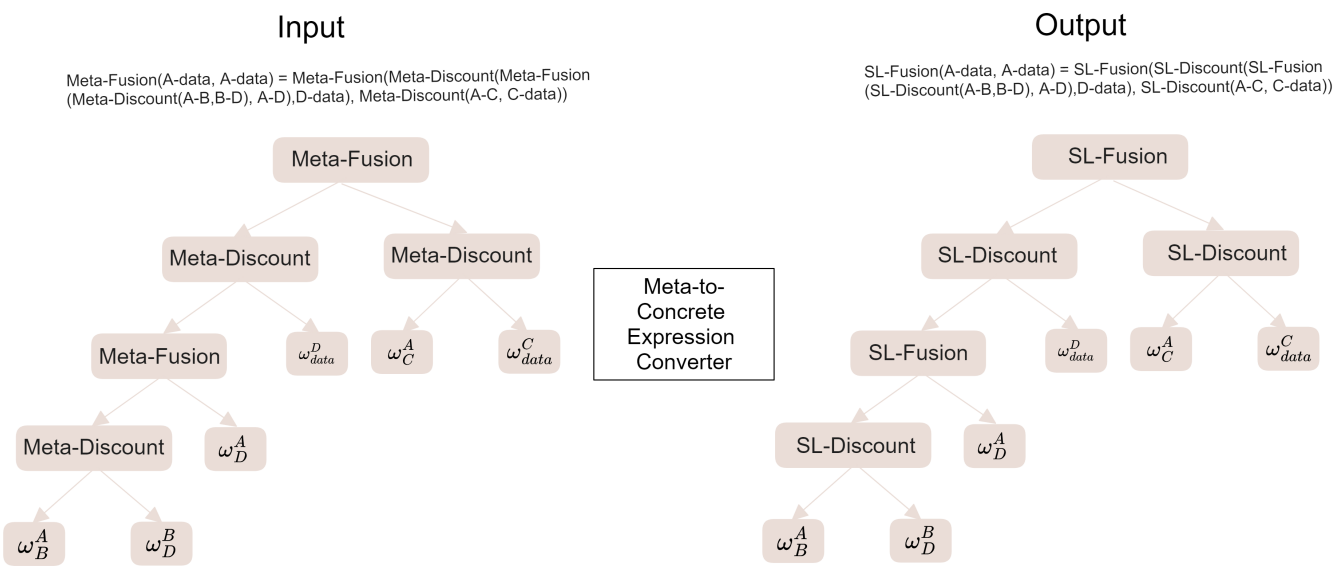



Figure 2.8: Meta-to-Concrete Expression Converter.

average, etc.) that will be used for all the META-FUSION occurrences that do not have a specified fusion operator.

*Evaluator.*

```
Input: Concrete-Tree Expression + Trust Opinions on Trust Relationships
Output: Trust Opinion on Trust Network and (optional) stringified Expression
```

In the final stage of the computation process, the Evaluator ingests the Concrete-Tree Expression generated by the preceding module alongside the cached numerical values of the Trust Opinions on Trust Relationships (TO-TR). These TO-TR values, originally calculated by the Trust Source Manager (TSM) and supplied by the Trust Assessment Manager (TAM), are retrieved on demand from the Opinion Information Point (OIP) depicted in Figure 2.5. Up to this phase, the TLEE has processed the trust network purely at a symbolic level, treating trust opinions as abstract variables residing at the leaves of the expression tree (as shown in Figure 2.8).

The Evaluator transitions this process from the symbolic to the concrete. By substituting the leaf variables with the fetched numerical TO-TR values and executing the concrete discount and fusion operations, it calculates the final numerical trust opinions for the entire trust model, yielding the ATL values for all atomic trust propositions that appear as leaf nodes in the tree structure. The ATL values serve as the primary output of the TLEE. Additionally, the Evaluator can output the stringified Concrete-Tree Expression (i.e., stringified refers to the textual representation of the resulted data structure). This stringified format provides an essential audit trail for applications that require strict tracking of the exact mathematical formulas used to derive the final trust decisions.

Looking beyond the first release, the concept of the OIP presents a unique opportunity for future exploration in the second release of the CASTOR Trust Assessment Framework, particularly within the Global TAF TLEE subcomponent. Because the OIP has access to the single-edge expressions required to derive the ATL for all atomic trust propositions within a trust model instance, it can be extended to formulate highly complex composite trust propositions. Through the `runTLEE()` request, the TAM could supply not only the latest snapshot of the Trust Model Instance but also a set of composite propositions structured as logical expressions of simpler atomic ones. Once the Evaluator computes the baseline atomic Actual Trust Levels (ATLs), it can synthesize them using additional Subjective Logic operators - such as Conjunction - to derive composite trust propositions. For instance, if we want to assess the trustworthiness

of a scenario where both atomic trust propositions, $p_1$ and $p_2$, must be satisfied, we apply the Subjective Logic Conjunction operator. This allows us to evaluate the combined trust opinion for the intersection of these two statements. This concept is directly analogous to standard probability theory: in typical probabilities, the likelihood of two independent events ($x_1$ and $x_2$) both occurring, you calculate their product: $P(x_1 \cap x_2) = P(x_1) \times P(x_2)$. Subjective Logic Conjunction [15] performs a similar mathematical intersection, but it accounts for the dimensions within the trust opinions.

As outlined in D2.1 [8], the ability to process composite trust propositions constitutes a core innovation of the CASTOR Trust Assessment Framework. By elevating atomic, device-level network behaviours into broader link-level, path-level, or even domain-level trust characterizations, the framework will ultimately unlock high-level trust propositions that can be directly linked to advanced traffic engineering (TE) processes: from high-level trust attributes required for the CASTOR Optimization problem as analyzed in Chapter 5 to the enforcement of trust-aware traffic engineering policies as discussed in the CASTOR Orchestration Layer in D5.1 [7].

#### 2.4.4.2 Fusion and trust discount operators

In what follows, we provide more details on the formalisms behind different trust fusion and trust discount operators. As part of Subjective Logic theory, Jøsang has proposed five *fusion operators* [15]. The description of the different SL operators is followed by real-life, exemplary situations where the operators are applicable. Additionally, we provide the mathematical formulas for each of the operators. This subsection can be used as a reference in the evaluation of selecting the appropriate SL operators in the context of the CASTOR Trust Engineering process in Chapter 3.

- **Cumulative Belief Fusion** (CBF) [15] is when it is assumed that, as more and more (independent) sources are included, the amount of independent evidence increases. For example, an IT department could fuse the observed connections of a defined network user over time, which produces an opinion with decreasing uncertainty about the most common locations of that user.

  Let $\omega_X^A$ and $\omega_X^B$ be source $A$ and $B$'s respective opinions over the same variable $X$. Let $\omega_X^{(A \diamond B)}$ be the opinion such that [15]:

  **Case 1:** For $u_X^A \neq 0 \vee u_X^B \neq 0$:

$$\begin{cases} b_X^{(A\diamond B)}(x) & = \frac{b_X^A(x)u_X^B + b_X^B(x)u_X^A}{u_X^A + u_X^B - u_X^A u_X^B} \\ u_X^{(A\diamond B)} & = \frac{u_X^A u_X^B}{u_X^A + u_X^B - u_X^A u_X^B} \\ a_X^{(A\diamond B)}(x) & = \frac{a_X^A(x)u_X^B + a_X^B(x)u_X^A - (a_X^A(x)+a_X^B(x))u_X^A u_X^B}{u_X^A + u_X^B - 2u_X^A u_X^B} \quad \text{if } u_X^A \neq 1 \vee u_X^B \neq 1 \\ a_X^{(A\diamond B)}(x) & = \frac{a_X^A(x)+a_X^B(x)}{2} \quad \text{if } u_X^A = u_X^B = 1 \end{cases}$$

  **Case 2:** For $u_X^A = u_X^B = 0$:

$$\begin{cases} b_X^{A\diamond B}(x) & = \gamma_X^A b_X^A(x) + \gamma_X^B b_X^B(x) \\ u_X^{A\diamond B} & = 0 \\ a_X^{A\diamond B}(x) & = \gamma_X^A a_X^A(x) + \gamma_X^B a_X^B(x) \end{cases}$$

  where

$$\begin{cases} \gamma_X^A & = \lim_{\substack{u_X^A \to 0 \\ u_X^B \to 0}} \frac{u_X^B}{u_X^A + u_X^B} \\ \gamma_X^B & = \lim_{\substack{u_X^A \to 0 \\ u_X^B \to 0}} \frac{u_X^A}{u_X^A + u_X^B} \end{cases}$$

  Then, $\omega_X^{(A \diamond B)}$ is called the cumulatively fused opinion of $\omega_X^A$ and $\omega_X^B$, representing the combination of the independent opinions of sources $A$ and $B$.

- **Averaging Belief Fusion** (ABF) [15] is when it is assumed that including more sources does not imply that more evidence is supporting the final conclusion (i.e., dependence between sources is assumed). In ABF, a vacuous opinion contributes the same weight as every other opinion, therefore ABF has no neutral element and is idempotent. An example of this type of situation is when an examination committee tries to grade a student after having observed her dissertation.

  Let $\omega_X^A$ and $\omega_X^B$ be source $A$ and $B$'s respective opinions over the same variable $X$. Let $\omega_X^{(A\underline{\diamond}B)}$ be the opinion such that [15]:

  $Case\ 1 : u_X^A \neq 0 \vee u_X^B \neq 0 :$

  $$\begin{cases} b_X^{(A\underline{\diamond}B)}(x) & = \frac{b_X^A(x)u_X^B + b_X^B(x)u_X^A}{u_X^A + u_X^B} \\ u_X^{(A\underline{\diamond}B)} & = \frac{2u_X^A u_X^B}{u_X^A + u_X^B} \\ a_X^{A\underline{\diamond}B}(x) & = \frac{a_X^A(x) + a_X^B(x)}{2} \end{cases}$$

  $Case\ 2 : u_X^A = u_X^B = 0 :$

  $$\begin{cases} b_X^{(A\underline{\diamond}B)}(x) & = \gamma_X^A b_X^A(x) + \gamma_X^B b_X^B(x) \\ u_X^{(A\underline{\diamond}B)} & = 0 \\ a_X^{A\underline{\diamond}B}(x) & = \gamma_X^A a_X^A(x) + \gamma_X^B a_X^B(x) \end{cases}$$

  where

  $$\begin{cases} \gamma_X^A & = \lim_{\substack{u_X^A \to 0 \\ u_X^B \to 0}} \frac{u_X^B}{u_X^A + u_X^B} \\ \gamma_X^B & = \lim_{\substack{u_X^A \to 0 \\ u_X^B \to 0}} \frac{u_X^A}{u_X^A + u_X^B} \end{cases}$$

  Then, $\omega_X^{(A\underline{\diamond}B)}$ is called the averaged opinion of $\omega_X^A$ and $\omega_X^B$, representing the combination of the dependent opinions of $A$ and $B$.

- **Weighted Belief Fusion** (WBF) [15] is suitable in cases where the source opinions should be weighted as a function of the confidence of the opinions. When the input opinions have equally confident argument opinions, the fusion is averaging. In the case where one of the opinions is confident and the other is uncertain, then the confident opinion contributes the highest weight, however the combined confidence does not increase. WBF is commutative, considers a vacuous opinion as a neutral element and is idempotent. An example of this type of situations is when, e.g. medical doctors express opinions about a set of possible diagnoses.

  Let $\omega_X^A$ and $\omega_X^B$ be source $A$ and $B$'s respective opinions over the same variable $X$. Let $\omega_X^{(A\hat{\diamond}B)}$ be the opinion such that [15]:

  $Case\ 1 : (u_X^A \neq 0 \vee u_X^B \neq 0) \wedge (u_X^A \neq 1 \vee u_X^B \neq 1)$

  $$\begin{cases} b_X^{(A\hat{\diamond}B)}(x) = \frac{b_X^A(x)(1-u_X^A)u_X^B + b_X^B(x)(1-u_X^B)u_X^A}{u_X^A + u_X^B - 2u_X^A u_X^B} \\ u_X^{(A\hat{\diamond}B)} = \frac{(2-u_X^A-u_X^B)u_X^A u_X^B}{u_X^A + u_X^B - 2u_X^A u_X^B} \\ a_X^{A\hat{\diamond}B}(x) = \frac{a_X^A(x)(1-u_X^A) + a_X^B(x)(1-u_X^B)}{2-u_X^A-u_X^B} \end{cases}$$

  $Case\ 2 : u_X^A = 0 \wedge u_X^B = 0$

  $$\begin{cases} b_X^{(A\hat{\diamond}B)}(x) = \gamma_X^A b_X^A(x) + \gamma_X^B b_X^B(x) \\ u_X^{A\hat{\diamond}B} = 0 \\ a_X^{A\hat{\diamond}B}(x) = \gamma_X^A a_X^A(x) + \gamma_X^B a^B(x) \end{cases}$$

where

$$\begin{cases} \gamma_X^A = \lim_{u_X^A \to 0, u_X^B \to 0} \frac{u_X^B}{u_X^A + u_X^B} \\ \gamma_X^B = \lim_{u_X^A \to 0, u_X^B \to 0} \frac{u_X^A}{u_X^A + u_X^B} \end{cases}$$

$Case\ 3 : u_X^A = 1 \wedge u_X^B = 1$

$$\begin{cases} b_X^{(A\hat{\diamond}B)}(x) = 0 \\ u_X^{(A\hat{\diamond}B)} = 1 \\ a_X^{A\hat{\diamond}B}(x) = \frac{a_X^A(x) + a_X^B(x)}{2} \end{cases}$$

Then, $\omega_X^{(A\hat{\diamond}B)}$ is called the weighted fusion opinion of $\omega_X^A$ and $\omega_X^B$.

The second operator that is fundamental to quantify trust is trust discounting. Trust discounting is innately related to the concept of trust transitivity (see definitions of trust principles in D4.1 [6]).

As part of prior works, Jøsang has proposed three operators for trust transitivity/discounting. We summarize those three operators in the following:

- **Uncertainty Favouring Trust Transitivity [16].** When agent $A$ distrusts recommending agent $B$, it implies that $A$ believes $B$ doesn't know the truth on $X$, leading $A$ to also not know the truth on $X$. This discounting operator is proven to be associative but not commutative, meaning the order of combining opinions matters. In paths with multiple recommending entities, opinion independence is assumed, prohibiting the same entity from appearing twice or more. Eq. 2.1 shows how to calculate the Uncertainty Favouring Trust discounted opinion.

$$\omega_X^{A:B} : \begin{cases} b_X^{A:B} &= b_B^A b_X^B \\ d_X^{A:B} &= b_B^A d_X^B \\ u_X^{A:B} &= 1 - (b_X^{A:B} + d_X^{A:B}) \\ a_X^{A:B} &= a_X^B \end{cases} \tag{2.1}$$

- **Base Rate Sensitive Trust Transitivity [16]** The above operator does not consider the base rate $a_B^A$ when discounting, which may seem counter intuitive. This operator is equivalent to the trust discount operator from [15]. An example of this is the scenario in which a stranger seeks a car mechanic in a town known for honesty and asks the first person he meets to direct him to a good car mechanic. Since the stranger does not know this person he will have a fully uncertain trust opinion him ($b = 0, d = 0, u = 1$). However, the base rate will be high since it's known that citizens of this city are honest. Therefore this should somehow impact the discounted result. Eq. 2.2 shows how to calculate the Base Rate Sensitive Trust discounted opinion.

$$\omega_X^{A:B} : \begin{cases} b_X^{A:B} &= E(\omega_B^A) b_X^B \\ d_X^{A:B} &= E(\omega_B^A) d_X^B \\ u_X^{A:B} &= 1 - (b_X^{A:B} + d_X^{A:B}) \\ a_X^{A:B} &= a_X^B, \end{cases} \tag{2.2}$$

where $E(\omega_B^A) = b_B^A + a_B^A u_B^A$.

Nonetheless, this approach requires caution. For instance, if the stranger has high trust expectations but receives a recommendation from someone with uncertain beliefs, the resulting belief may

seem counter intuitive. This issue could become more pronounced with longer trust paths. Therefore, a safety principle might be to apply base rate-sensitive discounting only to the last transitive link. In summary, the Uncertainty Favouring Trust discounting operator is safe and conservative, while the Base Rate-Sensitive operator can be more intuitive in some situations but requires careful application. Another solution to avoid this problem is to use only the uncertainty favouring operators and set the base rate to $\frac{1}{2}$ assuming that other information that might impact the base rate is already taken into account in the belief and disbelief.

- **Opposite Belief Favouring Trust Transitivity [16].** When agent A distrusts recommending agent B, it suggests A believes that B consistently suggests the opposite of B's true opinion about the truth value of x. Consequently, A not only distrusts x to the extent that B suggests belief, but also trusts x to the extent that B suggests disbelief because two disbeliefs combine to form belief in this scenario. This operator embodies the idea of "your enemy's enemy is your friend," applicable in certain situations. However, it should only be applied when plausible.

$$\omega_C^{A:B} : \begin{cases} b_C^{A:B} & = b_B^A b_C^B + d_B^A d_C^B \\ d_C^{A:B} & = b_B^A d_C^B + d_B^A b_C^B \\ u_C^{A:B} & = 1 - (b_C^{A:B} + d_C^{A:B}) \\ a_C^{A:B} & = a_C^B \end{cases} \tag{2.3}$$

  Note that using this operator can lead to an attack where an attacker who as a bad reputation suggests the real truth, knowing that another agent will not believe it because the attacker is known for suggesting the opposite of the truth.

As part of the future research efforts in CASTOR, we need to agree on the most adequate fusion and discount operators for our use cases, and even potentially develop our own operators based on our concrete requirements. Section 3.3 and Section 3.4 provide an initial set of experiments that provide critical insights on the selection of the appropriate SL operators to be employed within the TLEE for the realization of the atomic trust evaluations at the Global TAF level.

#### 2.4.4.3 Triggering the TLEE

The Trust Level Expression Engine (TLEE) is a core computational component within the TAF ecosystem. It exposes a primary runTLEE() interface, which is exclusively managed and invoked by the Trust Assessment Manager (TAM).

The TAM triggers the runTLEE() function under two primary conditions:

- **Initialization**: Immediately after a trust model instance is created, the TAM invokes the function to allow the TLEE's expression engine to prepare its internal models and structures. For this setup call, the TAM passes nil for the values parameter, and the TLEE is expected to return an empty list of ATLs.
- **Computation**: Whenever the TAM detects that a new trust calculation or update is necessary during run-time.

To ensure strict decoupling and predictable execution, the interaction between the TAM (caller) and the TLEE (callee) is governed by the following architectural contract:

1. **Execution and Structural Guarantees**

- Initialization First: The TAM guarantees that its very first invocation of the TLEE for any model will always be an initialization call with an empty list of values.
- No Redundant Invocations: The TAM will never call the TLEE twice with identical parameters. Every subsequent call implies an update, requiring at least a version increment and corresponding data changes.
- Structural Integrity: If a subsequent function call features the exact same Trust Model ID and structural fingerprint, the TLEE can safely assume that the foundational structure of that trust model instance has not changed, preventing unnecessary recalculations.

2. **Data Ownership and Immutability**

- Input Immutability: All parameters passed from the TAM to the TLEE are strictly immutable. Once the parameters are passed, the TAM will never access or modify those specific values again, effectively transferring data ownership to the TLEE.
- Output Immutability: Similarly, all return values passed back from the TLEE to the TAM are immutable. Once the result is returned, the TLEE relinquishes access, transferring ownership back to the TAM.

3. **State Management**

- Conceptual Statelessness: From the external perspective of the TAM, runTLEE() operates as a purely stateless and side-effect-free function. The output of the call is entirely deterministic, dictated solely by the parameters passed in that specific invocation.
- Internal Optimization: Hidden from the TAM and the broader TAF, the internal implementation of runTLEE() is free to utilize stateful mechanisms, caching, and pre-computations. This allows the TLEE to heavily optimize execution performance without breaking the external stateless contract.

### 2.4.5 Trust Decision Engine

The Trust Decision Engine (TDE) is a core TAF component responsible for making definitive determinations regarding the trustworthiness of a node or data. The Trust Assessment Manager (TAM) delegates these decisions to the TDE whenever an eventual trust decision is required (e.g., following TLEE function calls or prior to dispatching a response to a Trust Assessment Request).

Ultimately, the application issuing the TAR has the flexibility to either request a definitive decision from the TDE or simply receive the ATL to process through its own internal logic.

In the first version of the TAF prototype, a trust decision is a binary decision that can take one of the following two forms:

1. $POS\_TD$: A positive Trust Decision, implying that the assessed entity's ATL meets the Required Trust Level (RTL) and is deemed trustworthy under the current requirements.
2. $NEG\_TD$: A negative Trust Decision, implying that the assessed entity's ATL fails to meet the RTL and is deemed not trustworthy under the current requirements.

To reach a decision, the TDE operates on the relevant trust model instance - provided by the TAM - which contains both the ATL and the RTL. Currently, the TAF supports two primary approaches for this comparison:

- **Projected Probability Comparison**: As outlined in Deliverable 3.1, a baseline method involves calculating the projected probabilities of both the ATL and the RTL as binomial opinions. The decision evaluates the condition $P(ATL) \geq P(RTL)$. If the condition is met, the TDE outputs a $POS\_TD$; otherwise, it outputs a $NEG\_TD$.
- **Independent Component Thresholds**: Relying solely on projected probability can obscure critical nuances. To address this, the TDE also supports evaluating RTL thresholds independently. Rather than comparing a single probability, the RTL can specify distinct boundary conditions for the individual components of a Subjective zLogic opinion. For example, the TDE can enforce a strict minimum threshold for belief ($b$) while simultaneously capping the maximum allowable level of disbelief ($d$).

While the current probability and threshold-based comparisons provide a solid functional baseline, determining the optimal methods for trust evaluation remains an active area of research. By eventually supporting arbitrary multinomial opinions, the TDE can move beyond mere binary (trusted/untrusted) decisions. This granularity will unlock advanced trust-aware traffic engineering (TE) capabilities, allowing the system to output multiple, tiered trust levels. These tiers can then be leveraged by the CASTOR Orchestration Layer to enforce highly specific TE strategies (e.g., continuously maintaining a path profile as a flex-ago topology where the different trust tiers can be encoded as link colours) tailored to the various path profiles with explicit trust requirements.

Future iterations of the TDE will focus on two key architectural enhancements:

- Supporting Multinomial Opinions: Extending the TDE's mathematical capabilities beyond binomial logic to natively handle, evaluate, and compare arbitrary multinomial opinions, allowing for more complex and multi-faceted trust state representations.
- Expressive Evaluation Syntax: Developing a more fine-grained, highly expressive syntax for defining the rule sets used to compare ATLs and RTLs. This will enable domain operators to author highly customized, context-aware Trust Policies that go well beyond simple mathematical inequalities.

## 2.5 Interacting with a TAF instance

Having specified the key elements of the TAF system overview in Section 2.4, we now proceed to outline the interfaces required to realize the trust assessment framework within the context of CASTOR operations. As highlighted in the high-level CASTOR architecture in D2.1 [8], this enables local trust calculations at the router level via the Local TAF agent, as well as domain-level trust characterization of the network topology via the Global TAF. This section presents the interfaces the TAF exposes to other CASTOR artifacts, the external interactions it relies upon for its internal operations, and the listening interfaces used to passively receive notifications from external components.

### 2.5.1 TAF Communication Patterns

Before describing each of the external interfaces in detail, we first outline the underlying message exchange patterns. While all message interactions are fully supported by the Kafka-based messaging infrastructure selected for the CASTOR project, specific interfaces are also additionally exposed via an HTTP server to facilitate easier integration across the wider CASTOR framework. We assume three different message exchange patterns to interact with a TAF instance either at the in-router TNDE architecture or the CASTOR Orchestration Layer:

**Request/Response** messaging facilitates traditional communication patterns where a client interacts directly with a specific component. This approach requires the client to know the server's identity, its exposed capabilities, and its network address. This pattern also supports session semantics (e.g., within the Trust Assessment Framework), where a client must first establish a session via an initial handshake before issuing subsequent, session-specific requests.

**One-Way Notifications** involve messages dispatched from a single entity to a potentially undefined group of receivers, functioning similarly to network broadcasts or multicasts. In a Kafka-based infrastructure, these notifications can be implemented via a well-known, predefined topic that all potential receivers are expected to monitor. Alternatively, if the sending entity knows the intended recipients, it can target them dynamically using specific, context-aware topics (e.g., location-based or cell-based topics).

**Publish/Subscribe** is a pattern for exchanging events or notifications. When subscribers directly contact publishers to initiate a subscription, the two parties are tightly coupled, requiring mutual knowledge of their identities. Such explicit subscriptions are necessary when consumers dictate specific content filters to the publisher, or when subscriptions require explicit state management (e.g., timeout configurations). However, by leveraging messaging middleware like Kafka, this pattern becomes highly decoupled through the use of topics. Publishers and subscribers only need to know the relevant topic name, enabling communication without mutual identity awareness. A notable caveat of Kafka is its retention capability, which allows new consumers to read previously published messages. This behaviour must be actively managed to prevent unintended side-effects and to avoid treating the communication channel as an unbounded archive.

Table 2.2 summarizes the core categories of interfaces that are implemented in this first version of the CASTOR TAF prototype and their realization using the aforementioned communication patterns.

Table 2.2: List of the TAF external interfaces with the rest of the CASTOR architecture

| Interface Category Name | CASTOR Entities | Description |
|---|---|---|
| DLT | CASTOR DLT | TAF requests trust model template based on trust model template ID. In the context of the Global TAF, this interface is also used for pushing TAF reports to the CASTOR DLT. Specification of these interfaces is provided in D5.2 [11]. |
| Evidence Collection Interface (ECI) | Trust Sources | The TAF either actively queries a trust source for evidence or receives updates about evidence from trust sources. Such Trust Sources include the Attestation Source and the FSM Source [10]. |
| Node Discovery Interface (NDI) | TAF agents in Federation Mode | This category enables the associated TAF instance to discover other TAF instances, allowing the formation of the Federated TAF modality. Primarily, this allows the Federation Listener of a TAF instance (e.g., Global TAF) to process incoming messages from other TAF instances and update its internal information base reflected within the Trust Model Manager. Subsequently, this category can be extended to process additional types of messages that could lead to updates in the set of active TMIs. For instance, incoming Stamped Passports from neighbouring TNDIs in the context of IETF Trusted Path Routing [3] could allow the Local TAF agent to spawn a new TMI in order to monitor the trustworthiness of its next-hop vicinity. |

| Interface Category Name | CASTOR Entities | Description |
|---|---|---|
| Trust Assessment Query Interface (TAQI) | TAF agents in Federation Mode | Through these interfaces a TAF instance is able to actively interact with the discovered TAF instances in order to request for additional information during runtime (e.g., active TMIs, trust semantics embedded in TMTs). |
| Trust Assessment Service (TAS) | TN-DSM, TE Policy Engine, SSLA Monitor | The TN-DSM invokes the Local TAF agent in order to initiate the local trust characterization at the TNDI-level. Similarly, the TE Policy Engine and the SSLA Monitor may trigger the Global TAF at the Orchestration Layer for trust evaluations at a domain- or path-level. |

#### 2.5.1.1 Mapping to Kafka Messaging

To bridge the communication gap between the TAF and external components, our prototype relies on Apache Kafka [2] as its primary messaging layer. By combining a distributed commit log with a robust publish/subscribe model, Kafka acts as an active message broker. This architecture ensures strict decoupling between systems: producers simply push data to designated topics, and consumers pull that data independently. This foundational decision fundamentally shapes our service and interface designs, so we must first define exactly how inbound and outbound TAF traffic maps to this Kafka-based infrastructure. While Kafka provides excellent decoupling of components, it inherently enforces a one-way, asynchronous message exchange style. As a result, Kafka is a natural and highly efficient fit for unidirectional messaging, one-way notifications (such as topic-scoped broadcasts), and general publish-subscribe communication.

However, despite our adoption of Kafka, forcing the synchronous Request/Response communication pattern presented in Section 2.5.1 is not ideal. As Kafka is not built for synchronous exchanges, we had to specify a custom communication style to be used with the TAF whenever a Request/Response interaction is required. We intentionally decided to accept this trade-off in the first version of the prototype to maintain a unified and consistent technology stack across the system. To address the limitations of this approach, deliverable D6.1 [9] outlines a more optimal path forward. In D6.1, we leverage the built-in HTTP Server capabilities of a TAF instance in order to handle synchronous communication more naturally. This approach not only alleviates the architectural burden of routing Request/Response traffic through Kafka, but it also improves the handling of specific One-Way notifications through the use of webhooks.

This specification establishes the foundation for Request/Response communication over Kafka by addressing: (i) the identification and addressing of communicating entities, (ii) the reliable mapping of asynchronous responses to their original requests, and (iii) the multiplexing and demultiplexing of concurrent messages. Specifically:

1. Every entity acting as a *server* must provision a dedicated Kafka topic to serve as its conceptual inbox.
    - The topic name must be deterministically derived from the server's entity identity and the name of the service.
    - Clients send their requests directly to this topic.
    - *Note:* To maintain orderly processing, the server entity is the sole *consumer* of this inbox topic.
2. To facilitate asynchronous routing, any request sent to a server's inbox must contain the following mandatory fields:
    - *Client Identity:* The identity of the sending entity, enabling the server to recognize the requester.

- *Service Type Identifier:* Specifies the broader service being invoked.
- *Message Type Identifier:* Specifies the exact request type or operation within that service.
- *Response Topic:* A client-defined topic where the server should route its reply. Clients send messages to a shared server inbox and therefore the server relies on this field to know exactly where to return the response. Clients may opt to use a single unified inbox topic for all their outgoing request/response communications.
- *Message Identifier:* A unique ID for the specific request. This is critical for transactional semantics, allowing the client to later correlate the incoming asynchronous response with the original outgoing request.
- *Payload (Optional):* A blob of data representing the remainder of the request parameters.

3. Upon processing a valid request, the server generates and publishes a service-specific response to the client's specified response topic. This response must include:
   - *Server Identity:* The identity of the entity sending the response.
   - *Service Type Identifier:* Matches the service type from the request.
   - *Message Type Identifier:* Matches or corresponds to the specific operation.
   - *Message Identifier:* Must strictly match the unique message identifier from the original request to guarantee correct correlation.
   - *Payload (Optional):* A blob of data representing the service-specific execution results.
4. Kafka forces a decoupled, one-way messaging model, so clients must implement specific logic to handle responses:
   - *Message Demultiplexing:* Clients issuing concurrent requests must demultiplex incoming responses arriving in their receiving topic. This is achieved by matching the tuple of `(Server Identity, Service Type Identifier, Message Identifier)`.
   - *Handling Asynchronous Execution:* The client loses native synchronous execution flow (i.e., a blocking wait) when issuing a request over Kafka. Since a separate activity (the Kafka consumer thread) handles the incoming response, the client application must implement language-specific asynchronous concurrency patterns—such as Futures, Callbacks, or Continuations—to resume the logic once the response is successfully received and correlated.

Please note that the specification of how to handle request/response communication might also apply to parts of the publish/subscribe message exchanges when subscription requests are handled explicitly.

Here, the subscribing entity would invoke a request to the publishing entity to ask for the name of the topic that is used for notifications. If subscriptions are handled implicitly (i.e., by directly subscribing to a named, well-known Kafka topic), this does not apply.

### 2.5.2 Trust Assessment Service

The Trust Assessment Service (TAS) represents the primary service a TAF provides to its clients - the continuous assessment of trustworthiness of relevant entities. In order to receive such a trust assessment, clients (i.e., the TN-DSM for a Local TAF agent and the TE Policy Engine for the Global TAF) have to establish a session with the TAF and indicate the trust model template they are interested in. Once a session is established, the client can then either request trust assessments or subscribe for notifications in case of changes in the trustworthiness of the target entity.

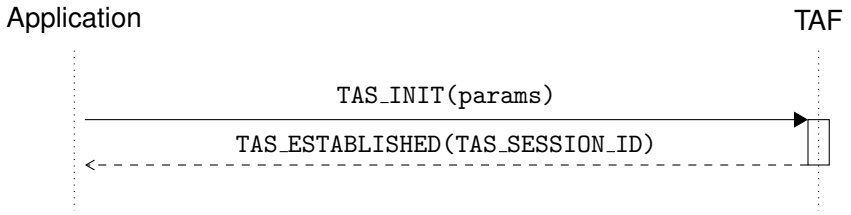
This service represents a stateful, session-based protocol in which a client indicates interest in a target trust model (by specifying the trust model template to be used) and the TAF ensures the processing of matching trust model instances for the lifetime of that session.

A client can establish zero or more sessions with a TAF. Each session requires exactly one trust model template to be used within that session. This means that a single client is allowed to run multiple sessions in parallel with the same TAF. Having multiple concurrent sessions is particularly necessary when a client is interested in trustworthiness assessments for different trust model templates at the same time.

For receiving ATLs during an established session, the client can choose between pull-based Trust Assessment Requests (TARs) (i.e., polling using request/response) or event-based subscriptions (i.e., publish/subscribe).

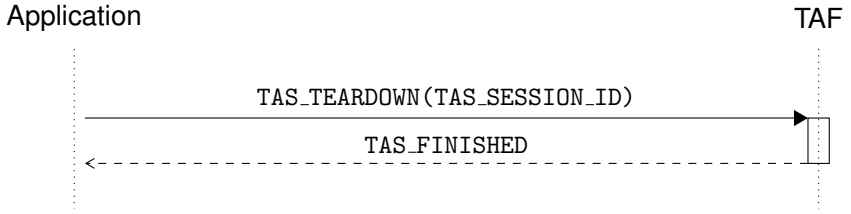
#### 2.5.2.1 Session Initiation

In session instantiation, the application tells the TAF which trust model template should be used by sending a TAS Initialization Message (TAS_INIT) which includes the Trust Model Template ID among other parameters. The TAF then internally prepares session handling for this session (e.g., by preparing internal data structures). Once ready, the TAF will respond with the ID of the new session.



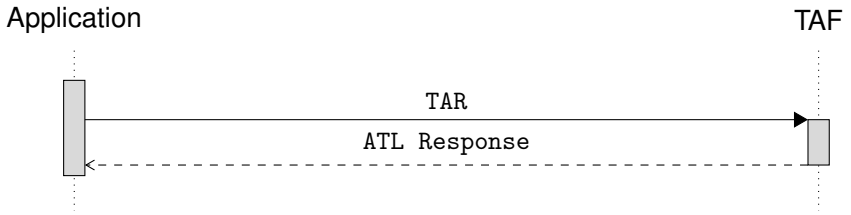

#### 2.5.2.2 Session Tear-Down

Before shutting down, the application must signal the TAF to tear down the session. This allows the TAF to free all resources allocated for this session. For simplicity, the current design does not include a lease-based session model in which sessions would be purged automatically in case of client inactivity or missing renewals of the session.



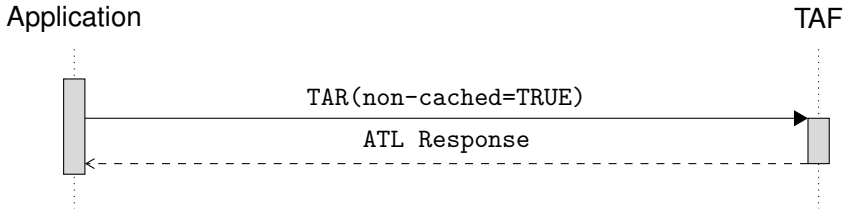

#### 2.5.2.3 Regular Trust Assessment Requests

Once a session has been established, the application can send TARs to the TAF whenever a recent trustworthiness assessment is needed. The TAF will respond to that request in a best-effort way and will potentially use cached results that had been calculated recently.



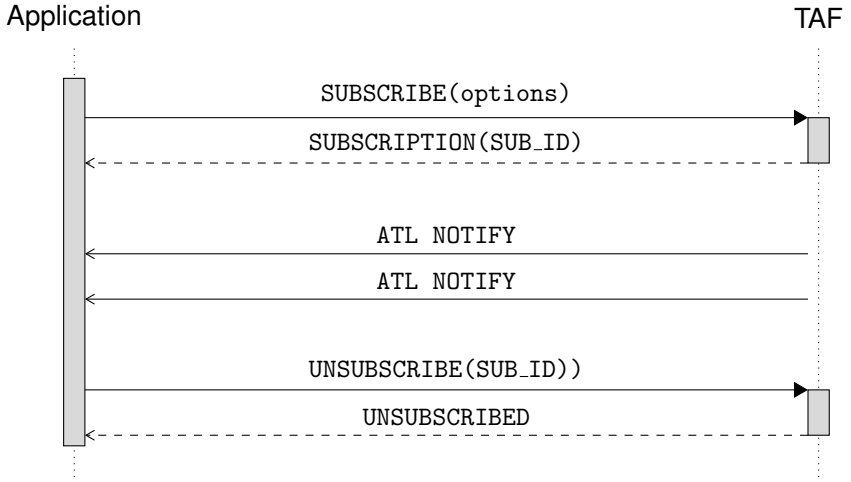

#### 2.5.2.4 Non-Cached Trust Assessment Requests

In case an application requires freshly calculated assessments and does not accept results potentially precalculated recently by the TAF, it can request non-cached results from the TAF. This is a deliberate trade-off by the application to favour the latest results over shorter response times.



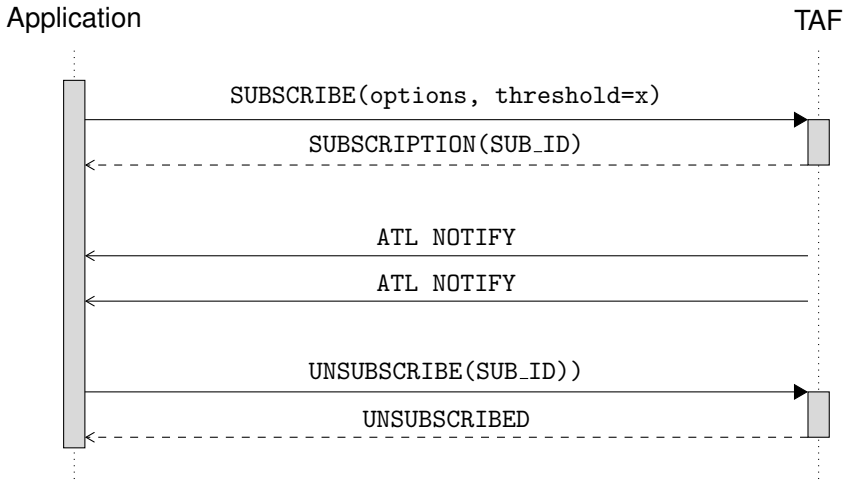

#### 2.5.2.5 Subscription for Trust Level Changes

In addition to the previously described polling-based approaches, an application can also subscribe for changes of ATLs. In the first approach, the application subscribes to any change. This means that whenever an ATL has been recalculated and is not identical to the previously published value, the application will get notified.




#### 2.5.2.6 Subscription for Threshold-based Changes

An alternative approach for publish/subscribe includes a threshold provided by the application during the subscription. In this case, the TAF will notify the application whenever an ATL has been (re-)calculated and the resulting value exceeds the threshold specified by the application.




### 2.5.3 Distributed Ledger Technology Interface

From the perspective of the TAF, the CASTOR DLT represents a distributed, tamper-evident ledger that serves two primary use cases: it acts as a key-value store for retrieving Trust Policies, and as an immutable log for recording Trust Reports.

Consequently, the TAF interacts with the DLT through two main operational flows. Full description of the interactions with the CASTOR DLT is presented in D5.2 [11].

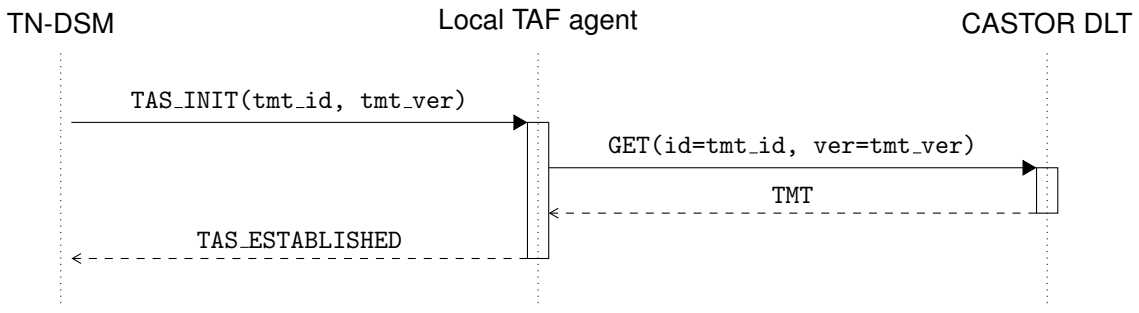
#### 2.5.3.1 Resolving Trust Policies

The DLT stores Trust Policies indexed by their ID and version number. Conceptually, the storage key consists of a tuple (trust policy identifier, trust policy version), and the value consists of the serialized representation of the actual Trust Policy. The TAF utilizes this query interface whenever it receives a request to establish a trust assessment session that references a Trust Policy it does not yet have locally. To resolve the policy, the TAF queries the CASTOR DLT:

- If the session request specifies a version number, the TAF retrieves that exact version of the Trust Policy.
- If the session request only specifies the identifier, the TAF defaults to retrieving the latest available version of that Trust Policy from the DLT.

If the specified Trust Policy (or the requested version) is unknown to the DLT, the remote resolution fails. Ultimately, if the exact Trust Policy is already stored locally, the TAF can continue to operate; otherwise, it is unable to function and cannot instantiate the session or the corresponding trust model instances.

The following sequence diagram shows how the Trust Policy is established within a Local TAF agent as part of the overall TNDE flow presented in D3.2 [10].

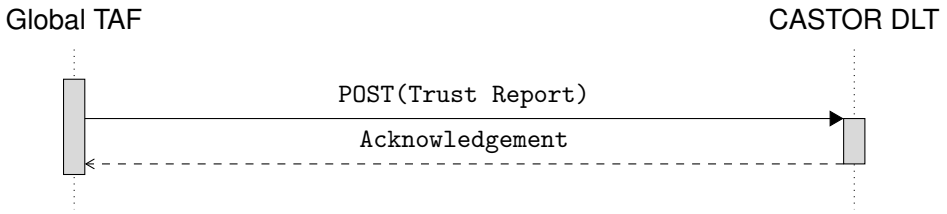


#### 2.5.3.2 Recording Trust Reports (Write Operation)

In addition to resolving policies, the TAF uses the DLT to persistently store the outcomes of its assessments. Once a trust assessment session yields results, the TAF generates a Trust Report.

The TAF then writes this Trust Report to the CASTOR DLT. This guarantees that the results of the trust assessment are immutably recorded, providing a transparent, tamper-evident, and globally auditable trail of trust evaluations for external entities and future reference.

In what follows, we capture the interface regarding the posting of Trust Reports by the Global TAF at the Orchestration Layer.



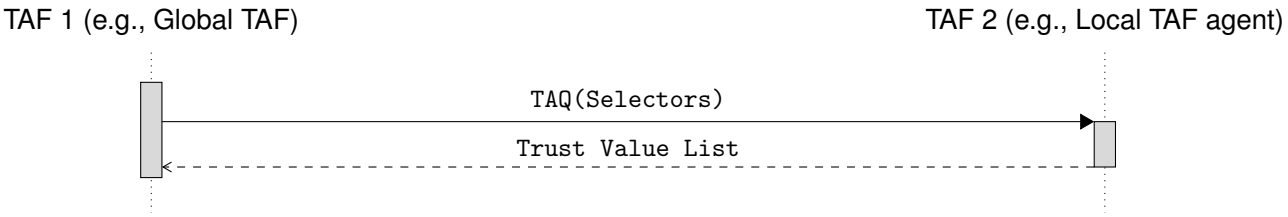

As presented in D3.2 [10], the recording of Local TAF reports and other TNDE-related information (e.g., traces) is mediated through the TNDI-SP protocol.

### 2.5.4 Trust Assessment Query Interface

The Trust Assessment Query Interface represents an auxiliary service of a TAF and enables other TAF instances to query trust assessments of that instance. Hence, this interface enables TAF-to-TAF interaction and can be seen as the first building block for pull-based federated behaviour in which multiple TAFs enhance their own perspective by requesting for assessments provided by other TAFs. It represents a stateless, read-only query protocol in which a TAF can ask for trust values of a remote TAF.

#### 2.5.4.1 Trust Assessment Query

A trust assessment query (TAQ) contains a selector to identify trust values the querying TAF is interested in. A selector specifies a trust model template type and potential paths in corresponding trust model instances with optional attributes (i.e., to identify certain trust models and contained nodes). By using wildcards, a selector allows for broader queries, e.g., queries that can ask for any trust model instance in which a node with a certain ID is part of. The queried TAF then returns a potentially empty list of trust values satisfying the selector.




### 2.5.5 Evidence Collection Interface

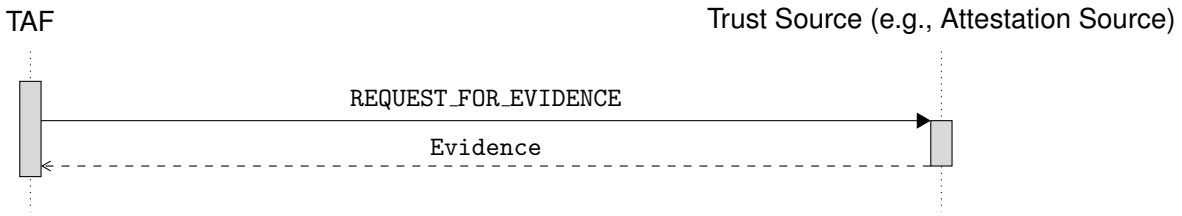
The Evidence Collection Interface provides different mechanisms to collect information about other entities to be used by the trust source manager.

This includes: (i) actively polling known nodes for evidence, (ii) subscribing for evidence notifications at know nodes, or (iii) receiving evidence updates via one-way messages.

The TAF implements the following interaction scheme so that the TSM can choose between them at runtime.

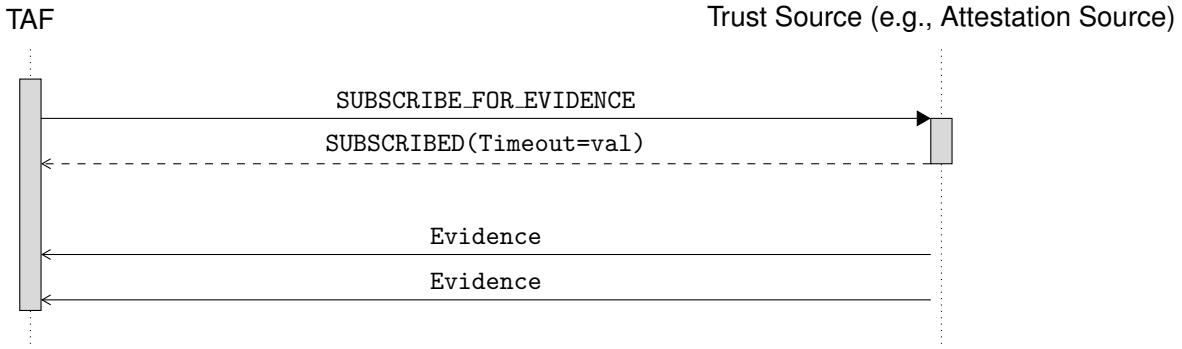
#### 2.5.5.1 Pull-based Evidence Collection

In this case, the TAF knows about an available evidence source (e.g., by using the Node Discovery Interface) and specifically queries the target for evidence.



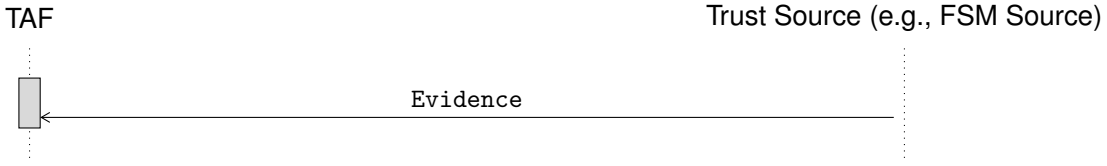

#### 2.5.5.2 Subscription-based Evidence Collection

As an alternative to the previous interaction, a TAF can also use a known source of evidence and subscribe for evidence updates. These subscriptions have a timeout and need to be renewed.




#### 2.5.5.3 One-Way Evidence Notifications

A third option is the reception of an unsolicited message that contains evidence. This evidence can then be used by the TAF, if applicable.

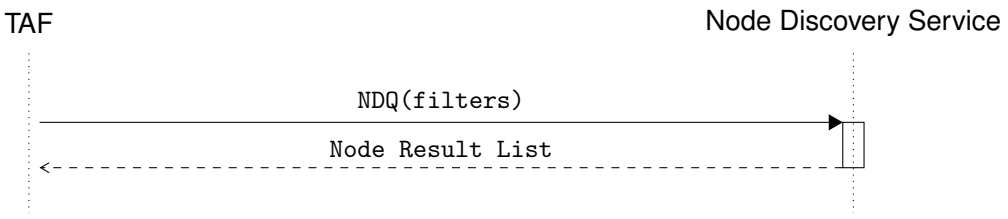



### 2.5.6 Node Discovery Interface

A Node Discovery Service allows the TAF to discover the presence of other entities (e.g., Local TAF agents, or TNDEs in general). As opposed to the Trust Assessment Query Interface in Section 2.5.4, this constitutes a first building block towards the push-based federation of the CASTOR TAF. Thus, this category of interfaces provides the capabilities for a TAF instance to act as a client.

#### 2.5.6.1 Node Discovery Query

A Node Discovery Query (NDQ) contains a potentially empty list of filters of discoverable nodes that the TAF wants to query. Filters can include the identifier of a node or the type of the node. The Node Discovery Service returns a list of all matching nodes on which the service has information.



# Chapter 3

# Evaluation of the CASTOR Trust Assessment Framework

In the following chapter, we present an evaluation plan aimed at demonstrating the core functionalities of the Trust Assessment Framework (TAF). This plan aims to validate correct TAF functionality from several perspectives, such as the implementation of fusion, discounting and the consideration of historical evidence. At this stage, the evaluation plan does not target performance and low-latency; rather, the focus is on behavioural validation through the implementation of several trust models capturing fundamental trust assessment scenarios forming the foundation of the TAF's capabilities.

Following this is a discussion on the implementation of these trust models, covering the various trust relationships within and how each trust model contributes to the established evaluation plan. Finally, preliminary results are presented in relation to the TAF's capabilities.

## 3.1 Trust Assessment Evaluation Plan

In this section, we detail an evaluation plan for the TAF that targets several core functionalities. For each functionality, we break down the concepts and rationale behind its implementation, as well as how we aim to test its capabilities. Overall, these tests aim to validate the correct behaviour of the trust assessment process from the perspective of the TAF. Preliminary results from these experiments are presented in Section 3.2.2, 3.3.2 and 3.4.2.

**Consideration of Historical Assessments.** It is important to consider how a trust opinion gets updated by both the most recent evidence as well as previous evaluations of that specific trust opinion. Consider the scenario in which a Global TAF receives a trust opinion from a Local TAF that differs wildly from previously reported opinions. Without any consideration of previous information, the new mostly recently calculated opinion will be taken at face value (aside from any necessary discounting and/or fusion steps) and implicitly over-write any and all previous knowledge, even if the latest opinion does not provide an accurate representation of the level of trust. By taking previous information into account, we can instead observe how trustworthiness as a "state" is evolving. This would allow the TAF to, for example, pay more attention to significant drops in trustworthiness level and accurately convey such steep degradations of trust, whilst simultaneously not allowing overly confident opinions of positive evidence to skew the trust posture of the network.

An Alpha value $\alpha$ can be used to determine the influence that previous evaluations have on the trust assessment process. This Alpha value is a parameter within the trust model that needs to be fine-tuned through experimentation to ensure that historical data is considered to the appropriate extent. Crucially, $\alpha$ must be dynamically adjusted to ensure a higher weighting when the trust level is dropping. As explained above, this conservative approach guarantees that the current trust level can be rapidly reduced when an

event results in suspicious or negative behaviour that would result in a lower ATL. It also ensures that it is now more difficult for the trust level to increase to its original higher level. It also ensures that the trust level is not overly-increased in the case where more positive evidence is provided. In practice, a higher $\alpha$ denotes more weight to be given to the latest opinion calculated from the latest evidence. An $\alpha$ of 0.5 would imply that the most recent opinion and the previous opinion are treated with equal weight.

One option is to maintain the previously-calculated trust level and compare it to the most recent; if we are transitioning to a less-trustworthy state then $\alpha$ can be increased, and if we are transitioning to a more-trustworthy state then $\alpha$ can be reduced, again to pre-defined values, based on simple conditional logic. As a step up from this, we could also consider a "window" of previously-calculated trust opinions (an adjustable time-frame within which trust opinions should be considered). Then, this set of opinions can be averaged and used as a basis of comparison to the most recently-calculated opinion.

We must also consider the fact that as it currently stands, asynchronous trust assessment (i.e. assessment taking place when evidence is received rather than through consistent monitoring) makes this more complex; any received evidence and opinions will always have an "age" associated with them, meaning that some time will have passed in every scenario and therefore the information may be out of date (leading to increased uncertainty). This concept of "longevity" must also be taken into account, and is discussed in what follows.

As we transition to the second release of the CASTOR TAF, an exploration aspect would be to extend these ideas with the concept of "time evolution". Over time, the validity of a trust opinion should decrease such that older trust opinions do not overly-influence the result of a trust decision. The ability of the TAF to produce high-quality trust opinions in real-time relies on freshly-collected evidence and rapid evaluation. An opinion from minutes (or even seconds) ago may no longer reflect the most up-to-date trust posture of the topology, and this uncertainty must be taken into account if the dated opinion is being used as part of the trust assessment process.

It will be important to carefully optimise the rate at which the belief of an opinion decays over time. Indeed, the rate of decay may even be dependent on the context in which the TAF is running; more safety-critical domains may require a more dramatic decay whereas other domains that are more tolerant of noise may permit the opposite. The rate of decay will also implicitly affect the latency with which the TAF is able to produce trust opinions; domains in which a lower rate of decay is permitted will be able to perform trust assessments faster in general, as older opinions could still be used without the need for re-evaluation.

However, for the first CASTOR release, these experiments aim to derive an optimal implementation of such an $\alpha$ value to support the appropriate consideration of historical evidence such that steep declines in trust are captured, and cases of continuous positive evidence are not overbearing. These experiments are detailed in Section 3.2, and preliminary results highlighting the importance of taking historical evidence into consideration are captured in Section 3.2.2.

**Opinion Fusion and Evidence Accumulation.** At the heart of the trust assessment process is the collection of evidence, and the ability to fuse multiple trust opinions into a single trust opinion that accurately takes into account the opinions of all participating entities whilst minimising losses in precision. There are various options to accomplish both tasks:

- As analyzed in the corresponding evaluations in Section 3.4, the chosen fusion operator directly influences how opinions are combined. The fusion operator determines how belief, disbelief and uncertainty are effectively "merged" from the opinions of participating entities. However, different fusion operators enact different assumptions about the handling of evidence, for example how overlapping, contradictory or incomplete evidence should be treated. The correct fusion operator is context-dependent, and choosing incorrectly may lead to evidence being over-counted, incorrectly disregarded or drastically distorted. These experiments aim to determine the most appropriate fusion operator for the trust assessment process to ensure that the resultant fused trust opinion is as faithful and accurate as possible to its sources. As part of these experiments, we must be able to

capture the fusion of multiple Local TAF agents on a single trust proposition, as well as

- Section 2.4.3 specifies the TSM framework, focusing specifically on the evidence quantification process. In essence, this process dictates how positive and negative pieces of evidence mathematically contribute to the overall trust assessment. Raw observations must be translated into verifiable quantities that directly inform a trust opinion's belief, disbelief, and uncertainty components. Proper quantification is therefore essential to ensure the trust models correctly interpret and distinguish between strong, weak, positive, and negative evidence. Improper evidence quantification (such as overweighting noisy data) can yield overly confident or biased opinions that fail to accurately reflect reality. Furthermore, evidence can be systematically weighted to allow predetermined, highly reliable sources to exert greater influence over the final trust assessment. Applying these weights in a sound, methodical manner ensures that diverse evidence sources receive the appropriate level of consideration. An initial evidence quantification process based on Jøsang's equations, where all types of trustworthiness evidence for a single trust proposition are weighted equally, is presented and evaluated in Section 3.2.

**Opinion Discounting.** Implemented trust models must be able to perform a discounting step such that opinions received from a Local TAF agent can be modulated with respect to the capabilities of the reporting entity. For example, if the Global TAF receives a confident opinion from a Local TAF, this opinion must not be taken at face value and should be discounted (i.e. have its uncertainty increased) by an amount relative to the Global TAF's faith in the Local TAF agent's ability to provide accurate trust opinions.

Similarly to fusion and evidence accumulation, an appropriate discounting operator must be selected. The discounting operator directly influences how much of the reporting node's belief and disbelief is inherited rather than converted into uncertainty. An incorrect discounting operator can lead to several problems. If discounting is too weak, unreliable sources could overly influence the trust assessment process with overly confident evidence. Too strong, on the other hand, and even known reliable sources may effectively be cancelled out, resulting in too much uncertainty at the Global TAF level. The evaluation presented in Section 3.3 showcases the importance of selecting the appropriate discounting operator for a given (trust) context.

## 3.2 Evaluation 1: Consideration of Historical Evidence in the Local TAF agent

In this section, we provide a walk-through of the evaluation setup used to conduct the experiments detailed in Section 3.1. We detail the individual trust models used in each scenario, we provide examples of the evidence used by the TAF, and we discuss the quantification algorithm used to map raw evidence appraisals into concrete trust opinions.

### 3.2.1 Evaluation Setup

As detailed in Section 3.1, the TAF must be able to correctly take into consideration historical evidence. In other words, rather than blindly accepting fresh evidence without question, the TAF, if required, must have the capability to analyse previous evidence in relation to the same proposition. In doing so, the TAF can compare how the trustworthiness of some phenomenon has evolved over time. Depending on if we are moving to a more or less trustworthy state, the weight of historical evidence can be adjusted to ensure that steep drops in trustworthiness are captured whilst increases in trustworthiness are gradual.

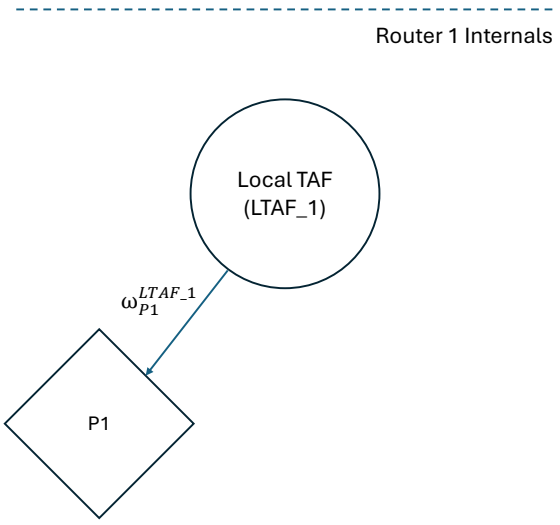



Figure 3.1: The trust model implemented to verify historical evidence consideration, depicting a Local TAF forming an opinion on P1 (a trust proposition stating that the device has high integrity).

The trust model shown in Figure 3.1 was used to test this scenario. Here, the Local TAF (LTAF_1) of Router 1 is forming a trust opinion on the trust proposition P1 that states: “Router 1 has high integrity”. We acknowledge that in general, an atomic trust proposition should be broken down to its minimum constituent parts (i.e., the proposition stating high integrity is in fact composite, composed of multiple atomic trust propositions in relation to individual pieces of evidence such as secure boot and control flow integrity). However, the focus for this evaluation was to focus on the consideration of historical data, so the chosen method of expressing the relationship between evidence and trust propositions is not relevant in this case.

In a real-world environment, a trust model would be designed such that separate atomic trust propositions are defined for each evidence source. Then, after individual opinions are derived at the Local TAF and sent to the Global TAF, the Global TAF would aggregate these opinions using logical operators to form an opinion that reflects a composite trust proposition (i.e. a higher-level, more abstract concept such as “high integrity” as opposed to secure boot being enabled). This functionality will be implemented and evaluated as part of CASTOR’s second release, and will also facilitate the derivation of link and path-level trust opinions at the Global TAF level.

We defined three scenarios over which to run the experiment: *steady*, *drop* and *rise*, each depicting different flows of evidence that may be encountered in a network environment. Whilst these scenarios are not exhaustive, they provide a varied foundation for verifying correct TAF behaviour when historical evidence is being considered. In each scenario, the experiment was performed at the Local TAF level, simulating the Local TAF performing a self-assessment of its own device-level integrity using locally-collected evidence.

In order to simulate the Local TAF running within a wider functioning network topology, we ingest 10 reports of evidence uniformly across 10 time-steps. In practical terms, this emulates the Local TAF performing a local self-assessment of its integrity at uniform time intervals across a given time-frame. That being said, this behaviour is not strictly limited to the Local TAF in practice. It should be noted that the results obtained from these experiments are also applicable in the case that historical evidence consideration is implemented within a Global TAF.

**Evidence Collection.** Evidence was provided to the Local TAF in JSON format through the Kafka UI. In this case, an evidence report consists of six separate evidence appraisals that each contribute to the

proposition P1 in Figure 3.1. An example of such an evidence report can be seen in Listing 1. This file provides appraisals for each evidence source, where an appraisal of $> 0$ indicates positive evidence, otherwise the evidence is considered as negative. Therefore, in the example provided, all evidence is in favour of vRouter1 having high integrity.

In practice, any number of evidence sources can be defined. Furthermore, the evidence itself is intrinsically linked to the Risk Assessment process discussed in Chapter 4. Each piece of evidence provides guarantees that the relevant risks have, or have not, been mitigated. Therefore, in a real scenario, the provided evidence sources are dependent on the risk assessment process itself, rather than chosen arbitrarily. Finally, we add that in the context of our experiments, evidence is guaranteed to be provided at each time-step. In practice, there are no guarantees that evidence is received at uniform time intervals, or indeed that any evidence is received at all. In these situations, mitigations can be deployed such as increasing the uncertainty of the opinion in relation to subsequent evidence reports. However, this functionality will be implemented and tested as future work.

**Listing 1** Example of Evidence Collected by a Local TAF

```
"trusteeReports": [
  {
    "attestationReport": [
      {
        "appraisal": 1,
        "claim": "SECURE_BOOT"
      },
      {
        "appraisal": 1,
        "claim": "ROLLBACK_PROTECTION"
      },
      {
        "appraisal": 1,
        "claim": "ACCESS_CONTROL"
      },
      {
        "appraisal": 1,
        "claim": "EPC_ENFORCEMENT"
      },
      {
        "appraisal": 1,
        "claim": "CONTROL_FLOW_INTEGRITY"
      },
      {
        "appraisal": 1,
        "claim": "CONFIGURATION_INTEGRITY_VERIFICATION"
      }
    ],
    "trusteeID": "vRouter1_Integrity"
  }
]
```

**Evidence Quantification.** The quantification phase enables us to map raw evidence appraisals into the belief, disbelief and uncertainty components of the resultant subjective logic opinion. Our chosen method to achieve this is based on Equation 3.1[15].

$$b_x = \frac{r_x}{W + r_x + s_x}, d_x = \frac{s_x}{W + r_x + s_x}, u_x = \frac{W}{W + r_x + s_x} \tag{3.1}$$

Here, $r_x$ corresponds to the quantity of evidence in support of the trust proposition $x$, $s_x$ denotes the quantity of evidence against it, and $W$ is a weight applied to uncertainty. This weight was set to two in our experiments, although in practice should be set based on context to ensure that the trust model is not overly sensitive to a single piece of evidence whilst still only requiring a reasonable amount of evidence to reduce uncertainty. The experiments in this evaluation focus on the effects of an alpha value scaling a trust opinion with respect to previously observed evidence and so the optimisation of $W$ is not currently within scope. Ultimately, the resultant values form the trust opinion $\omega_x$ that the TAF holds on the trust proposition.

At this stage, we do not weight any individual piece of evidence in our evidence quantification phase. That is, each of the six pieces of evidence shown in Listing 1 are treated equally. A point of future work is to apply weighting to certain types of evidence based on the path profile requirements and specific trust property being assessed, allowing for certain appraisals to carry more influence in the resulting opinion.

**Alpha Application.** In order to account for historical data, we implemented an alpha value in the trust model as a tuple. The first element represents the alpha value to be used in the event that we are transitioning to a less-trustworthy state (therefore placing greater emphasis on the most recent observation), and the second represents the alpha value to be used in the event that we are transitioning to a more-trustworthy state (therefore placing greater emphasis on the previous observation). Intuitively, the alpha value is used to scale the belief and disbelief components of an opinion, as seen in Algorithm 1.

**Algorithm 1** Update Belief and Disbelief Using Alpha Values

**Require:** Previous state $\omega_{\text{prev}}$, new state $\omega_{\text{new}}$
**Require:** Weights $\alpha_1, \alpha_2$
**Ensure:** Updated values $belief$, $disbelief$
**if** $\text{Belief}(\omega_{\text{prev}}) > \text{Belief}(\omega_{\text{new}})$ **then**
  $belief \leftarrow (1 - \alpha_1)\,\text{Belief}(\omega_{\text{prev}}) + \alpha_1\,\text{Belief}(\omega_{\text{new}})$
  $disbelief \leftarrow (1 - \alpha_1)\,\text{Disbelief}(\omega_{\text{prev}}) + \alpha_1\,\text{Disbelief}(\omega_{\text{new}})$
**else**
  $belief \leftarrow (1 - \alpha_2)\,\text{Belief}(\omega_{\text{prev}}) + \alpha_2\,\text{Belief}(\omega_{\text{new}})$
  $disbelief \leftarrow (1 - \alpha_2)\,\text{Disbelief}(\omega_{\text{prev}}) + \alpha_2\,\text{Disbelief}(\omega_{\text{new}})$
**end if**

It is important to note that, as belief and disbelief are scaled independently during the opinion formation process, uncertainty is implicitly scaled with respect to alpha, as $b + d + u = 1$. It is also crucial to note that the baseline level of trust (i.e. the opinion at build-time) was set such that $\omega_x^{LTAF} = (0.7, 0.1, 0.2, 0.5)$, i.e. the experimental runs start in a state of high belief, low disbelief and low uncertainty.

Choosing an appropriate value for alpha is dependent on several factors. For example, it may depend upon the trust property being assessed as well as the criticality of the operational environment in which the TAF is running. Safety-critical environments may be highly sensitive to trust degradation. In addition, the chosen value must be carefully considered with respect to the risk assessment process discussed in Chapter 4, to prevent the situation in which an ATL can never exceed the calculated RTL threshold due to overly scaling the belief and disbelief (or vice versa).

### 3.2.2 Evidence-based theory and historical opinion consideration

In the following section, we present results from preliminary experimentation conducted to verify the correct behaviour of a Local TAF that takes historical opinions into consideration. These experiments

were ran using the trust model presented in Figure 3.1.

#### 3.2.2.1 Steady Evidence

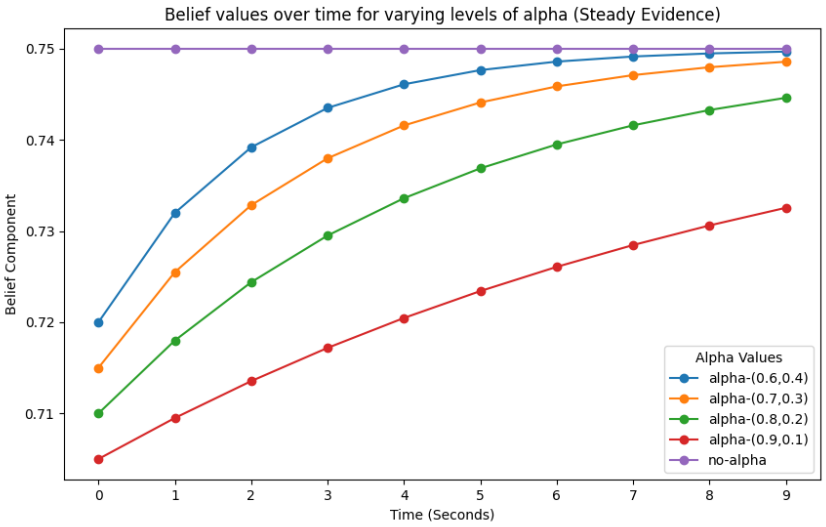



Figure 3.2: Belief over time for various values of $\alpha$ for evidence that doesn't change

Figure 3.2 shows the belief component of the trust opinion formed by the Local TAF at each time-step across the 10-step time-frame. Also included is an experimental run in which no alpha value is used, such that the differences in behaviour when using an alpha value can be contrasted against the default behaviour. In this case, the belief immediately jumps to $0.75$ (as there is no alpha value impacting the rate at which belief can rise from the initial value of $0.7$), and remains at this value across the entire run. This behaviour is to be expected; the evidence does not change, and therefore the belief remains the same.

At every time-step in this scenario, we are transitioning to an equally-trustworthy state meaning that the lower of the alpha pair values was used (see Algorithm 1). This places greater emphasis on historical data. As seen in Figure 3.2, the chosen alpha tuple directly affects the rate at which the belief component can grow. For example, in the case of $(0.9, 0.1)$, the rate of growth for belief is at its lowest. This is because the lower of the alpha values is used (in this case $0.1$). As an example, at t=0, the belief value is $0.705$, built from 90% of the previous state (where $b = 0.7$) and 10% of the newly calculated belief of $0.75$.

It is clear that in this scenario, as the lower of the two alpha values increases, the rate at which belief can grow increases. Intuitively, this is because as this value increases, we are introducing more impact on the resultant belief value as a result of the newly calculated belief component. Furthermore, as the disparity between the previously-calculated belief and newly-calculated belief diminishes, the rate of growth begins to plateau. In practice, it will be important to carefully choose the optimal alpha value that provides sensible rates of belief growth without allowing for overly-dramatic spikes in belief that may be the result of malicious interference. As previously discussed, choosing this value is dependent on multiple factors and must be balanced with the risk assessment process to ensure that RTL thresholds are realistically attainable.

#### 3.2.2.2 Drop in Evidence

In the scenario visualised in Figure 3.3, we demonstrate the TAF's capability in handling a sudden drop in the ATL value due to receiving highly negative evidence during a stream of positive evidence. Between t=0 and t=3, the TAF receives the same evidence as used in the scenario discussed in Section 3.2.2.1;

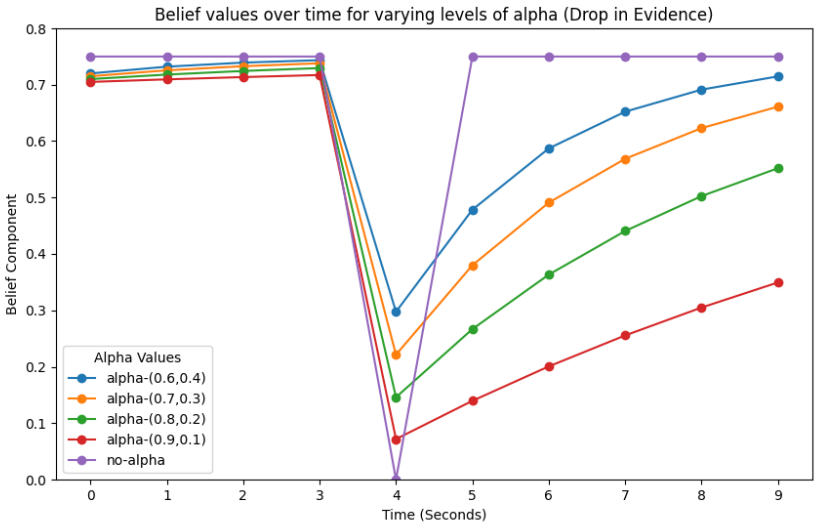



Figure 3.3: Belief over time for various values of $\alpha$ for a sudden drop in evidence

that is, fully positive evidence. As such, this region on the graph depict the same functionality as seen in the previous scenario.

However, at t=4, we inject a single occurrence of highly negative evidence. In this case, all six appraisals are set to zero, indicating absolute zero belief in the case that no historical data is considered. This behaviour is represented on the graph as the "no-alpha" line. Similarly, at t=5, we return to fully positive evidence (i.e. all six appraisals are set to a value of one). Again, when historical data is not considered, the belief instantly spikes back up to its maximum value of $0.75$, and remains at this value until the end of the experiment. In practice, this behaviour is likely undesirable; it is likely that the drop in evidence happened for a reason (such as a router-level fault resulting in untrustworthy data reporting, or even malicious intervention), and therefore the trustworthiness of the reporting router should be modulated for a sufficient time period to ensure it is back to functioning capacity.

Figure 3.3 additionally shows the results of the experiment conducted at four different levels of alpha. In this case, the chosen value of alpha has two main effects: the rate at which belief drops after an occurrence of negative evidence, and the rate at which belief can grow after such an event. For t=0 to t=3, we are moving to an equally-trustworthy state and so the lower of the alpha pair values is used (as in the previous scenario). However, at t=4, we transition to a less-trustworthy state. At this point, the higher of the alpha pair values is used, placing greater emphasis on the most recent observation. This approach allows us to accurately convey a steep drop in ATL value in the event of negative evidence, the severity of which is directly influenced by the chosen alpha value.

At t=5, we begin transitioning to a more-trustworthy state, thus reverting to the lower of the alpha pair values. We now return to the behaviour seen in Section 3.2.2.1, where greater emphasis is placed on the historical observation. As a result, belief does not spike back to its original value but instead grows conservatively with respect to the last observation.

#### 3.2.2.3 Rise in Evidence

Figure 3.4 demonstrates the TAF's capability to handle a sudden and unexpected rise in ATL due to receiving highly positive evidence. For t=0 to t=3, the TAF receives evidence with three positive and three negative appraisals. Without consideration of historical data, this results in a belief of $0.6$; clearly, this value is lower than the initial belief state of $0.7$, thus we observe a gradual decline in belief in the cases when an alpha value has been implemented. Similarly to the scenario in Section 3.2.2.2, the value of

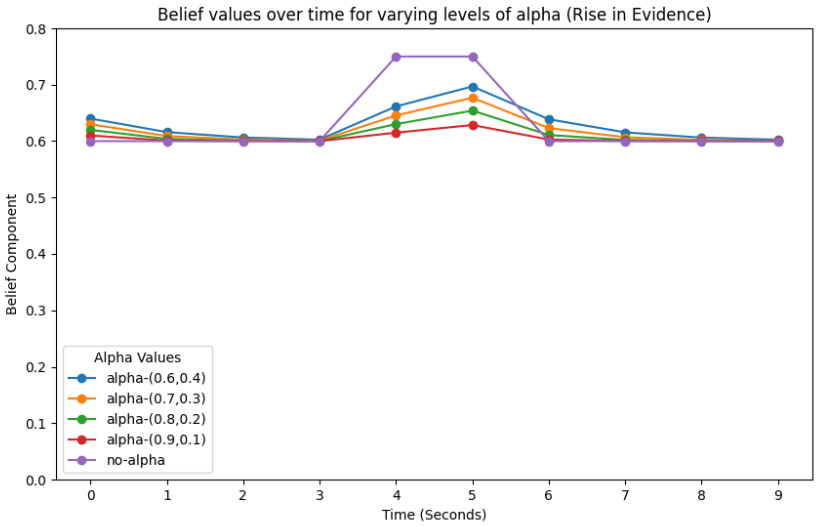



Figure 3.4: Belief over time for various values of $\alpha$ for a sudden rise in evidence

alpha directly impacts the rate at which belief decreases, although the drop-off is far less dramatic here as the lowest possible belief value with the provided evidence is $0.6$. During this time, the higher of the alpha pair values is used, as we are transitioning to a less-trustworthy state.

At t=4 and t=5, fully positive evidence is injected. As a result, we transition to a more-trustworthy state and the lower of the alpha pair values is used. Recall that we aim to minimise steep spikes in belief. Figure 3.2.2.3 shows how the chosen alpha value affects this rate of growth, with the alpha value pair of $(0.9, 0.1)$ resulting in the shallowest rate of growth and $(0.6, 0.4)$ resulting in the steepest. In any case, belief does not spike to the maximum value of $0.75$ as would be the case in which historical data is not considered, captured by the "no-alpha" line.

Finally, at t=6, we return to the original evidence reports, resulting in a transition to a less-trustworthy state (therefore using the higher of the alpha pair values), with belief gradually approaching the minimum of $0.6$ for the remainder of the experiment. Interestingly, this figure emphasises the importance of choosing the most appropriate alpha value. When an alpha value pair of $(0.9, 0.1)$ is used, minimal deviations can be observed across the entire experimental run despite major changes in evidence. In practice, this behaviour may result in any potential trustworthiness information from significant changes in evidence quality across short time-periods to be lost. Arguably, this is desired behaviour in this specific scenario due to the subsequent return to mixed evidence from t=6. However, in the real world, there are no guarantees as to the quality of future evidence, and an overly-restrictive alpha value may result in it being mathematically impossible for an ATL value to reach an RTL threshold within an acceptable time-frame (or even at all). This concept is discussed further in Section 3.5.

## 3.3 Evaluation 2: Evaluating Trust Transitivity at the Global TAF

The evaluation presented in Section 3.2 addresses the foundational challenge of knowledge-based trust evaluation, moving well beyond instant assessments that rely on a single piece of evidence. This challenge is universally applicable to all types of TAF instances—whether a central Global TAF or a Local TAF agent—and remains highly relevant across both the standalone and federated CASTOR TAF modalities.

The remaining TAF evaluations in this deliverable focus strictly on federated TAF concepts (i.e., historical opinions are not taken into consideration in the remaining evaluations), which constitute the core innovation of the CASTOR Trust Engineering process. In multi-agent systems, a primary challenge lies in

seamlessly consolidating trust evaluations produced by disparate TAF agents. As specified in D4.1 [6], exploring various trust relationships within the network traffic engineering domain — alongside their derived static and dynamic trust models (detailed in Section 2.3) — necessitates two critical trust operations: discounting and fusion.

In the context of the CASTOR architecture, discounting - the evaluation of which is discussed in this section - enables the Global TAF to dynamically adjust the weight of a Local TAF agent's report based on the central orchestrator's confidence in that specific agent. Conversely, fusion - whose evaluation is presented in Section 3.4 refers to the process where the Global TAF consolidates multiple independent trust evaluations from various Local TAF agents regarding a single, common trust proposition into one unified, consensus-driven assessment.

### 3.3.1 Evaluation Setup

The primary goal of this evaluation is to assess the impact of established Subjective Logic (SL) discounting operators on the final Assessed Trust Level (ATL) calculation. Within the CASTOR framework, a typical trust model resembles the dynamic topology presented in Figure 2.3a. In this scenario, the Global TAF maintains a direct trust relationship with a Local TAF agent, which in turn performs local assessments on the trust propositions of a single vRouter. This constitutes a textbook transitive trust scenario: the Global TAF must systematically "adjust" (discount) its perception of the monitored trust propositions based on its underlying confidence in the reporting Local TAF agent.

In order to evaluate the discounting operators in different scenarios, we consider two dimensions of experimentation: i) the Local TAF reports on a specific proposition, and ii) the Global TAF confidence on the Local TAF agent capabilities to make such trust evaluations.

Regarding the first dimension, Figure 3.5 presents the evolution of the trust opinion that is reported by the Local TAF agent. Specifically, we observe that at t=2 seconds, the Local TAF agent starts by reporting a rather confident trust opinion $\omega_x^{LTAF} = (0.9, 0, 0.1)$. Subsequently, the reported opinion becomes uncertain by reducing the belief value by 0.2. At t=6 seconds, the Local TAF opinion on the evaluated proposition becomes even more uncertain with an uncertainty factor of 0.8. Finally, we also include the case where even though the belief factor remains the same, at t = 8 seconds, the Local TAF agent reports an opinion with a non-negligible disbelief component set to 0.6. Through this final case, we want to capture the behaviour of the selected SL operators at the Global TAF level when, apart from uncertainty, there is also an indication of disbelief in proposition $x$. In the case of an honest Local TAF agent, the former case would imply that there could be gap in the frequency of reported evidence at the Trust Source Manager, whereas in the latter case the processed trustworthiness evidence provide observable indications against the target trust proposition $x$.

Section 2.4.4.2 presents three distinct SL discounting operators that are well-defined in the literature. By definition - and by inspecting the equations 2.1 and 2.2 - it appears that these two operators aim to achieve a similar behaviour: increasing the level of uncertainty in the Global TAF evaluation in the cases where the Local TAF agent is misbehaving. However, the third discounting operator, the "Opposite Belief Favouring", suggests that a misbehaving Local TAF agent will always report false statements. In what follows, we focus on these two families of discounting operator: the "uncertainty favouring" vs. the "opposite belief favouring".

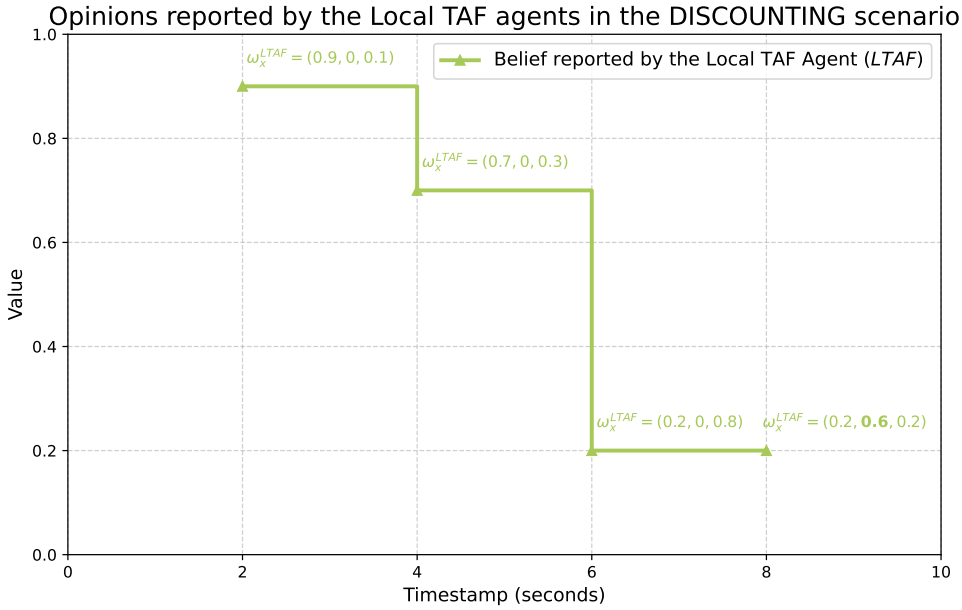



Figure 3.5: Discounting Evaluation Scenario description

## 3.3.2 Selecting the appropriate Discounting Operator

### 3.3.2.1 Fully Trusted Local TAF Agent

As illustrated in Figure 3.6, this baseline evaluation tracks the gradual decrease in the Local TAF agent's reported opinion over time. Following our established setup, the final two evaluation points (at t=6 and t=8) share an identical belief component. However, they differ significantly in their other dimensions: at t=8, there is a sharp drop in the uncertainty factor and a corresponding spike in disbelief, directly reflecting the concrete negative evidence the Local TAF agent has actively observed regarding proposition $x$.

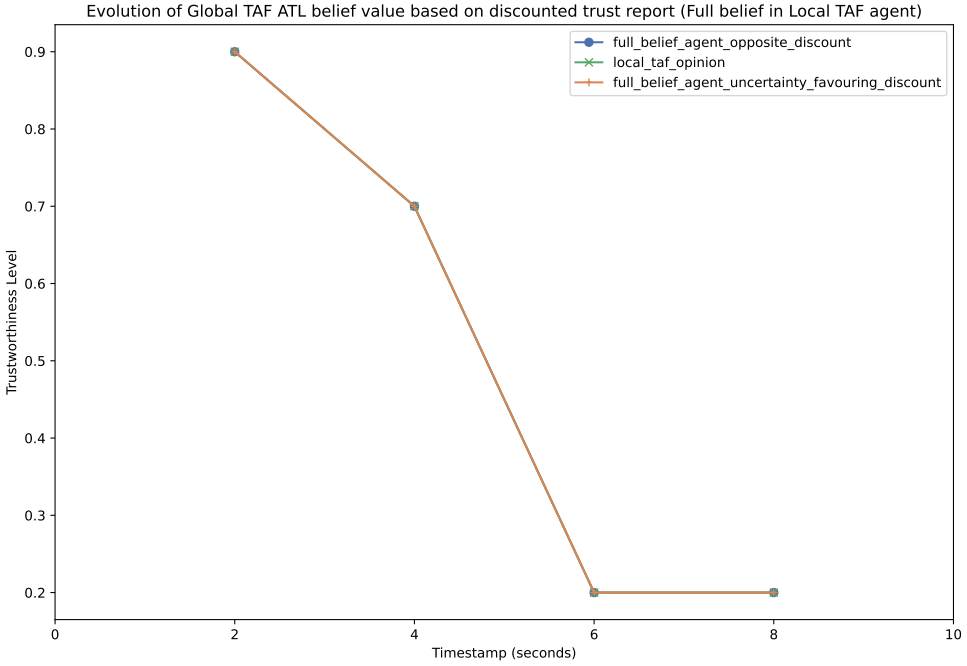



Figure 3.6: Discounted ATL value from a Fully Trusted Local TAF Agent

Because the Local TAF agent is considered fully trusted in this scenario, the Global TAF essentially accepts these recommendations blindly. Consequently, the discounted belief calculated by the Global TAF

is completely identical to the Local TAF agent's raw report. Whether applying the uncertainty-favouring or the opposite-belief-favouring operator, the Global TAF's ATL perfectly mirrors the local evaluations throughout the entire timeline, as no penalization or discounting is mathematically triggered.

#### 3.3.2.2 Non-negligible level of uncertainty in the Local TAF Agent

In the second scenario, depicted in Figure 3.7, the Local TAF agent is no longer viewed as fully trusted, but rather trusted with a specific level of uncertainty. Because of this structural doubt, both discounting operators successfully reduce the resulting Global TAF belief. The inherent uncertainty regarding the Local TAF's reliability is accurately absorbed and reflected in the overall ATL belief value reported by the Global TAF, although the general downward trend remains consistent with the first experiment.

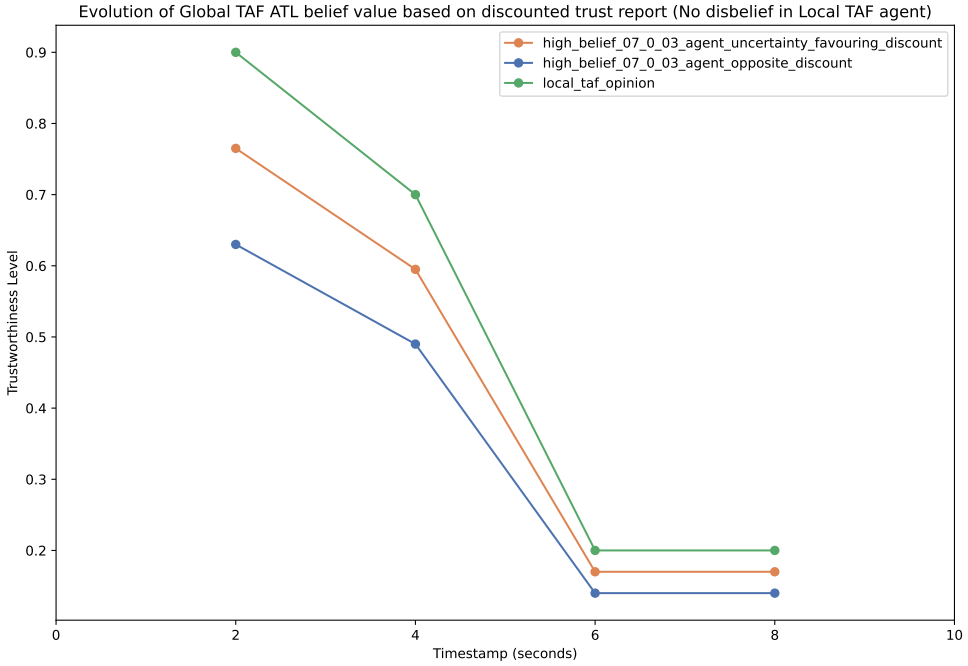



Figure 3.7: Discounted ATL value from a Local TAF Agent with a level of uncertainty

When comparing the two discounting strategies, the opposite-belief-favouring operator demonstrates a noticeably stricter penalization. While the uncertainty-favouring operator gently degrades the score, the opposite-belief operator aggressively pushes the initial high-confidence belief value down from 0.9 to a much lower 0.6. This highlights its highly conservative mathematical approach when forced to rely on uncertain reporting agents.

#### 3.3.2.3 Non-negligible disbelief in the Local TAF Agent

In this final evaluation, shown in Figure 3.8, the Global TAF actively introduces a level of disbelief (set to 0.2) regarding the Local TAF agent's fundamental capability to make accurate observations on trust proposition $x$. This foundational distrust from the central orchestrator, combined with the ongoing drop in the Local TAF's own reported belief, drives a much steeper and more aggressive reduction in the Global TAF's ATL belief value compared to the previous experiments.

However, this high level of distrust leads to a highly counter-intuitive scenario at t=8. When the Local TAF reports strong negative statements (high disbelief), the Global TAF's ATL belief on proposition $x$ actually begins to increase, despite the accumulation of negative evidence. This occurs due to the fundamental design of the opposite-belief-favouring operator, which assumes that a compromised or strictly unreliable agent will consistently report false statements. Therefore, when the highly distrusted Local TAF

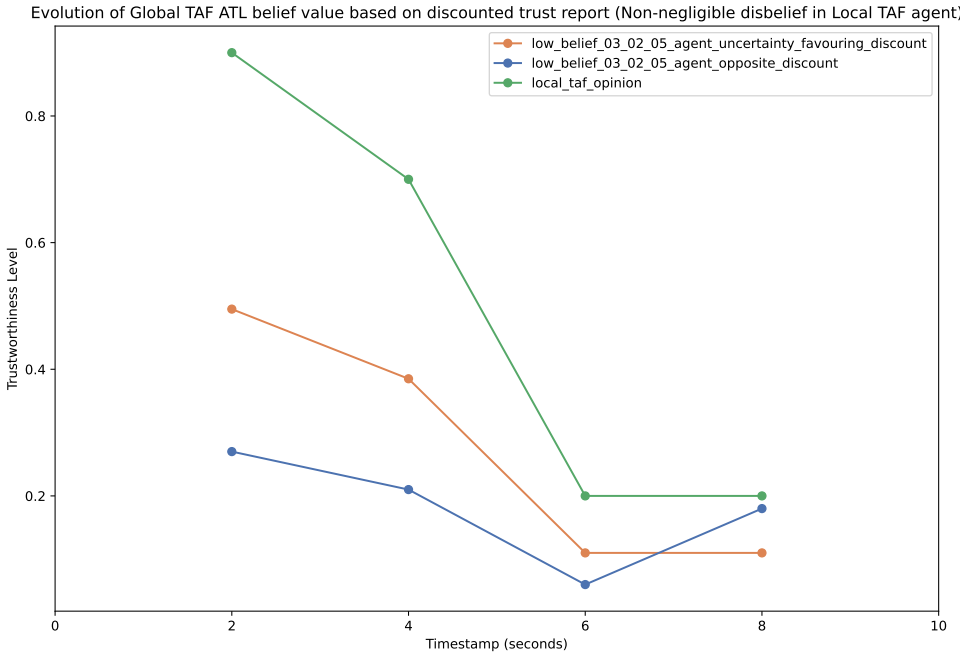



Figure 3.8: Discounted ATL value from a Local TAF Agent with a level of uncertainty and disbelief

asserts that proposition $x$ is malicious or untrustworthy, the Global TAF mathematically inverts this logic, concluding with a specific level of certainty that proposition $x$ is actually trustworthy.

## 3.4 Evaluation 3: Evaluating Trust Fusion at the Global TAF

Building upon the foundational single-agent topology explored in the discounting evaluation (see Section 3.3), our focus now shifts to assessing the Subjective Logic fusion operator. In this evaluation setup, the Global TAF acts as the central consolidator for two distinct Local TAF agents, both of which are concurrently reporting trust opinions regarding a single, common trust proposition (e.g., the behaviour of the same target vRouter). Unlike the previous discounting scenario, which tested the vertical adjustment of a single agent's report based on the Global TAF's confidence in that agent, this experiment evaluates the horizontal synthesis of multiple independent Local TAF perspectives on the same proposition $x$. The primary objective is to observe how the Global TAF mathematically merges these parallel reports to produce a singular, consensus-driven Assessed Trust Level (ATL) for the target proposition.

This specific topological setup is fundamentally driven by the concept of referral trust, as specified in D4.1 [6]. During the CASTOR proactive phase, a new TNDI executes the Secure On-boarding protocol through its TNDE, as detailed in D3.2 [10]. Within this process, there is a brief timeframe where the new TNDI (having just attested its TNDE platform) can exchange attestation evidence with adjacent nodes. This exchange in form of Stamped Passports allows the establishment of trusted connections with next-hop peers, aligning with the IETF's Trusted Path Routing paradigm [3]. Consequently, neighboring, already on-boarded TNDIs have the opportunity to evaluate the trustworthiness of the newly arriving node. These adjacent nodes can formulate a preliminary trust opinion and forward it to the Global TAF even before the new node's TNDE is fully configured (i.e., allowing the Local TAF agent to join the TAF federation).

This scenario leads to a trust model as the one depicted in Figure 3.9. In this context, a direct (functional) trust relationship between the Global TAF and every participating entity — such as Router N — cannot be guaranteed throughout the Router lifecycle. Instead, the Global TAF must rely on adjacent, previously authenticated nodes to locally gather evidence, such as device integrity or attestation reports, and formulate initial trust opinions regarding the new TNDI. By collecting these indirect, neighbour-generated assessments and fusing them, the Global TAF can robustly evaluate target propositions even in the absence of

direct observation, securely extending the framework's trust visibility across complex and evolving network topologies.

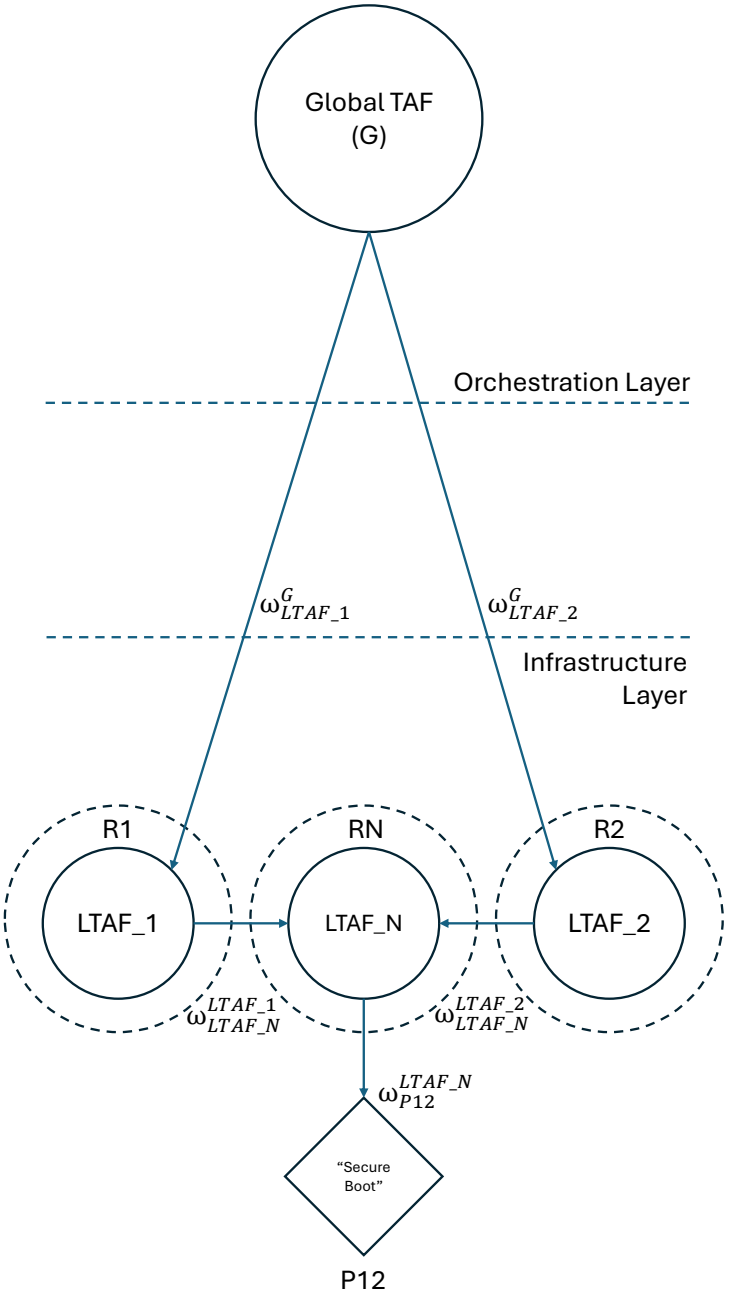



Figure 3.9: Trust Model Instance at the Global TAF. It captures Router 1 and Router 2 forming an opinion on the onboarding router, Router N, and sharing their local assessment with the Global TAF

### 3.4.1 Evaluation Setup

To systematically evaluate the fusion operator across various states of consensus, we apply the exact same time-series step function utilized in the discounting evaluation setup (see Section 3.3), but deploy it across the two Local TAF agents with a temporal offset. Specifically, the first Local TAF agent begins its reporting sequence at $t = 1$, while the second agent initiates the identical evolutionary pattern starting at $t = 2$. By employing an identical step function in the behaviour of each Local TAF agent belief on proposition $x$ (as shown in Figure 3.10), we envision to evaluate the fusion operator's performance as the system naturally progresses through three critical phases: periods where both agents share perfectly

aligned opinions, phases where their reports become misaligned due to differing levels of uncertainty, and ultimately, highly volatile windows where the Global TAF is forced to reconcile strictly conflicting observations.

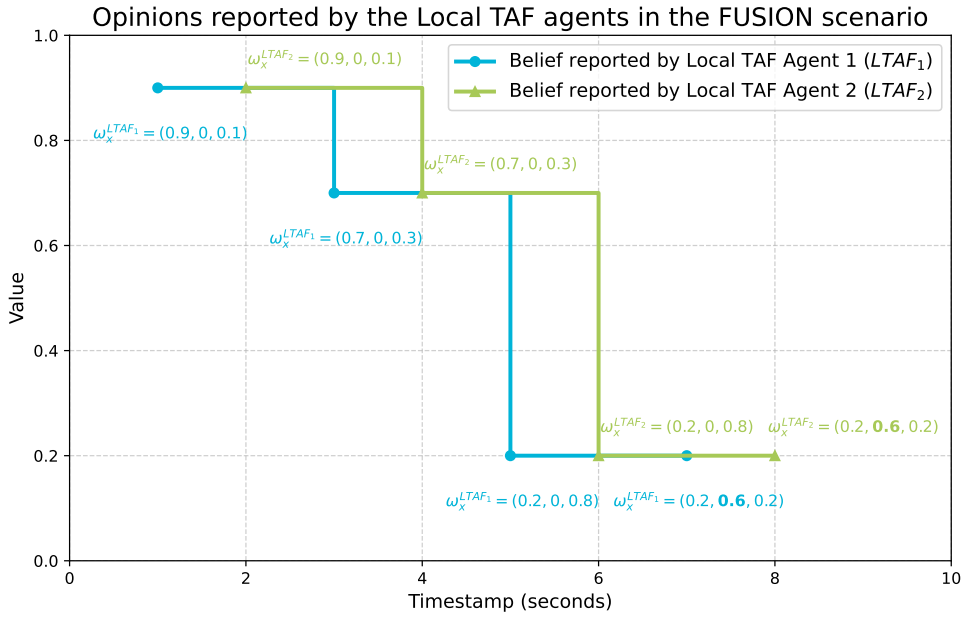



Figure 3.10: Fusion Evaluation Scenario description

As detailed in Section 2.4.4.2, the literature provides a rich taxonomy of Subjective Logic fusion operators designed for different mathematical contexts. For this evaluation, we focus exclusively on three core operators: Averaging, Cumulative, and Weighted Belief fusion. These three are fundamentally evidence-based operators that perfectly align with the CASTOR architecture, where Local TAF agents continuously aggregate observable network events. Specifically, Averaging and Cumulative fusion allow us to capture the baseline behaviours for dependent and independent evidence gathering, respectively, while the Weighted Belief operator dynamically adjusts the synthesis based on the relative certainty of the two reporting agents.

In future evaluations, the goal is to enhance the assessment with additional SL fusion operators that address higher-scale network complexities. Most notably, one such operator is the Consensus & Compromise [15] Fusion operator, which can be employed in scenarios where more than two Local TAF agents are concurrently sharing opinions on a common target. This specific mechanism becomes extremely relevant when assessing highly connected network elements (e.g., a central ”hub node” in the routing topology) where consolidating a large multitude of diverse or conflicting perspectives requires a sophisticated, compromise-driven mathematical approach.

## 3.4.2 Selecting the appropriate Fusion Operator

### 3.4.2.1 Fully Trusted Local TAF Agents

As illustrated in Figure 3.11, the observations for this fusion evaluation are intentionally displayed starting from $t = 2$, despite the first ATL belief value being available at $t = 1$. This is because the primary focus of this experiment is the ATL resulting from a fusion operation; at $t = 1$, the Global TAF simply derives its ATL by discounting the sole available opinion from the first Local TAF agent. Once both agents are reporting, the synthesized opinions allow the Global TAF to track the expected gradual decrease in the final ATL belief value. Consistent with the evaluation setup, the final two evaluations share a common belief component at $t = 6$ and $t = 8$, but the state at $t = 8$ is uniquely characterized by a sharp drop in

the uncertainty factor and a corresponding increase in disbelief, directly reflecting the concrete negative observations made by the Local TAF agents.

When analyzing the specific operators, the cumulative fusion operator inherently produces higher belief values (peaking at $0.93$) that actually exceed the maximum local opinions provided by the individual agents ($0.9$). This mathematical boost occurs because two entirely independent agents are providing highly confident opinions, allowing the Global TAF to significantly increase its overall certainty regarding the statement. In contrast, the averaging and weighted fusion operators do not accumulate this confidence, but rather find the mathematical middle ground. When the two local reports are perfectly aligned, these two operators yield identical results. However, when discrepancies arise (e.g., at $t = 3$ and $t = 5$), the weighted fusion operator diverges by assigning more significance to the most confident, lowest-uncertainty opinion. For instance, at $t = 5$ (see Figure 3.10), Local TAF agent 2 reports a relatively confident opinion of $\omega_x^{LTAF_2} = (0.7, 0.0, 0.3)$, whereas agent 1 reports a highly uncertain opinion of $\omega_x^{LTAF_1} = (0.2, 0.0, 0.8)$. In this case, the weighted operator rightly pulls the final ATL closer to Local TAF Agent 2's evaluation.

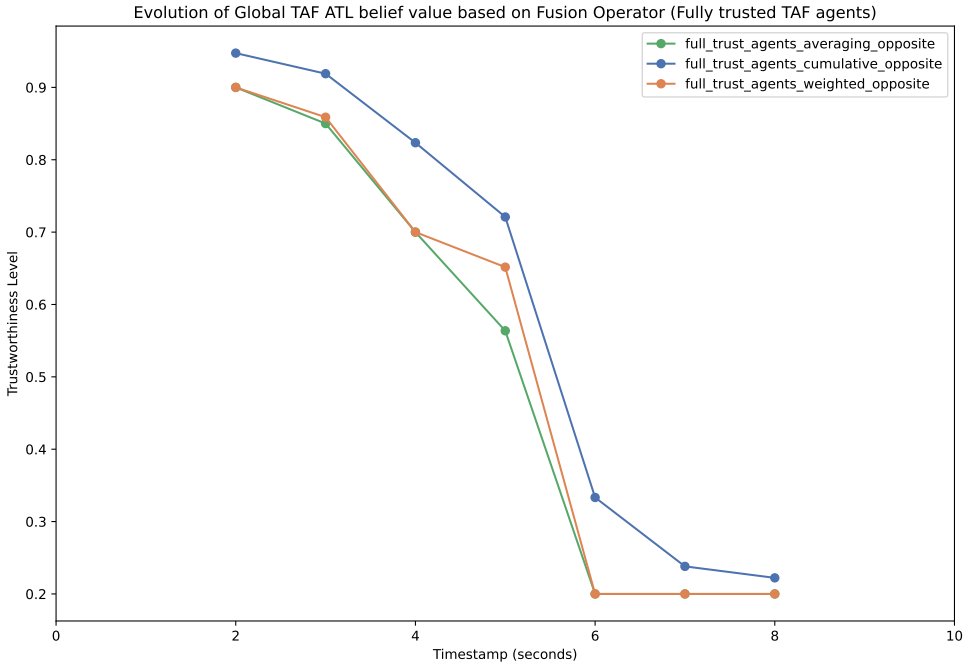



Figure 3.11: Fused ATL value from Fully Trusted Agents

#### 3.4.2.2 Equally Trustworthy Local TAF Agents

In the second scenario, shown in Figure 3.12, neither Local TAF agent is considered fully trusted; instead, the Global TAF relies on their assessments with a non-negligible, equal level of uncertainty (set to $0.3$). Because the foundation of the trust reports is now inherently uncertain from the orchestrator's perspective, the fusion results are universally dampened. Most notably, we observe that even the historically aggressive cumulative fusion operator is no longer sufficient to allow the Global TAF to reach the locally reported belief peak of $0.9$. The orchestrator's baseline doubt prevents the accumulated evidence from fully restoring the maximum trust score.

Despite this overall reduction, the behavioural dynamics between the averaging and weighted fusion operators remain highly consistent with the first experiment. The magnitude of the discrepancy between these two operators remains directly tied to the difference in confidence between the local reports. For example, at $t = 3$, the difference in the uncertainty factor between the two agents is relatively minor ($0.2$, where agent 2 has an uncertainty of $0.1$ and agent 1 has $0.3$). This leads to a correspondingly small

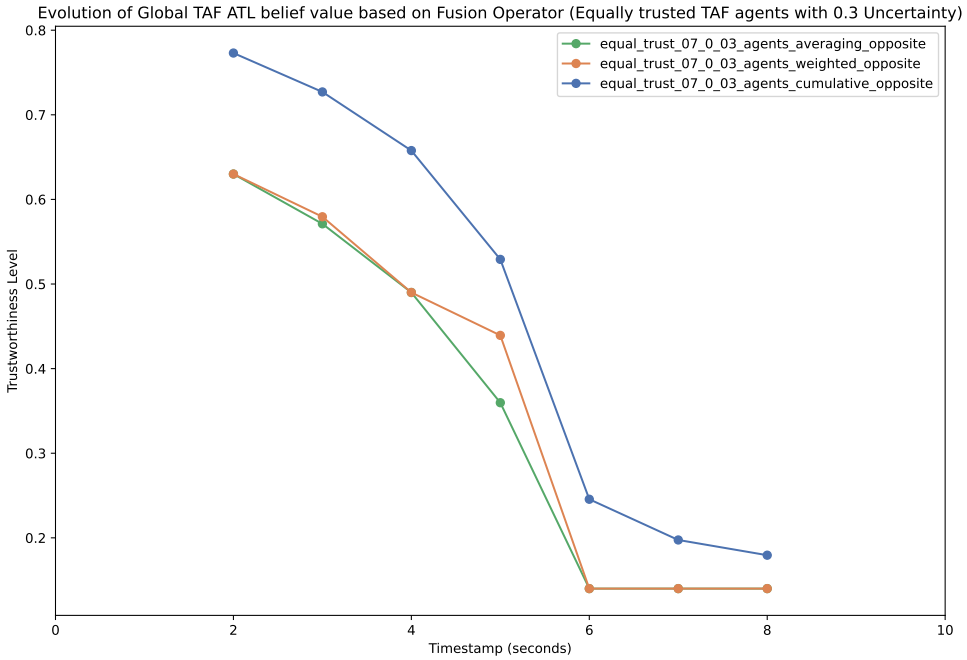



Figure 3.12: Fused ATL value from Fully Trusted Agents

difference between the averaging and weighted fusion outputs. However, as the gap in local uncertainty widens to $0.5$ in later timestamps, the weighted operator pulls significantly further away from the strict mathematical average.

#### 3.4.2.3 Unequally Trustworthy Local TAF Agents

In the final evaluation, depicted in Figure 3.13, the Global TAF assigns different levels of trustworthiness to the two reporting agents. While Local TAF agent 1 remains at the same moderate level of trustworthiness used in the previous experiment, Local TAF agent 2 suffers from a significantly reduced reputation, with the Global TAF holding an opinion of $\omega^{Global\_TAF}_{Local\_TAF_2} = (0.3, 0.2, 0.5)$ regarding its reliability. Consequently, the nominal belief value of the final ATL is severely reduced across the board. The two averaging-based operators start with a much lower initial belief value of approximately $0.5$, while the cumulative fusion operator only manages to boost that baseline by $+0.1$.

This fundamental difference in agent trustworthiness has a profound and continuous impact on the relationship between the averaging and weighted fusion operators. Unlike previous scenarios where the lines occasionally converged, the unequal trust weights cause the two operator trajectories to remain consistently separated throughout the timeline, with the standard averaging operator acting as the more conservative metric (yielding lower belief values at all times). Finally, we observe a sudden, counter-intuitive incline in the belief values across all operators after $t = 6$. As detailed in Section 3.3, this spike is a direct consequence of employing the opposite-belief discounting operator prior to fusion: because the highly distrusted Local TAF agent 2 begins reporting heavy negative evidence, the Global TAF logically inverts that assessment, ultimately fusing it as a mathematically positive contribution to the final ATL.

## 3.5 Towards the evaluation in the CASTOR integrated framework

In this summary, we explore the ways in which the TAF functionality detailed throughout this chapter will be beneficial to the wider scope of the CASTOR project. We also consider the impact that the Risk

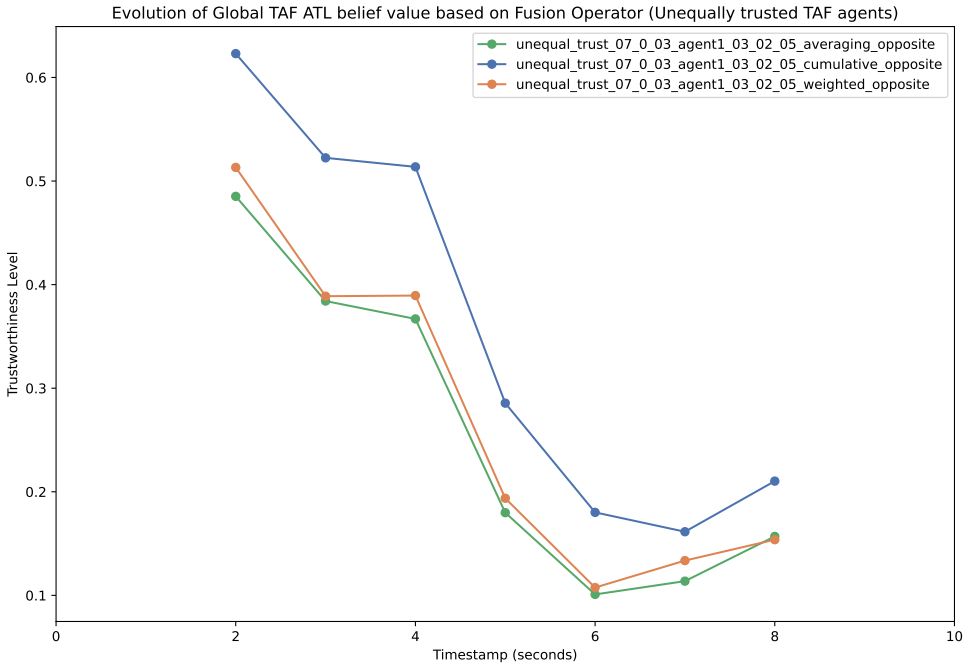



Figure 3.13: Fused ATL value from Non-equally trustworthy Local TAF Agents

Assessment process has on the optimisation of trust models, before finally detailing further functionality planned for the second CASTOR release and PoC evaluation.

**Impact of RTL Methodology on Evidence Collection and Trust Model Fine-Tuning.** As discussed throughout this chapter, both the evidence collection process as well as the optimisation of a trust model's parameters is directly dependent on the underlying Risk Assessment (RA) methodology. For evidence-based ATL calculation, the RA directly influences both the type and frequency of collected evidence based on the offered security guarantees (and therefore the risks that are subsequently mitigated) that are to be used in the evaluation of trust for a specific trust property. Furthermore, the RA process also directly impacts the implementation of a trust model with respect to the fine-tuning of ATL values, for example through the optimisation of alpha values and the correct choice of discounting and fusion operators. This optimisation step must be carefully calibrated with the RA process to ensure that the trust decision process yields accurate ATLs that can realistically meet RTL constraints. The RA methodology is detailed in Chapter 4.

**Towards the PoC Evaluation.** Whilst the experiments detailed in this chapter have demonstrated correct TAF functionality within an experimental environment, a primary goal is to evaluate the federated trust assessment concept within a realistic vRouter topology as detailed in D6.1 [9]. The work presented in D4.2 demonstrates the ability to fine-tune a given trust model at the Local and Global TAF levels, with respect to the Trust Policies that will be provided as part of the PoC evaluation. The PoC evaluation itself aims to verify the functionality of the end-to-end trust assessment process – from the collection of evidence from the TNDE Trust Sources to the local and global trust evaluations themselves – including correct functionality of the capabilities introduced throughout this chapter. In addition, we would like to verify the TAF's ability to compose composite trust propositions (as opposed to the atomic trust propositions explored in this chapter), such as to allow for the evaluation of link and path-level trust as well as support advanced traffic engineering processes, as discussed in Section 2.4.4.1.

# Chapter 4

# Risk Assessment Engine

## 4.1 Overview

Deliverable D4.1 [6] introduces the importance of a robust risk assessment process in the context of the overarching CASTOR Trust Assessment Framework. Starting from the CASTOR preparedness phase (defined in D2.1 [8]), the risk analysis of the forwarding plane serves as the foundation of the trust engineering process, contributing to the identification of the appropriate trustworthiness evidence that need to be monitored throughout the operational lifecycle of the network elements in order to attain to a specific trust posture. In the context of network traffic engineering, a behaviour can be a function of a network element's critical control plane logic, such as a router accurately computing and propagating path metrics (network- and trust-related ones) for a specific Segment Routing Flexible Algorithm (Flex-Algo) to guarantee constraint-based traffic steering. The trustworthiness evidence that is collected from the assets of the target system is directly linked with the security mechanisms representing the assets' capabilities. In the context of CASTOR, such evidence may come from different Trust Sources which are part of the deployed TNDE platform, as presented in D3.2 [10]. The presence of these capabilities is beneficial for the evaluation of trustworthiness with respect to specific trust properties (i.e., characteristics such as security, availability, resilience, and robustness).

Based on the high-level trust assessment framework detailed in D4.1 [6], the CASTOR TAF evaluates the required level of trustworthiness of a particular trust object (e.g., a network node) to eventually compute a Required Trust Level (RTL) for a specific network service or topology. Parallel to this, using the trustworthiness evidence collected from various trust sources during runtime, the TAF calculates the Actual Trust Level (ATL) that characterizes a network element at any given moment. A crucial enabler bridging these two trust calculations is the identification of the prominent threats and risks affecting the target trust properties. Hence, a comprehensive risk assessment framework serves as the foundational common denominator: it captures attack vectors across the network continuum and evaluates their impact. For instance, the inherent risk profile and operational exposure of an asset dictate the stringency of its RTL, while the real-time manifestation of active risks dynamically lowers its ATL.

However, treating the risk profile of each network asset as an isolated entity, relying solely on intrinsic vulnerability scores or disconnected external threat intelligence, is insufficient for an accurate RTL calculation. Failing to contextualize an asset within its operational environment can yield an unattainable or inaccurate RTL, which ultimately hinders the overarching trust assessment process and leads to inconsistent, flawed trust decisions. This limitation necessitates a topology-aware evaluation of risk, ensuring that the centrality and operational importance of a node are accurately reflected in the risk quantification and, consequently, the RTL methodology. Consider, for example, two routers in a network that suffer from the exact same critical vulnerability. In an isolated risk assessment, both would yield identical risk scores. Nevertheless, if Router A acts as a central hub node routing traffic for critical backend databases, and Router B acts as an isolated edge router for a guest network, their true risk profiles differ drasti-

cally. The compromise of Router A enables extensive lateral movement and systemic failure, demanding a significantly higher RTL than Router B. When existing risk assessment methodologies evaluate these network elements in isolation, they restrict their scope exclusively to direct attacks against a specific node or protocol. By doing so, they provide a dangerously incomplete (and often inaccurate) view of the network's true risk posture. However, this narrow perspective fails to account for aggravated, cascading attacks. If an adversary successfully breaches a less secure adjacent node (such as Router B) they can exploit that leverage to pivot internally. Without mapping these dependencies, the assessment may neglect that central, vital hub nodes are highly susceptible to attacks with a much higher likelihood when targeted via internal lateral movement. Consequently, failing to model these cascading multi-step attacks ensures that central nodes are assigned RTLs that do not reflect their true, elevated exposure to intra- and inter-domain threats.

### 4.1.1 Towards a topology-aware RTL methodology

In previous works [23], calculating a topology-aware Required Trust Level (RTL) relied on security administrators to exhaustively and manually list all potential attack paths and assess their execution likelihood. As modern network topologies become increasingly complex (both in terms of graph scale and dynamic runtime behaviour) this manual approach has become severely cumbersome and unscalable. To overcome these limitations, in the specification of Engineering Story-II in D4.1 [6], we outline the requirement of modelling the cascading attack path calculation problem as a Markov Decision Process which would allow us to provide an RTL methodology with an accurate representation of the topology's risk posture.

We present the first version of the CASTOR Risk Assessment Engine (RAE). In this initial release, the CASTOR RAE performs a complete risk analysis iteration, including the derivation of context-specific RTL calculations. While this first implementation calculates the RTL in a primarily isolated manner (i.e., not adjusting the attack likelihood of a node based on cascading effects), it establishes the core framework for automated attack path analysis. Specifically, this chapter presents the core dimensions (see Section 4.3) that form the basis for the probabilistic modelling of attack path realization. We detail how simulating various "what-if" scenarios through Monte Carlo sampling can automatically estimate the likelihood of discovered single- and multi-step attack paths across the router topology, revealing how attack paths cross-impact one another. This establishes a foundation to systematically model the state transitions a threat actor may trigger, allowing us to evaluate both the baseline transition probabilities of an attack step's success and the probabilistic degree of belief regarding the resulting state.

Regarding the latter dimension and looking forward to the second release, our research and experimentation shifts toward systematically modelling the dependencies among possible exploits using Bayesian Logic to capture the precise conditional probabilities of cascading attacks: *if Router A is compromised, what is the exact probability that Router B is successfully exploited next, and what is our degree of belief regarding Router B's resulting state, thereby allowing the attack path analysis to progress toward the final target endpoint?* By coupling this probabilistic threat model with a Markov Decision Process, we aim to formulate an automated, stochastic decision-making framework that will allow the CASTOR RAE to mathematically model the probability of an attack path being activated and - through the revised RTL methodology in the second version - the systemic cost incurred by the security administrator to deploy the necessary mitigation guarantees.

The remainder of this chapter presents the system overview, placing a particular focus on these attack path calculation capabilities. We first expand on the key dimensions that influence the realization of an attack step, which form the foundation of our probabilistic decision process. Embedded within the Monte Carlo simulation, this process calculates all possible attack paths and their respective likelihoods. Looking ahead to the engine's second release, we also outline how this iterative sampling could extend beyond mere path evaluation to systematically refine the underlying transition matrices. This ultimately yields contextually adjusted risk levels for all assets in the topology, providing the basis for accurate,

dynamically generated RTL values. Finally, the section concludes with a straightforward case study scenario evaluating the proposed RTL methodology. This scenario highlights the tangible impact of elevating risk assessment from isolated, device-level calculations to topology-aware evaluations. Furthermore, we demonstrate how the introduction of security controls mitigates cascading topology-aware risk. This step is critical, as it aligns the risk model with runtime monitoring by identifying the exact pieces of evidence that must be collected to model the ATL, thereby enabling semantically aligned and accurate trust decisions.

## 4.2 System Overview

This section describes the system modelling basis for the advanced RTL generation mechanisms developed within CASTOR. In addition to the architectural foundations described in D4.1 [6], the model of the system defines a precise mathematical representation of the network infrastructure, threat landscape, attacker's abilities, and security measures. This abstraction will serve as the common input source for each of the analytical functions of the CASTOR Risk Assessment Engine (RAE), as well as the Monte Carlo sampling process presented in Section 4.4.

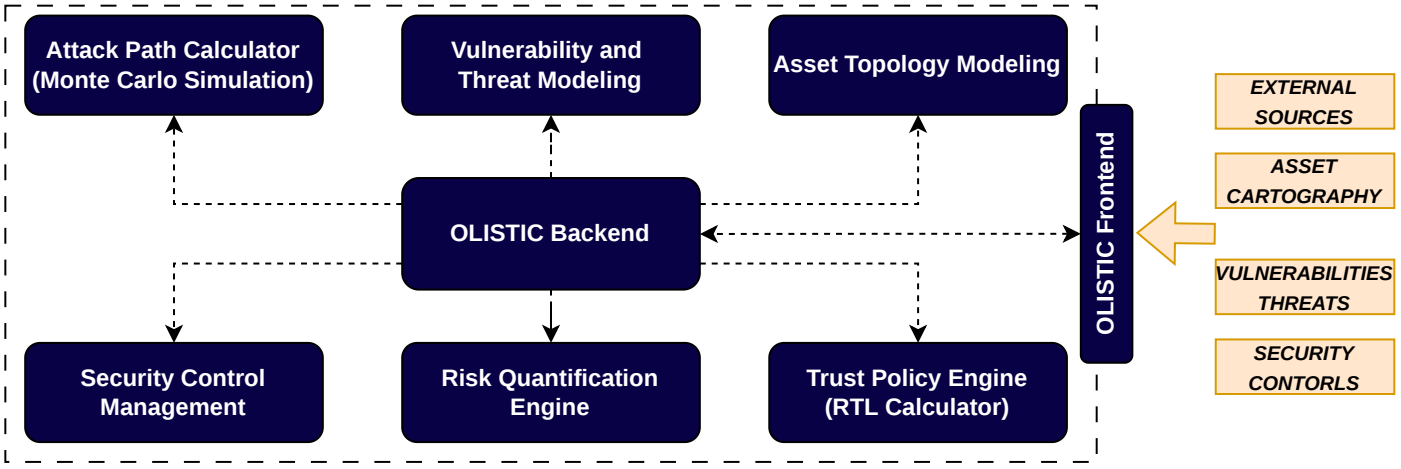



Figure 4.1: CASTOR Risk Assessment Engine

### 4.2.1 Internal Architecture

As illustrated in Figure 4.1, the internal architecture of the CASTOR Risk Assessment Engine consists of several subcomponents. From a functional standpoint, the execution of a new risk assessment task is initiated via a request to the CASTOR backend. This process can be triggered either manually (e.g., through an action performed by a security administrator) or automatically (e.g., via periodic scheduling or event-driven network telemetry). The output is a comprehensive representation of the risk report - such as a topology-aware risk graph or a detailed table with risk results per asset - shared with the intended entities, namely the security administrator and other authorized stakeholders (e.g., Domain Operators). The CASTOR Risk Assessment Engine follows a modular architecture, allowing the incorporation of multiple risk assessment methodologies (focusing on isolated, or topology-aware evaluations) that could eventually impact the final Trust Policy to be employed in the CASTOR Trust Assessment Framework.

To enable the execution of these risk methodologies, it is assumed that a continuous and thorough threat analysis is performed over the monitored 6G network infrastructure. This incorporates information regarding vulnerabilities and threats sourced from CASTOR's artifacts (e.g., risk indicators detected from

CASTOR TNDE [10]), external threat intelligence feeds (e.g., NVD [4]), or directly from the Domain Manager. The realization of the network cartography - that is, the comprehensive inventory of assets comprising the forwarding plane - is also required to accurately depict all network elements (e.g., infrastructure elements, vRouter software stack) and their interdependencies (e.g., physical links, logical connections, and dynamic data flows).

Finally, whether defined manually or through publicly available security advisories, it is essential to outline mitigation strategies. This enables the CASTOR RAE to assess the effectiveness of specific security controls that can be dynamically enforced across the network topology. Collectively, these functionalities facilitate the derivation of context-aware mechanisms to assess the Actual Trust Level (ATL) that each network node must maintain during runtime to be considered trusted. The remainder of this section briefly presents each of the internal components.

#### 4.2.1.1 OLISTIC Backend and Frontend

The OLISTIC backend offers the necessary APIs to orchestrate all the backend operations of the engine and works in synergy with the frontend to offer the necessary functionalities to the security analyst. The backend component is also responsible for performing the access control rules that allow only authorized users to use the available API. Finally, it provides the necessary interface for participating (i.e., publishing and subscribing) to Kafka topics that enable the asynchronous consumption of security incidents reported by other CASTOR components. The OLISTIC frontend offers an interactive dashboard which is used for visualizing the digital representation of the cyber-physical environment. Of course, this dashboard offers a plethora of operations that can be performed, ranging from simple asset addition/editing/deletion to the creation of attack scenarios, management of vulnerability and threat profiles of assets, consideration of controls and mitigation actions, the execution of the risk assessment and many others.

#### 4.2.1.2 Asset Topology Modelling

This component is responsible for modelling the list of assets that comprise the monitored forwarding plane. We consider an asset any entity, tangible or intangible, that participates in the ecosystem and carries a level of risk. To address the first functional requirement, an abstract notion of the term asset is adopted to represent hardware assets (e.g., commodity router equipment or bare metal in the case of vRouter instances) or services running on top of them (e.g., Cisco operating system in the case of IOS XRD vRouters). Apart from a uniquely identifiable name in the topology, security administrators can provide additional information such as the level of criticality of each asset in the domain (i.e., business value). In addition, it is possible to provide the Common Product Enumeration (CPE) identifier, if applicable. This facilitates the seamless and automated association of the assets with vulnerabilities and threats. The CASTOR RA Engine also allows for an arbitrary set of attributes in a key-value format.

By treating each network element as a System-of-Systems, this subcomponent captures all asset interdependencies both within and beyond the boundaries of a single vRouter. This enables the modeling of diverse relationships, ranging from physical network connections to software dependencies and data flows. A preliminary, non-exhaustive list of these asset relationships is presented below:

- *IsConnectedTo*: to express network connections between hardware or virtual assets (e.g., Hardware Router A is connected to Hardware Router B via an intermediate distribution switch),
- *IsUsedBy*: to express logical dependency among assets (e.g., a Segment Routing policy is used by the vRouter's forwarding engine to steer traffic),
- *IsProcessedBy*: to express the fact that a piece of data is accessed by an asset (e.g., incoming BGP UPDATE messages are processed by the vRouter's control plane to compute routing tables),

- *isLocatedIn*: to express geospatial dependency among assets (e.g., the commodity hardware server hosting the vRouter is located in Rack 12 of the data center),
- *isStoredOn*: to express the fact that a piece of data is stored on an asset (e.g., the Forwarding Information Base (FIB) is stored in the vRouter's RAM),
- *isInstalledOn*: to express a dependency between software and hardware assets (e.g., the vRouter software is installed on the x86 bare-metal server).

The CASTOR RA Engine offers a graphical user interface based on which the security administrator is able to manage the asset cartography. Firstly, the tool supports the visualization of the data flow diagram for each function and the inspection of the list of assets defined in the monitored infrastructure. Finally, to support the steps of the Risk Assessment process, the Asset Modelling and Visualization component, also, accommodates for the association of assets with specific damage scenarios and their impact in the context of specific trust properties.

#### 4.2.1.3 Vulnerability and Threat Modelling Component

The association of assets with their vulnerabilities is crucial for conducting a risk analysis and determining the required risk level for a given context. To enable this process, the Vulnerability and Threat Modelling component provides the means to store and manage a knowledge base comprising of well-known vulnerabilities. On the one hand, the component supports the automated and periodic synchronization with publicly available repositories such as the National Vulnerability Database (NVD) [4]. These repositories provide additional information that enables the calculation of the impact of a vulnerability based on the Common Vulnerability Score System specification (CVSS) [24]. In addition, this component leverages other cybersecurity awareness initiatives that focus on the association of specific vulnerabilities with common attack patterns (CAPEC [18]), weakness (CWE [19]), and product catalogues (CPE [20]) that enable the seamless association of vulnerabilities with specific product identifiers.

#### 4.2.1.4 Risk Quantification Engine

The CASTOR Risk Quantification Engine calculates the Individual Risk Level (IRL) for specific network assets by evaluating distinct threats and vulnerabilities. Grounded in the NIST SP 800-30 [21] risk management lifecycle, the engine generates a measurable risk score by combining a subjective threat likelihood with a standardized, objective vulnerability impact assessment[1].

The methodology is built upon two primary pillars:

- **Threat Probability:** A subjective, semi-quantitative metric determined by a security administrator. It assesses the likelihood of an adversary initiating a specific threat event based on the network's inherent characteristics, ranging qualitatively from "Very Low" to "Very High".
- **Vulnerability Impact:** A systematic evaluation of a vulnerability's severity using the globally recognized Common Vulnerability Scoring System (CVSS) [24] developed by FIRST [12]. It calculates impact using three metric groups: the mandatory *Base Metrics* (capturing fundamental features like exploitability and impact on the CIA triad), and the optional *Temporal* and *Environmental Metrics* (adjusting the score based on time-dependent factors, such as exploit code maturity, and unique user infrastructure requirements).

[1] While this initial implementation relies on the domain-agnostic NIST SP 800-30 [21] and CVSS [24] frameworks, risk quantification models can vary significantly across different network operators and domains. Designed with modularity in mind, the CASTOR RAE ultimately serves as a harmonization layer capable of integrating diverse risk methodologies. This ensures a transparent, extensible, and robust risk analysis pipeline to feed subsequent trust calculations.

Calculating the Individual Risk Level (IRL) By multiplying the subjective threat likelihood by the objective CVSS impact score, the engine computes the quantitative risk for a specific asset-vulnerability-threat combination:

$$IRL(Asset_i, Vulnerability_j, Threat_k) = ThreatLevel_{Threat_k} \times VulnerabilityImpact_{Vulnerability_i}$$

This yields a quantifiable risk score that categorizes the individual risk level in a five-tier scale: ”Very Low (VL)”, ”Low (L)”, ”Medium (M)”, ”High (H)”, ”Very High (VH)”. Because network contexts vary significantly, the engine leaves the final aggregation of these individual risks (i.e., combining multiple IRLs into a single cumulative risk score for an asset) to the discretion of the security administrator, who can apply methods such as taking the maximum IRL value or calculating a weighted average.

#### 4.2.1.5 Security Control Management Component

Two critical functional requirements of the CASTOR RA Engine refer to the management of the various security controls and their impact on the risk level within the monitored topology. In the context of CASTOR, there are three major types of security controls:

- *Implemented security controls:* This category includes security measures that are in place and configured correctly but may not provide real-time evidence of their effectiveness during runtime. Examples include encryption protocols like MACsec, secure boot, and secure configurations.
- *Implemented and executed security controls (i.e., Runtime attestation controls):* These controls not only include security measures that are implemented but also provide ongoing evidence of their execution and effectiveness during runtime. Examples include runtime attestation protocols, integrity measurement frameworks, and control flow attestation. These controls provide additional levels of assurance, allowing administrators to verify that security measures are actively enforced and responding as expected to potential threats.
- *Behavioural detection:* These controls are primarily focused on identifying and responding to anomalous or unauthorized behaviour within the vRouter environment. The CASTOR FSM Source constitutes such a security control which is able to alert administrators or trigger ATL recalculations when a deviation from the expected device model is detected.

The Security Control Management Component supports the modelling of all types of controls. It also allows security administrators to signal the effectiveness of each security control by configuring the various parameters that contribute to the overall risk level for an asset or a trust relationship. By combining multiple security controls for various assets, security administrators can define specific profiles, namely Mitigation Strategies. The component enables the management of the various mitigation strategies and the evaluation of their impact on the overall risk. As illustrated in the case study in Section 4.5, this allows for the identification of the accepted set of security controls that need to be active during runtime at an asset-level in order for the ATL to reach to an accepted level (as framed by the RTL thresholds).

#### 4.2.1.6 Attack Path Calculator

The Attack Path Calculator serves as the core analytical engine for deriving topology-aware risk metrics within the CASTOR architecture. In its initial release, this subcomponent focuses on establishing the foundational data structures and interfaces required for cascading risk evaluation. Specifically, it provides the necessary user and API interfaces that allow a network security administrator to manually specify potential attack paths across the forwarding plane and assign estimated attack likelihoods to each attack

step. By processing these inputs alongside the network cartography and threat intelligence feeds, the calculator computes a preliminary topology-aware Required Trust Level (RTL) for the targeted assets for a given trust property (an example of such RTL walkthrough is provided in Section 4.5). While this manual approach relies heavily on expert knowledge and static estimations, it successfully validates the underlying RTL calculation logic before introducing complex Markov decision process in the second version.

Building upon this baseline, the second release of the Attack Path Calculator aims at fully automating the discovery and evaluation of these attack vectors. Rather than relying on static, administrator-defined paths, the upgraded calculator will systematically evaluate the dimensions (e.g., exploitability, cascading likelihood, attacker intent) of cascading attacks presented in Section 4.3. By integrating the automated Monte Carlo sampling process detailed in Section Section 4.4, the engine will be able to simulate multiple ”what-if” scenarios of potential threat actor trajectories through the asset topology.

#### 4.2.1.7 Trust Policy Engine

The Trust Policy Engine is responsible for operationalizing the parameters that govern the network's security and trustworthiness posture. As previously established in Deliverable D4.1 [6], the Trust Policy acts as the central blueprint that ”guides” the overall trust engineering process across the entire Trust Assessment Framework (TAF) federation. A comprehensive Trust Policy (see Chapter 2) encompasses the specific Trust Model to be evaluated, the exact trustworthiness evidence that must be collected from designated Trust Sources, and the target trust propositions required for the final trust report: for example, the Trust Policy for a Local TAF agent dictates the evaluation of atomic trust propositions concerning the specific integrity properties of a given vRouter. Conversely, the Trust Policy for the Global TAF outlines the broader, domain-wide trust evaluations necessary to inform the CASTOR Optimization Engine (details on the Optimization problem are presented in Chapter 5) and the overall CASTOR Orchestration layer, as detailed in Deliverable D5.1 [7]. Crucially, another fundamental element of this policy is the specification of the Required Trust Level (RTL) values, which serve as the definitive quantitative thresholds framing the trust decision process within any given TAF instance.

In this initial release, the Trust Policy Engine focuses primarily on implementing the core RTL methodology, synthesizing the risk and attack path data currently available from the associated subcomponents to establish these baseline thresholds. An exemplary case study that shows the impact of the risk methodology employed and the set of the security controls that are considered in the RTL calculation is presented in Section 4.5. Nevertheless, the overarching goal for the second release is to significantly extend this engine's capabilities to fully incorporate the broader trust relationships and distributed Trust Sources. Once this integration is complete, the engine will be capable of autonomously generating comprehensive Trust Policies that are subsequently encoded and pushed to the CASTOR DLT (see D5.2 [11]) and the Global TAF.

### 4.2.2 Risk Assessment Engine API Specification

This subsection presents the API specification of the CASTOR RA Engine. Following the architecture schema in Figure 4.1, most of the APIs presented below allow the Olistic Backend to orchestrate any request to the appropriate risk assessment subcomponents. Indicatively, these interfaces relate to the threat analysis by a Security Administrator, the communication for the RTL calculations as well as the reception of indication of risks (e.g., failed attestation evidence from a vRouter's TNDE).

Apart from the HTTP endpoints which are defined below - and can be used by the CASTOR framework or even the OLISTIC frontend for graphical user interfacing - the CASTOR RAE supports communication via the KAFKA [2] Publish-Subscribe protocol which constitutes the primary integration mechanism for the entire CASTOR Orchestration Layer. Details on these external interfaces (e.g., Kafka topics, information

flows with other CASTOR artifacts) is documented in D6.1 [9]. These interfaces are essential for ensuring the continuous and dynamic risk assessment process following the latest updates to the asset topology (e.g., a new link has been established/configured between two network elements) and its extracted risk indicators (e.g., the CASTOR TNDE element associated with an edge router has reported a deviation from its expected behaviour).

| **Name: VULNERABILITY_THREAT_INTERFACE** | | | |
|---|---|---|---|
| **Description** | Interface offered by the Vulnerability and Threat Modeling for managing threats, vulnerabilities. This is essential for enabling the threat analysis prior to any risk assessment task. | | |
| **Component providing the interface** | Vulnerability and Threat Modeling | | |
| **Consumer components or External Entities** | Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend | | |
| **Type of interface** | REST | | |
| **Input /Output Data** | **Methods or endpoints of the interface** | **Parameters of the method** | **Return Object or Values of the method** |
| | GET /api/v1/vulnerabilities /{id} | Vulnerability Identifier | Vulnerability Information |
| | POST /api/v1/vulnerabilities /{id} | Vulnerability Information | Vulnerability Identifier and name |
| | DELETE /api/v1/vulnerabilities /{id} | Vulnerability Identifier | HTTP OK |
| | GET /api/v1/threats /{id} | Threat Id | Threat information |
| | POST /api/v1/threats /{id} | Threat Information | Threat Identifier and name |
| | DELETE /api/v1/threats /{id} | Threat Identifier | HTTP OK |

Table 4.1: Vulnerability and Threat Modeling API

| **Name: ASSET_MODELLING_INTERFACE** | | | |
|---|---|---|---|
| **Description** | Interface offered by the Asset Topology Modeling component to enable security administrators to manage assets and asset topologies. This will enable the execution of fine grained risk analysis in pre-defined asset topologies corresponding to different trust models. | | |
| **Component providing the interface** | Asset Topology Modeling component | | |
| **Consumer components or External Entities** | Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend | | |
| **Type of interface** | REST | | |
| **Input/Output Data** | **Methods or endpoints of the interface** | **Parameters of the method** | **Return Object or Values of the method** |
| | GET /api/v1/assets /{id} | Asset Identifier | Asset information |
| | POST /api/v1/assets /{id} | Asset Information | Asset Identifier and name |
| | DELETE /api/v1/assets /{id} | Asset Identifier | HTTP OK |
| | GET /api/v1/processes /{id} | Process Id | Process information |
| | POST /api/v1/processes /{id} | Process Information | Process Identifier and name |
| | DELETE /api/v1/processes /{id} | Process Identifier | HTTP OK |
| | GET /api/v1/processes /{id}/visualize | Process Id | Asset topology information |

Table 4.2: Asset Topology Modeling component API

| **Name: ATTACK_PATH_AND_RTL_CALCULATOR_INTERFACE** | | | |
|---|---|---|---|
| **Description** | Interface offered by the Attack Path Calculator and the RTL calculator component to manage attack path assessments as well as the update of the already identified attack paths with new ones by the security administrator. | | |
| **Component providing the interface** | Attack Path Calculator, RTL Calculator | | |
| **Consumer components or External Entities** | Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend | | |
| **Type of interface** | REST | | |
| **Input/Output Data** | **Methods or endpoints of the interface** | **Parameters of the method** | **Return Object or Values of the method** |
| | GET /api/v1/attack-path-assessment /{id} | Attack Path Assessment Identifier | Attack path assessment information including the topology-aware RTL constraints |
| | POST /api/v1/attack-path-assessment /{id} | Attack path assessment information | Attack path assessment Identifier and name |

| | | | |
|---|---|---|---|
| | DELETE /api/v1/attack-path-assessment /{id} | Attack path assessment Identifier | HTTP OK |
| | POST /api/v1/attack-path-assessment /{id}/execute | Attack path assessment Id | HTTP OK |
| | POST /api/v1/attack-path-assessment /{id}/visualize | Attack path assessment Id | Attack paths and their respective risk in the format of a graph. |

Table 4.3: Attack path calculator component API

| **Name: RISK_QUANTIFICATION_ENGINE_INTERFACE** | | | |
|---|---|---|---|
| **Description** | Interface offered by the risk quantification engine that coordinates risk assessment tasks from the submission to their completion | | |
| **Component providing the interface** | Mitigation Strategies component | | |
| **Consumer components or External Entities** | Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend | | |
| **Type of interface** | REST | | |
| **Input/Output Data** | **Methods or endpoints of the interface** | **Parameters of the method** | **Return Object or Values of the method** |
| | GET /api/v1/risk-assessments /{id} | Risk Assessment Identifier | Risk Assessment information |
| | POST /api/v1/risk-assessments /{id} | Risk Assessment Information | Risk Assessment Identifier and name |
| | DELETE /api/v1/risk-assessments /{id} | Risk Assessment Identifier | HTTP OK |

Table 4.4: Risk Assessment strategies component API

| **Name: SECURITY_CONTROL_MANAGEMENT** | | | |
|---|---|---|---|
| **Description** | Interface offered by the mitigation strategies component to enable security administrators to manage the security controls that are associated with specific assets. | | |
| **Component providing the interface** | Mitigation Strategies component | | |
| **Consumer components or External Entities** | Risk Quantification Engine, OLISTIC Backend, OLISTIC Frontend | | |
| **Type of interface** | REST | | |
| **Input/Output Data** | **Methods or endpoints of the interface** | **Parameters of the method** | **Return Object or Values of the method** |
| | GET /api/v1/controls /{id} | Control Identifier | Control information |
| | POST /api/v1/controls /{id} | Control Information | Control Identifier and name |
| | DELETE /api/v1/controls /{id} | Control Identifier | HTTP OK |

Table 4.5: Mitigation Strategies component API

## 4.3 Properties that affect attack probability

To effectively capture the chance of a successful multi-hop attack, the overall attack success rate should be decomposed into a number of independent and measurable attributes or characteristics. This is in contrast to simply relying upon the coarse-grained CVSS score that conflates both the ease of exploiting a vulnerability (exploitability) and the potential impact of such exploitation. The proposed method will use a three dimensional decomposition to characterize the primary factors that influence the ability of an attacker to exploit a given node, traverse a network link, and have sufficient motivation to do so. Those dimensions include:

1. *Node Exploitability*
2. *Cascade Probability*
3. *Attacker Intent*

Each dimension represents one of the components of the per-step transition probability used by the Monte Carlo simulation described later in Section 4.4. In order to capture how these values may change over time rather than being set and fixed once, the framework considers each attribute (dimension) as a degree of belief which will update when new information becomes available. For instance, recently identified vulnerabilities, new attack patterns observed, or new threat intelligence data. Thus, the model is able to respond to evolving threats while avoiding the need to fully reassess all aspects of it.

### 4.3.1 Dimension 1: Node Exploitability

The first dimension measures the likelihood that an attacker can successfully target a particular node $v_i \in V$. Existing vulnerability scoring methods, such as CVSS, aggregate exploitability, which governs how likely it is that an attacker can reach and compromise a node, and impact, which reflects the consequences of successful compromise, into a single score. On the other hand, CASTOR offers a more granular approach by treating these two asymmetrically, by incorporating both exploitability and impact as separate variables while assigning a strictly higher weight to exploitability, i.e., $weight_{exploitability} > weight_{impact}$.

The exploitability component is derived from the four CVSS v3.x exploitability sub-metrics, which characterize the conditions under which an attacker can reach and compromise the target system:

- **AV** (Attack Vector): Represents the context through which the vulnerability could potentially be used to compromise the targeted system. Assigns a greater value the more remote an attacker can be to exploit the vulnerable component (e.g., Network: 0.85, Physical: 0.20)
- **AC** (Attack Complexity): Describes the requirements beyond the attacker's control that must exist to successfully exploit the vulnerability, where the Base Score is the highest possible for the least complex attacks (e.g., Low: 0.77, High: 0.44)
- **PR** (Privileges Required): Describes the privilege level an attacker must have prior to successfully compromising the targeted system via exploitation of the vulnerability, where the Base Score being greatest if no privileges are required (e.g., None: 0.85, Admin: 0.27)
- **UI** (User Interaction): Captures the requirement for a human user, other than the attacker, to interact with the targeted system so that the successful compromise of the vulnerable component can occur, with the Base Score being greatest when no user interaction is required (e.g., None: 0.85, Required: 0.62)

Following the CVSS v3.x specification [24], the base exploitability score is:

$$P_{CVSS\_expl}(v_i) = 2 \cdot AV_i \cdot AC_i \cdot PR_i \cdot UI_i \tag{4.1}$$

The CVSS Impact Sub-Score (ISS), which combines the impact metrics of Confidentiality ($C_i$), Integrity ($I_i$), and Availability ($A_i$), captures the impact component:

$$ISS(v_i) = 1 - (1 - C_i)(1 - I_i)(1 - A_i) \tag{4.2}$$

These two components are further complemented by the Exploit Prediction Scoring System (EPSS) and the patch status of the node. EPSS is a data-driven risk assessment framework designed to estimate the likelihood of Common Vulnerabilities and Exposures (CVE) being exploited in the wild within the near future. All four factors are combined into the final node explotaitability estimate as a weighted linear model:

$$P_{exploit}(v_i) = a_1 \cdot P_{CVSS\_expl}(v_i) + a_2 \cdot ISS(v_i) + a_3 \cdot EPSS(v_i) + a_4 \cdot Patch\_status(v_i) \quad (4.3)$$

where $a_1, a_2, a_3, a_4 \geq 0$ are weighting coefficients that satisfy $\sum_k a_k = 1$ and the constraint $a_1 > a_2$ enforces the importance of exploitability over impact. $Patch\_status(v_i) \in \{0, 1\}$ indicates whether a corrective patch has been applied, acting as a mitigation factor that reduces the effective exploitability. The specific values of these coefficients are empirically explored in the case study of Section 4.5.

### 4.3.2 Dimension 2: Cascade Probability

The second dimension measures the structural and network-level likelihood that an attacker moves from one node to another on a graph after an initial node has been compromised. The per hop cascade probability $\omega_{ij}$, where $i$ and $j$ are adjacent nodes on the same edge $(i, j)$ in graph G, is not a property of a single node, but rather the relationship between the two adjacent nodes. It defines how likely one lateral move is within a given segment of the network structure.

#### Per-Hop Cascade Probability

Each per-hop cascade probability $\omega_{ij}$ is modeled as a weighted sum of four factors:

$$\omega_{ij} = \beta_1 \cdot seg_{ij} + \beta_2 \cdot proto_{ij} + \beta_3 \cdot bw_{ij} + \beta_4 \cdot dist(v_j, v_t) \quad (4.4)$$

$\beta_1, \beta_2, \beta_3, \beta_4 \geq 0$ are positive weighting coefficients satisfying $\sum_k \beta_k = 1$, and each factor is defined as follows:

- $seg_{ij} \in (0, 1]$ is the boundary of the network segment of the link from $v_i$ to $v_j$, since all edges in $\mathcal{G}$ are communication links directly connecting adjacent nodes, a simple proximity metric will be uniformly non-informative. Rather, $seg_{ij}$ captures the cost to the attacker of crossing a segment boundary, reflecting the amount of separation or isolation imposed by the network boundaries between segments. Higher values indicate greater ease of lateram movement across domain boundaries.
- $proto_{ij} \in [0, 1]$, represents the risk that an attacker can intercept and manipulate data from a link connecting nodes $v_i$ and $v_j$ through protocol and communication medium. It represents two complementary factors for the link-layer attack surface of a given link. The first factor represents the communication protocol and the security measures it incorporates. For example, unencrypted protocols such as Telnet or FTP have a higher value for this factor than their encrypted counterparts, such as SSH and TLS. The second one represents the physical medium through which communication takes place, for example, wireless links represent a greater opportunity for an attacker to be able to listen in on communication and inject false messages into communication channels than wired or fiber-optic links, and thus, they would be assigned a higher value for this factor. The combined score reflects the overall vulnerability of the communication channel to interception or manipulation by an attacker who has already gained a foothold on $v_i$.
- $bw_{ij} \in [0, 1]$ is the bandwidth exposure of the link, defined as:

  $$bw_{ij} = \frac{BW_{ij}}{BW_{\max}}$$

  where $BW_{ij}$ is the raw bandwidth of link $(v_i, v_j)$ and $BW_{\max}$ is the maximum observed bandwidth in the topology. High-throughput connections provide attackers with the ability to rapidly transfer tools, payloads, or exfiltrate data, as well as provide sufficient traffic volume so that malicious activities

can be hidden inside legitimate network flows and therefore decrease the probability of detecting anomalies. On the other hand, low-throughput connections will limit what an attacker can do and increase the probability of detecting abnormal traffic patterns.

- The normalized topological distance $dist(v_j, v_t)$ is the hop count from $v_j$ to the target node $v_t$ and is given by

$$dist(v_j,\ v_t) = 1 - \frac{h(v_j\ , v_t)}{D}$$

where h($v_j$,$v_t$) is the hop-count of the shortest-path between nodes $v_j$ and $v_t$ and D is the diameter of the network. Therefore, nodes that are closer to the target in terms of their topology have higher intermediate "pivot" values for an attacker to use as part of a per-hop cascade.

**Residual Security Control Factor**

The raw cascading probability $\omega_{ij}$ represents the probability of propagation when no countermeasures are applied between node $v_i$ and node $v_j$. In practice, each link $(v_i, v_j)$ may be protected by a number of countermeasures including firewalls, intrusion detection systems, intrusion prevention systems, etc. Each countermeasure has a corresponding effectiveness represented by $\theta_c \in [0, 1]$. If there were no countermeasures at all ($\theta_c = 0$), then the effectiveness would be zero, and if all countermeasures were completely effective ($\theta_c = 1$), then the effectiveness would be one. With the assumption that each countermeasure operates independently, the residual cascading factor after applying all countermeasures to a link is:

$$\Theta_{ij} = \prod_{c \in \mathcal{C}_{ij}} (1 - \theta_c) \tag{4.5}$$

When no controls have been deployed on a link, $\mathcal{C}_{ij} = \emptyset$ and $\Theta_{ij} = 1$ (i.e. there is no impact). When additional controls are deployed or the existing controls are enhanced, $\Theta_{ij}$ will decrease toward 0. The total cascade potential along the entire path will be:

$$P_{\text{cascade}}(\mathcal{P}) = \prod_{k=0}^{n-1} \omega_{k,\,k+1} \cdot \Theta_{k,\,k+1} \tag{4.6}$$

Since each factor $\omega_{k,k+1} \cdot \Theta_{k,k+1} \leq 1$, the product will always decrease monotonically with respect to the path length; that is, the longer the attack path, the greater the cumulative complexity for the attacker to be successful at every intermediate hop against all controls. Thus this formulation inherently encodes the notion that topological distance reduces the viability of attacks, without the need for a decay term.

The per-hop cascade probability $\omega_{ij}$ is considered a prior degree of belief in the probability of an individual's move laterally from $v_i$ to $v_j$ based on observable network properties. This prior belief will be adjusted using Bayes inference, to create a posterior estimate for each iteration that better represents the actual environment at assessment. This view of estimating uncertainty is consistent with the overall Monte Carlo approach. In the Monte Carlo method, all uncertain parameters are iteratively sampled, so there is no need to assign a value to a parameter at a specific time.

### 4.3.3 Dimension 3: Attacker Intent

The third dimension captures the motivation of an attacker to target a particular node $v_j$, while dimensions 1 and 2 represent the technical vulnerability (i.e., the exploitability) of the individual nodes and the

structure and control adjusted conditions required for propagation between nodes. Dimension 3 represents the strategic value of a node as a target for an adversary, that is, the degree to which its compromise would advance the attacker's objectives.

In the Monte Carlo Simulation framework, all three dimensions represent vectors of attributes for each node and edges within the network. While some of the attributes can be observed directly, e.g. known CVSS scores, published segmentation policies etc., many are unknown during assessment, such as whether a patch was successfully applied, or if a countermeasure was actually implemented and is active. As a result, we treat the uncertain attributes as hidden variables during the simulation process and sample them from appropriately defined distributions over multiple iterations to generate a distribution of attack probabilities instead of just a single estimated probability of an attack occurring.

The target attractiveness score $W_{\text{target}}(v_j) \in [0, 1]$ is modelled as a weighted sum of two complementary factors:

$$W_{\text{target}}(v_j) = \gamma_1 \cdot crit(v_j) + \gamma_2 \cdot BC_{\text{norm}}(v_j) \tag{4.7}$$

where $\gamma_1 + \gamma_2 = 1$, and each factor is defined as follows:

- $crit(v_j) \in [0, 1]$ represents the criticality of assets for a node $v_j$, essentially the operational or business risk posed by losing access to this asset. The criticality is assessed before launching the campaign and depends on the role played by the node within the system, for example, a central router or trust anchor receives a higher criticality score opposed to a peripheral monitoring node.
- $BC_{\text{norm}}(v_j) \in [0, 1]$ is the normalized betweenness centrality of $v_j$. Betweenness centrality is first computed as:

$$BC(v_j) = \sum_{\substack{s,t \in V \\ s \neq t,\ s \neq v_j,\ t \neq v_j}} \frac{\sigma_{st}(v_j)}{\sigma_{st}} \tag{4.8}$$

where $\sigma_{st}$ is the total number of shortest paths from $s$ to $t$, and $\sigma_{st}(v_j)$ is the number of those paths that pass through $v_j$. It is later normalized over all nodes in the topology, similarly to other properties:

$$BC_{\text{norm}}(v_j) = \frac{BC(v_j)}{\max\limits_{v \in V} BC(v)} \tag{4.9}$$

A node with high Betweenness Centrality is a structurally key position within the network. Its attack will allow an attacker to either intercept, disable, or alter a disproportionately large number of the networks traffic flow. The ability for an attacker to do so creates a strategic incentive for attacking this node. This incentive is based on its structural importance and does not include the nodes inherent value.

**Combining all dimensions** All three dimensions are combined to calculate the total probability of an end-to-end attack path from ingress node $v_s$ to target node $v_t$:

$\pi = (v_s, v_{i_1}, \ldots, v_t)$

These three dimensions follow a Markov approach: Each hop has a transition probability dependent solely upon its current position and the potential next hop; this does not depend on all previous hops that were followed to reach the present position. Therefore, the simulation will progress as a probabilistic state machine in which it samples the probabilities determined by these three dimensions at the current location in the graph, versus computing the probability of a complete path.

To begin with, we will express the contribution of each step as a weighted sum of all steps in the attack path:

$$P_{path}(\pi) = \sum_{k} \Big[ P_{exploit}(v_{i_k}) + P_{cascade}(v_{i_k} \rightarrow v_{i_{k+1}}) + W_{target}(v_{i_{k+1}}) \Big] \tag{4.10}$$

It is worth noting that the transition probability calculation in Eq. 4.10 uses a ”local” (or ”one-hop”) approach to evaluate each step individually; the value of the path from node $v_{i_k}$ to $v_{i_{k+1}}$ does not depend upon any of the previous hops in the path, and therefore this local method represents a direct way to satisfy the Markov Property of evaluating the ”current” state based on all available information about that state alone. An alternative strategy to use a product or other multiplicative operation across all the transitions in a path remains an area of potential improvement.

### 4.3.4 Topology-Driven RTL Escalation

Beyond their role in the attack propagation model, the network structural attributes presented in section 4.3.2, have a direct impact on RTL derivation. A node with high betweenness centrality will often appear across many different routes of attack. This means even if no single path has a high probability of attack, the node is still a high-risk target due to its structural location in the network. Thus, such nodes should have stricter RTL thresholds, irrespective of any individual path probability computed by the Monte Carlo simulation.

A node that acts as a connection point for variously separated areas of a network, also known as a hub, is a common characteristic of ring and hierarchical based networks typically used within an operational environment. Therefore, when a hub is compromised, all possible paths of attack are opened with this single breach. Consequently, nodes having a higher degree of betweenness centrality should have a more restrictive RTL. That is, the more central the node, the more security evidence needs to be collected and verified before being added to a trusted path.

## 4.4 Monte Carlo Simulation for Attack Propagation Analysis

### 4.4.1 Motivation and Design Rationale

CASTOR uses Monte Carlo simulations for reasoning under uncertainty: the unknown (hidden) variables are the attack parameters that cannot be measured, such as patch level of vulnerability, countermeasures activated, or behavior of attackers; by sampling these parameters over several iterations it generates a distribution of probabilities of attack instead of an exact number.

The three dimensional Attack Propagation Model defined in detain in section 4.3 serves as a base-line for assessing the probability of success for attacks that traverse multiple hops. However, developing a scalable method to translate individual-step probabilities into a global measure of how vulnerable the overall network may be to such attacks is challenging. In addition to the multitude of possible attack paths, the model must address the variability of attack methods, stochastic input parameter uncertainty, and goal-directed behaviour of adversaries. CASTOR addresses those requirements by implementing Monte Carlo simulations in order to provide probabilistic what-if scenarios about the network topology $\mathcal{G}$.

In contrast to a deterministic calculation of a single attack probability, the simulation executes N separate attack simulations. Each of these simulations represents a possible series of exploitation and lateral movement actions from the ingress node $v_s$ to the target node $v_t$. The results of the multiple simulations are statistically aggregated to produce a distribution of attack probabilities. These distributions are sensitive to both defensive posture of individual links and nodes within the network, while also taking into consideration the topology of the network. As mentioned in Section 4.3, most of the characteristics

included in the three dimensions, such as patch status or the active status of a particular countermeasure, are not directly defined at assessment time. Therefore, these inputs are treated as hidden variables throughout the iterative process, ensuring that the simulation corresponds to epistemic uncertainty rather than generating overconfident point estimates.

### 4.4.2 Incorporating the Tree Dimensions into the Simulation

Each iteration models a single attack campaign as a goal-directed traversal of $\mathcal{G}$, in which an attacker attempts to move from the source $v_s$ towards the target node $v_t$ , using a sequence of exploitation and lateral movement steps. As the attacker moves through the graph, the probability of the next successful movement and in which general direction they should move, is determined by all three aspects of their propagation, as explained in detail below.

**Step 1 : Node Exploitation (Dimension 1)** Upon arriving a particular node $v_i$, the attacker will attempt to exploit a vulnerability. The probability of success is evaluated using the full Dimension 1 model (Equation 4.3):

$$P_{exploit}(v_i) = a_1 \cdot P_{CVSS\_expl}(v_i) + a_2 \cdot ISS(v_i) + a_3 \cdot EPSS(v_i) + a_4 \cdot Patch\_status(v_i)$$

At runtime, the values for AV, AC, PR, UI as well as C, I, A which are sub-metrics of CVSS exploitability and impact scores respectively, will be retrieved from the NVD API, whilst the value for EPSS will be obtained from the First API. Therefore, the values used in the simulation will reflect current information about the vulnerabilities. The simulation uses Bernoulli trials to simulate each potential step taken by the attacker. A trial with probability $P_{exploit}(v_i)$ determines whether exploitation succeeds. In case of failure, the path terminates at $v_i$.

**Step 2 : Next-Hop Selection (Dimension 2 and 3)** When exploitation is successful, the adversary will choose his next hop in the graph from the adjacent nodes of $v_i$, using a weighted approach based on both Dimensions 2 and 3:

$$w(v_i \to v_j) = P_{cascade}(v_i \to v_j) \cdot W_{target}(v_j) \tag{4.11}$$

where $P_{cascade}(v_i \to v_j) = \omega_{ij} \cdot \Theta_{ij}$ encodes the structural traversability of the link and the residual effectiveness of deployed countermeasures (Equations 4.4 and 4.5), and $W_{target}(v_j) = \gamma_1 \cdot crit(v_j) + \gamma_2 \cdot BC_{\text{norm}}(v_j)$ captures the attractiveness of the possible next hop with respect to Dimension 3 ( Equation 4.7). The next hop node $v_j$ is the chosen according to a multinomial distribution defined by normalized weights $w(v_i \to v_j)$ on all successors of $v_i$. By modelling the behaviour of a logical, goal-directed adversary rather than an opportunistic one, this guarantees that the random walk is biased toward nodes that are both strategically valuable and easy to reach.

**Step 3 : Cascade Attempt (Dimension 2)** Having determined a potential next hop to an adjacent node $v_j$, the adversary must execute the lateral movement. The success or failure of the move is determined by a second Bernoulli trial, whose probability for success is given as $P_{cascade}(v_i \to v_j)$. Upon a failure in the lateral movement, the path will terminate. Alternatively, upon successful completion of the lateral movement, the attacker advances to node $v_j$ and repeats from Step 1. The simulated attack moves strictly forward through the topology towards the target $v_t$ by excluding nodes already visited along the current path from the candidate set at each subsequent step, preventing cycles within a single campaign.

**Termination.** A campaign ends based on one of three criteria: the attacker successfully reaches the target node $v_t$, a Bernoulli trial fails at the exploitation or cascade step, or the current node has no outgoing links in $\mathcal{G}$. Paths usually end before reaching $v_t$, due to unsuccessful exploitation or lateral movement. The resulting path $\pi = (v_s, v_{i_1}, \ldots, v_k)$ is recorded regardless of whether $v_t$ was reached, as partial paths still contribute to the node compromise and edge cascade probability estimates.

### 4.4.3 Simulation Output and Connection to RTL Derivation

Following execution of N iterations of the simulation, the system generates three complementary outputs measuring the likelihood of successful attacks against the test bed network at various abstraction levels:

- **Node compromise probability** $P_{comp}(v_i)$: the proportion of iterations in which $v_i$ has been found on an attack path. It provides a quantitative estimate of the overall exposure of each node during all simulated attacks, combining direct exploitation and use as a hub in multi-hop chains.
- **Path probability** $P_{path}(\pi)$: the proportion of iterations in which the specific node sequence $\pi = (v_s, v_{i_1}, \ldots, v_k)$ was recorded as a successful attack path. Thus this method identifies the most frequently traversed routes in the network topology, while also directly supporting what-if analysis by providing the probabilities of certain multi-hop attack scenarios.
- **Edge cascade probability** $P_{casc}(v_i, v_j)$: the proportion of iterations in which the lateral movement from node $v_i$ to node $v_j$ was selected, indicating the most exploited links in the network.

The output of these modules is directly fed into the RTL derivation process as described in Section 4.5. The connection works as follows: nodes which appear most often across simulated attack paths will have a greater degree of certainty that they are being targeted or used as a stepping stone by an attacker. This is reflected by assigning a lower disbelief threshold $d_{RTL}$ to that node, that is, the TAF must obtain additional attestations with no negative evidence before admitting the node to a trusted path. In concrete terms, $P_{comp}(v_i)$ scales $d_{RTL}$ downward with the frequency of appearance of $v_i$ in simulations.

As previously introduced in D4.1 [6], the belief threshold $b_{RTL}$ is calculated from the maximum risk level $R_{max}$ using a baseline belief $b_t$ and a uniform step-size $\Delta$.

$$\Delta = \frac{1 - b_t}{5} \tag{4.12}$$

$$b_{RTL} = b_t + (R_{\max} - 1) \times \Delta \tag{4.13}$$

where $b_t \in [0, 1]$ is an organizational baseline that ensures minimum trust requirements even in low-risk scenarios. The maximum risk $R_{max}$ denotes the maximum risk level (on a scale of 1-5 derived from risk assessment). $\Delta$ divides the range of available belief (from $b_t$ to 1) into five equal intervals corresponding to the five risk levels. Finally, the disbelief threshold $d_{RTL}$ is determined using the CIA impact profile of the vulnerability:

$$d_{RTL} = 1 - I_w \tag{4.14}$$

where $I_w \in [0, 1]$ is the numerical impact weight, mapped from the CVSS-impact-rating (None $\rightarrow 0.00$, Low $\rightarrow 0.50$, High $\rightarrow 1.00$), following the severity classification of [1].

## 4.5 Case Study: From Risk Analysis to topology-aware RTL calculation

This section showcases complete operation of CASTOR Risk Assessment Engine as an example case to demonstrate how the three-dimensional attack propagation model and Monte Carlo simulation combined, influence the derivation of RTL while providing quantifiable examples of the impact of topology knowledge and security mechanisms on the trust levels derived from the engine. Thus, the evaluation is structured by means of three successive test cases which cover include all functionalities of the first release of the CASTOR RAE prototype.

As outlined in D4.1 [6], RTL constraints in binomial opinions can frame the acceptable trust decisions in three dimensions: belief, disbelief, and uncertainty (illustrated within the triangular value space of trust opinions in Figure 4.2). As a deliberate design choice for this first release, the CASTOR Risk Assessment Engine focuses exclusively on systematically deriving RTL thresholds for the belief and disbelief dimensions. Cascading attacks introduce two distinct types of uncertainty: the stochasticity of transition probabilities (aleatoric uncertainty) and incomplete knowledge regarding potential topological impacts (epistemic uncertainty). It is envisioned that both forms of uncertainty are reflected in the revised, topology-aware risk values calculated for each node, assuming full confidence in the impact each security control has on an identified risk. Consequently, the derived belief and disbelief RTL thresholds inherently reflect these combined uncertainties in the transition probabilities, as well as the degree of confidence in the outcome of each attack step within the cascading scenarios that ultimately impact the final risk values. For this initial version, defining an explicit, separate uncertainty RTL threshold is left to the discretion of the Security Administrator. This allows human operators to manually constrain the acceptable trust opinion space further, typically to establish a strict upper bound on the level of vacuity permitted in the runtime Actual Trust Level (ATL) values.

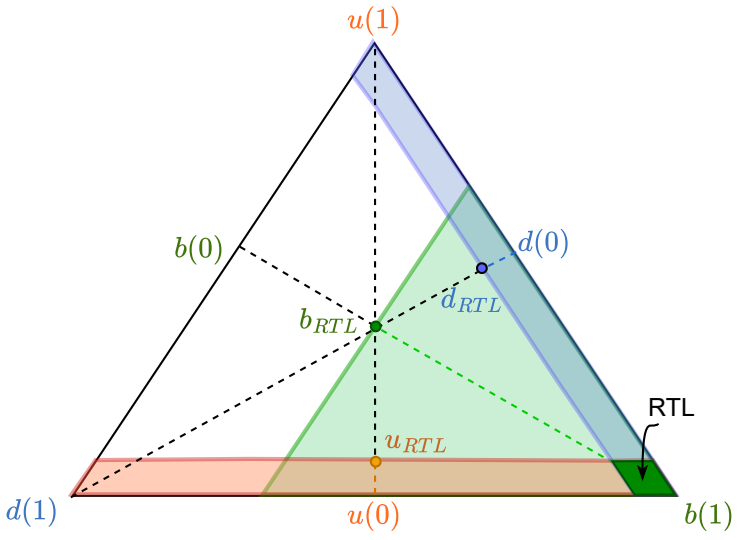



Figure 4.2: Graphical representation of RTL thresholds within the subjective logic triangle. The RTL constraints for belief ($b_{RTL}$), disbelief ($d_{RTL}$), and uncertainty ($u_{RTL}$) define an acceptable region for trust opinions (ATL) points. In this example, the trust decision is positive in the cases where the ATL opinion is positioned in the bottom right region marked as ”RTL”.

### 4.5.1 Evaluation setup: Asset Topology and Vulnerability Profile

Our case study utilizes a linear topology composed of four nodes to simulate a part of a CASTOR enabled routing environment:

$$R_1 \longrightarrow R_2 \longrightarrow R_3 \longrightarrow R_4$$

Node $R_1$ is an IS-IS level-1 edge router and acts as the attacker ingress point. $R_2$ and $R_3$ are in the IS-IS level-2 environment, while $R_4$ is a high value node. Each node has at least one real CVE drawn from a public vulnerability list. The according CVSS v3.1 scores were downloaded from the NIST National Vulnerability Database (NVD), while EPSS scores were obtained using the FIRST API. The vulnerability profile is summarised in Table 4.6.

Table 4.6: Vulnerability profile of the case study topology.

| Node | CVE | Type | CVSS | C | I | A |
|---|---|---|---|---|---|---|
| $R_1$ | CVE-2024-20323 | DoS / PCEP | 7.5 | L | H | N |
| $R_2$ | CVE-2023-38802 | RCE / BGP | 7.5 | N | N | H |
| $R_2$ | CVE-2024-39531 | BGP crash | 7.5 | N | N | H |
| $R_3$ | CVE-2024-0012 | Auth bypass | 9.8 | H | H | H |
| $R_3$ | CVE-2021-31375 | RPKI bypass | 7.2 | N | L | L |
| $R_4$ | CVE-2024-0012 | Auth bypass | 9.8 | H | H | H |

Link parameters (segmentation boundary, protocol risk, bandwidth exposure) state how the IS-IS level-1/level-2 area boundaries affect the topology of the network: The link from $R_1$ to $R_2$ crosses a domain boundary ($seg = 0.5$), while all core links are known to be connected inside the same segment ($seg = 1.0$). As mentioned in Section 8.3 in D4.1 [6], the RTL Methodology involves the definition of distinct thresholds for each of the components that form the ATL trust opinion. For instance, in the context of binomial opinions - the possible values of which can be visually represented in the Subjective Logic triangle of Figure 4.2 - the possible RTL thresholds can be $b_{RTL}$, $d_{RTL}$, and $u_{RTL}$. Each RTL value is calculated separately for each CIA property (Confidentiality (C), Integrity (I) and Availability (A)), and so those values are explicitly stated in Table 4.6. Each column is used to compute the per property Disbelief Threshold $d_{RTL}$, reflecting the asymmetric impact profile of each vulnerability with respect to the three security dimensions.

### 4.5.2 Scenario 1: Isolated Vulnerability Assessment

In the first scenario, an independent assessment of routers is conducted, ignoring both attack path propagation and use of security controls. The belief thresholds $b_{RTL}$ are established as the highest risk level from each CVEs associated with the node. On the other hand, disbelief threshold $d_{RTL}$ is computed separately for each of three CIA properties individually based on its respective impact rating. This baseline scenario is equivalent to a conventional, static risk assessment that ignores the network context of each node. In the interest of brevity, we present the RTL derivation step by step for just $R_1$ which is the entry point of this campaign and demonstrates how topology-awareness and security control can be incorporated into the RTL as each successive scenario progresses.

**Step-by-step derivation for $R_1$.** To illustrate the calculation procedure, we walk through the full RTL derivation for $R_1$, which carries a single CVE: CVE-2024-20323 (CVSS 7.5, AV:N/AC:H/PR:N/UI:N, C:L/I:H/A:N).

**Step 1 - CVSS exploitability score.** The four CVSS v3.x exploitability sub-metrics are mapped to their according numerical values (AV:N = 0.85, AC:H = 0.44, PR:N = 0.85, UI:N = 0.85) and combined Equation 4.1:

$$P_{CVSS\_expl}(R_1) = 2 \times 0.85 \times 0.44 \times 0.85 \times 0.85 = 0.541$$

**Step 2 - Feasibility score**. The exploitability score is mapped to a discrete risk level (1-5 scale) used by the RTL derivation framework:

$$F = \text{round}(0.541 \times 5) = 3$$

A feasibility level of 3 indicates that it would be moderately possible for an attacker to exploit this vulnerability. This value was derived based on the high Attack Complexity (AC:H) of CVE-2024-20323, therefore reducing the total feasibility of the vulnerability, as opposed to simpler, fully network-accessible vulnerabilities.

**Step 3 - Maximum risk level per CIA property**. For each CIA property, the impact value is mapped to an impact level using the $CIA\_TO\_LEVEL$ mapping (None $\rightarrow$ 1, Low $\rightarrow$ 2, High $\rightarrow$ 4). The maximum risk level $R_{max}$ represents the most adverse scenario concerning feasibility and impact:

$$R^C_{\max} = \max(F,\ level(C{:}L)) = \max(3,\ 2) = 3$$
$$R^I_{\max} = \max(F,\ level(I{:}H)) = \max(3,\ 4) = 4$$
$$R^A_{\max} = \max(F,\ level(A{:}N)) = \max(3,\ 1) = 3$$

The integrity property prevails in this case as CVE-2024-20323 possesses a High Integrity impact, resulting a higher $R^I_{\max}$ than the feasibility level alone would indicate.

**Step 4 - Belief threshold per CIA property**. Using the baseline belief $b_t = 0.2$ and step size $\Delta = (1 - 0.2)/5 = 0.16$ (Equations 4.12 and 4.13):

$$b^C_{RTL} = 0.2 + (3 - 1) \times 0.16 = 0.52$$
$$b^I_{RTL} = 0.2 + (4 - 1) \times 0.16 = 0.68$$
$$b^A_{RTL} = 0.2 + (3 - 1) \times 0.16 = 0.52$$

A higher $b_{RTL}$ means that the TAF must collect more positive trust evidence before admitting the node into a trusted path.

**Step 5 - Disbelief threshold per CIA property**. The CIA impact values are mapped to numerical impact ratings following the severity classification proposed in [1] (None $\rightarrow 0.00$, Low $\rightarrow 0.50$, High $\rightarrow 1.00$) and the disbelief threshold is computed as $d_{RTL} = 1 - I_w$ (Equation 4.14):

$$d^C_{RTL} = 1 - 0.50 = 0.50 \quad (C{:}L \rightarrow I_w = 0.50)$$
$$d^I_{RTL} = 1 - 1.00 = 0.00 \quad (I{:}H \rightarrow I_w = 1.00)$$
$$d^A_{RTL} = 1 - 0.00 = 1.00 \quad (A{:}N \rightarrow I_w = 0.00)$$

Consequently, each pair of belief and disbelief RTL thresholds derived for a specific trust property serves as the definitive baseline for evaluating the corresponding, evidence-driven ATL opinion at runtime. Similarly for $R_2$, $R_3$ and $R_4$, the complete set of individualized RTL values can be determined. A significant observation regarding this particular example is that isolated assessments result in identical RTL values assigned to $R_3$ and $R_4$ nodes, despite the fact that they have structurally differing roles within the topology. The above limitation is overcome in Scenario 2.

### 4.5.3 Scenario 2: Attack-Path-Aware RTL Derivation

In the second scenario, the full three-dimensional Monte Carlo simulation and topology-driven RTL escalation are activated. For this purpose, previously computed normalized betweenness centrality values will increase the $b_{RTL}$ thresholds of structurally central nodes compared to those produced by the compromise probabilities alone derived from the simulation. The simulation executes 10,000 iterations of goal-oriented attacks from $R_1$ to $R_4$, resulting in the following compromise probabilities at each node:

$$P_{\text{comp}}(R_1) = 0.0865, \quad P_{\text{comp}}(R_2) = 0.0865, \quad P_{\text{comp}}(R_3) = 0.0364, \quad P_{\text{comp}}(R_4) = 0.0194$$

The most probable complete attack path for an attacker to reach the target $R_4$, is the path $R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_4$ with an associated path probability of $P_{\text{path}} = 0.0194$. Interestingly, the most common partial path found over all iterations of simulation is $R_1 \rightarrow R_2$ ($P = 0.0501$), which confirms that $R_2$ is the major lateral movement bottleneck within this particular topology. The normalized betweenness values, which were pre-computed using the bidirectional graph of the topology are:

$$BC_{\text{norm}}(R_1) = 0.0, \quad BC_{\text{norm}}(R_2) = 1.0, \quad BC_{\text{norm}}(R_3) = 1.0, \quad BC_{\text{norm}}(R_4) = 0.0$$

**From simulation outputs to RTL adjustments.** The Scenario 1 RTL constraints represent the base case for this example. Two mechanisms independently alter the values produced from the simulation, and both mechanisms address a different component of the RTL triple.

The first mechanism alters the disbelief threshold $d_{RTL}$ of the individual nodes. Through repeated simulations of attacks on networks, some nodes appear repeatedly. Because these nodes have a higher probability to be targeted by adversaries, or because they can be used as stepping stones, the exposure of these nodes is greater. Therefore, the disbelief threshold for such a node is reduced based upon its node compromise probability $P_{\text{comp}}(v_i)$.

$$d_{RTL}^{\text{adj}}(v_i) = d_{RTL}(v_i) \cdot \big(1 - P_{comp}(v_i)\big) \tag{4.15}$$

Since $d_{RTL}$ defines the maximum level of disbelief that the TAF can accept in a node's ATL in order to derive a positive Trust Decision, lowering that threshold sets a higher barrier for admission: the Global TAF assigns a positive trust evaluation only to nodes whose ATLs reflect minimal to no adverse evidence (see Section 2.4.5 for the discussion on TAF's Trust Decision Engine).

The second mechanism impacts the belief threshold $b_{RTL}$ through the topology-driven RTL escalation procedure. All nodes with normalized betweenness centrality $BC_{\text{norm}}(v_i) > 0$ will have its beliefs increased by the same amount as its degree of centrality.

$$b_{RTL}^{\text{adj}}(v_i) = \min\big(1,\ b_{RTL}(v_i) + BC_{\text{norm}}(v_i) \cdot w_{BC}\big) \tag{4.16}$$

Where $BC_{\text{norm}}(v_i)$ is described in Equation 4.9 and $w_{BC} \in [0, 1]$ is set by the security administrator to determine how much influence the topological increase has on the belief escalation. Therefore, for that particular node, the TAF will need to gather a greater quantity of positive trust evidence than would otherwise be required to enter into a trusted path. Since $BC_{\text{norm}}(R_1) = 0$, there is no impact from the topology-driven escalation of the belief thresholds on $R_1$'s RTL compared to Scenario 1. Its changes are entirely based upon $P_{comp}(R1) = 0.0865$, which reflects its position as the campaign entry point.

The RTL results for $R_1$ are reported in Table 4.7.

By comparing Scenario 2 to Scenario 1, we can observe that topology information impacts RTL. For $R_1$, the confidentiality property has moved from HIGH to MEDIUM risk, which follows the slightly lower $d_{RTL}$ value driven by $P_{comp}(R1) = 0.0865$. Availability maintains its HIGH risk due to the low availability loss associated with CVE-2024-20323 (A:N), as it imposes a high level of disbelief regardless of the topology adjustment.

Table 4.7: Scenario 2: RTL values for node $R_1$ per CIA property (attack-path-aware, no controls). The values in bold reflect the changes in the RTL thresholds compared to Scenario 1.

| Node | Prop. | $b_{RTL}$ | $d_{RTL}$ | Risk |
|---|---|---|---|---|
| $R_1$ | C | 0.520 | **0.457** | MEDIUM |
| $R_1$ | I | 0.680 | 0.000 | LOW |
| $R_1$ | A | 0.520 | **0.914** | HIGH |

### 4.5.4 Scenario 3: Security-Control-Aware RTL Derivation

The third scenario extends Example 2 with the introduction of four new security controls on each node. Each protection has an efficiency rating $\theta_c \in [0, 1]$ that indicates how much each protection can reduce the remaining cascade potential $\Theta_{ij}$ on a protected edge (Equation 4.5). Each security measure can be applied to one or more of the links adjacent to R1. In addition, we assigned empirical values to the effectiveness of these four security measures against our current topological vulnerability profile. Those empirical assignments were based upon the estimated level of protection they would provide against each type of vulnerability in this topology. We assign those empirical estimates as follows:

- **Secure Boot** ($\theta_c = 0.60$)
- **Control Flow Integrity (CFI)** ($\theta_c = 0.35$)
- **Rollback Protection** ($\theta_c = 0.20$)
- **Access Control** ($\theta_c = 0.40$)

Each control is assessed independently, with all other controls inactive, so that the individual risk reduction characteristics of each mechanism is manifestly present.

Before we present the results of our research, we need to explain how the different types of security controls influence the RTL threshold values. Security controls do not affect $b_{RTL}$ directly, but rather through their indirect effects in terms of decreasing the exploitability of the node in question. To be more specific, for every control with effect $\theta_c$, $P_{expl}$ is decreased by a factor of $(1 - \theta_c)$ (Equation 4.1). As a result of this decrease, $F$ decreases and $R_{\max}$ decreases as well, ultimately resulting in a decrease in $b_{RTL}$ via Equation 4.13.

In addition to these indirect effects, security controls also have direct effects on the disbelief tolerance according to:

$$d_{RTL}^{ctrl}(v_i) = d_{RTL}(v_i) + \theta_c \cdot \big(1 - d_{RTL}(v_i)\big) \tag{4.17}$$

The above equation indicates that since a security control is actively engaged, it is likely that the residual attack surface of a link has been reduced, thus increasing the strictness of the disbelief constraint.

Table 4.8 reports the risk classification of $R_1$ exclusively in each configuration of a graphical form, alongside our Scenario 2 baseline. Each of the controls used to enhance protection against lateral movement of an adversary is going to reduce the residual cascade factor $\theta_{ij}$ (Equation 4.5), thereby lowering the probability of the adversary's ability to travel across a protected link and thus will increase the disbelief threshold $d_{RTL}$. As shown in Table 4.8, the security controls exhibits the greater impact for Confidentiality and Availability trust properties. More specifically, Secure Boot lowered Availability from HIGH to LOW and Confidentiality to MEDIUM, whereas CFI, Rollback Protection, and Access Control were able to partially lower both Confidentiality and Availability to MEDIUM. We also observe that Integrity retained HIGH

across all control configurations, as the high belief threshold imposed by the I:H impact rating of CVE-2024-20323 ($b_{RTL} = 0.68$) is not reduced by any individual control at the effectiveness levels considered here.

Table 4.8: Scenario 3: Risk and RTL thresholds (two-decimal precision) for $R_1$ per CIA property under different mitigation strategies. Each strategy considers the application of a single security control. Risk values presented in the example are: High (H), Medium (M), Low (L).

| | Trust Property | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Confidentiality | | | Integrity | | | Availability | | |
| **Mitigation Strategy** | **Risk** | $\mathbf{b^C_{RTL}}$ | $\mathbf{d^C_{RTL}}$ | **Risk** | $\mathbf{b^I_{RTL}}$ | $\mathbf{d^I_{RTL}}$ | **Risk** | $\mathbf{b^A_{RTL}}$ | $\mathbf{d^A_{RTL}}$ |
| 1. **No Controls** | H | 0.52 | 0.46 | H | 0.68 | 0.00 | H | 0.52 | 0.91 |
| 2. **Secure Boot ($\theta_c = 0.60$)** | M | 0.36 | 0.73 | H | 0.68 | 0.54 | L | 0.20 | 0.91 |
| 3. **CFI ($\theta_c = 0.35$)** | M | 0.36 | 0.61 | H | 0.68 | 0.32 | M | 0.36 | 0.91 |
| 4. **Rollback ($\theta_c = 0.20$)** | M | 0.36 | 0.55 | H | 0.68 | 0.18 | M | 0.36 | 0.91 |
| 5. **Access Control ($\theta_c = 0.40$)** | M | 0.36 | 0.63 | H | 0.68 | 0.36 | M | 0.36 | 0.91 |

As observed in Scenario 1, the disbelief component of availability does not change with the control configurations since the A:N impact rating of CVE-2024-20323 fixes $d^A_{RTL}$ = 1.0 before it was adjusted in a Monte Carlo simulation; there is then no room to relax the controls based on this component, and the reduction is limited to $0.91$. Therefore, the trust decision on the availability property is primarily determined by the $b^A_{RTL}$ threshold.

This evaluation demonstrates how security controls effectively reduce the cascading probability, $P_{cascade}$ (see Equation 4.6), which ultimately impacts the final risk and RTL values. For instance, under the second mitigation strategy (Table 4.8), applying Secure Boot reduces the Confidentiality risk for $R_1$ from HIGH to MEDIUM (compared to the "No Controls" mitigation strategy), corresponding to a roughly 30% decrease in the belief threshold $b^C_{RTL}$. The impact is even more apparent in the context of the Availability trust property, where the most efficient risk reduction causes the belief threshold $b^A_{RTL}$ to drop by over 60%. As detailed in the following section, these metrics highlight a direct relationship between deployed security controls and evidence collection: based on this evaluation scenario, the CASTOR TAF must collect the appropriate type of evidence to prove these controls are active, thereby generating an ATL opinion whose belief component is strong enough to satisfy the respective RTL constraint.

### 4.5.5 Discussion and Critique: Towards a robust RTL methodology

#### RTL methodology and Trust Decision

In what follows, we reflect upon the core takeaway messages from this case study analysis. Depending on the modelling and the parameterization in the RTL methodology, we may establish strict or unattainable targets to be exceeded by the ATL, or overly relaxed constraints. For instance, Scenario 1 in Section 4.5.2 shows that the disbelief threshold on availability, $d^A_{RTL}$, is maximized to 1, implying that any level of disbelief in an ATL trust opinion does not affect the trust decision outcome. In parallel, in the rest of the scenarios (see Section 4.5.3 and Section 4.5.4) we observe that the same disbelief threshold remains at high level, namely $d^A_{RTL} \approx 0.91$. This implies that the evidence-based ATL trust opinion, computed during runtime, needs to exhibit a rather confident evaluation of unavailability in order to lead to a negative trust characterization of the Availability of $R_1$.

#### Desiderata between security controls and mitigation strategies

Scenario 3 provides a clear process in the Risk Assessment lifecycle where a Security Administrator is able to specify a set of security controls in terms of mitigation strategies that may reduce the impact

or likelihood of specific one-step (Scenario 1) or cascading attacks (Scenario 2). In Scenario 3 (Section 4.5.4), we consider four distinct mitigation strategies (i.e., apart from the "No security control applied" case), each consisting of a single security control. This is done to better depict the efficiency of different controls on the risk posture of $R_1$; however, in principle, a mitigation strategy may consist of multiple controls targeting different risks within a router. However, in realistic deployment scenarios, a mitigation strategy rarely relies on a single security measure; rather, it consists of a complete set of interconnected security controls designed to collectively bring the overall system's risk down to an acceptable level. The combination of these controls does not necessarily imply a simple, additive reduction in risk. In certain configurations, combining a subset of controls may inadvertently result in the *negation of control impact*, or even escalate into an *expansion of attack surface*.

**From security controls to observable trustworthiness evidence**

As highlighted in the Security Control Management subcomponent of the CASTOR RA Engine (see Section 4.2.1.5), there are multiple types of security controls, a subset of which may require runtime evidence to be continuously monitored by the TAF's ATL evaluation. In fact, this continuous monitoring, reflected in the ATL value, ensures that the operational risk posture of each device remains at the acceptable level for a given trust property, as specified by the Security Administrator. The challenge of dealing with the interplay between security controls within a mitigation strategy is also reflected in the corresponding types of evidence that need to be collected throughout the operational lifecycle of the network topology. Therefore, this leads us to two core considerations when specifying in the Trust Policy the set of evidence that will form the ATL trust opinion for a given context (i.e., trust proposition as defined in D4.1 [6]).

First, different types of evidence must be collected at varying frequencies to accurately verify the existence and successful enforcement of security controls. For instance, trustworthiness evidence regarding Secure Boot may only be required during the initial deployment of the TNDE platform. In contrast, evidence confirming CFI validation must be collected much more frequently, such as being dynamically triggered upon each reconfiguration of the Routing Information Base (RIB) - irrespective of whether the update originates from the Orchestration Layer or the highly dynamic, frequency-intensive forwarding plane. All these intricacies need to be carefully reflected in both the ATL calculation process (e.g., Section 3.2 discusses the need for adjusting the confidence of historical ATL evaluations over time) and the RTL methodology. This will ensure a common semantic layer between these two core dimensions of the CASTOR Trust Assessment Framework so as to enable an accurate trust decision outcome.

Second, complex mitigation strategies may generate a multitude of evidence relating to the exact same behaviour evaluated by the CASTOR TAF during runtime. It is worth highlighting that the mere volume of reported evidence does not necessarily yield a proportional increase in confidence within an ATL trust opinion. In Subjective Logic, highly correlated or dependent evidence does not contribute new knowledge in the monitored system. Therefore, this reduction in epistemic uncertainty can only be achieved if the provided types of evidence result in coalescing knowledge, allowing the TAF instance to increase its systemic understanding rather than just processing overlapping data.

**Towards the systematic modelling of transition and state uncertainty in cascading attacks**

Building upon these foundational mechanisms established in this first version of the CASTOR RAE, our immediate next steps will focus on advancing the engine's probabilistic modelling capabilities. While the current implementation successfully bridges design-time RTL constraints and runtime ATL evidence through topology-aware Monte Carlo simulations, these "what-if" scenarios serve primarily as a crucial stepping stone. Moving forward, in the second release of the CASTOR Risk Assessment Engine, the goal is to evaluate how this stochastic baseline can be formalized into a Markov Decision Process. Transitioning to an MDP framework will allow for the systematic modelling of state and transition uncertainties across cascading attack paths. Ultimately, this advanced mathematical foundation will enable the CASTOR RAE to autonomously evaluate the complex trade-offs between the dynamic likelihood of an attack

taking place and the operational cost of deploying the appropriate security mitigation measures.

# Chapter 5

# Optimization Engine

## 5.1 Overview of the Optimization Approach

Within the CASTOR framework, the derivation of trusted routing paths is formulated as an optimization problem that jointly considers network performance metrics and trust-related attributes. Modern communication infrastructures are characterized by increasing scale, heterogeneity, and dynamic trust conditions, which significantly complicate routing decisions. Rather than optimizing a single network metric, the routing process must simultaneously account for multiple, often conflicting objectives related to performance, reliability, and trustworthiness. Consequently, the problem naturally takes the form of a multi-objective combinatorial optimization task.

To address this challenge, CASTOR introduces the *Optimization Engine*, a core component responsible for deriving routing recommendations that satisfy both network-level and trust-level requirements. The Optimization Engine operates on information provided by the Trust Assessment Framework (TAF), which continuously evaluates the trustworthiness of network entities and communication links based on collected evidence. By combining these trust assessments with traditional network attributes, the Optimization Engine identifies routing paths that satisfy the required trust levels while maintaining acceptable network performance.

From a computational perspective, the routing task can therefore be formulated as a multi-objective optimization problem defined over the network topology. The objective is to determine a routing configuration that minimizes a set of cost functions associated with network and trust attributes while satisfying structural constraints that ensure the validity of the selected path.

As a baseline approach, CASTOR considers classical multi-objective routing algorithms. In particular, a multi-objective extension of Dijkstra's algorithm is employed to compute Pareto-optimal routing paths under multiple criteria. Following the strategy outlined in Deliverable D4.1, the initial evaluation focuses on a bi-objective formulation of the routing problem. Restricting the analysis to two objectives allows for a clear representation of the Pareto frontier while maintaining manageable computational complexity. In this setting, the bi-objective Dijkstra algorithm provides the exact set of Pareto-optimal paths, which serves as a reference for evaluating alternative optimization methods.

Beyond classical algorithms, CASTOR also investigates quantum-inspired optimization approaches for solving the routing problem. To enable the use of such methods, the routing problem is reformulated as a *Quadratic Unconstrained Binary Optimization (QUBO)* problem, where routing decisions are encoded through binary variables and routing constraints are incorporated into the objective function via penalty terms. The QUBO formulation can subsequently be mapped to an equivalent *Ising Hamiltonian*, enabling the application of Ising-based optimization techniques such as Simulated Bifurcation.

The following sections present the formal definition of the routing optimization problem, introduce the considered optimization algorithms—namely the multi-objective Dijkstra algorithm and the Simulated Bi-

furcation approach—and provide a comparative evaluation of their performance. This analysis aims to assess the suitability of these methods and support the selection of the most appropriate optimization strategy for achieving the objectives of the CASTOR Optimization Engine.

## 5.2 Bi-objective Routing Optimization

Based on the above considerations, the routing task can be formally expressed as a multi-objective optimization problem defined over the network topology. As a first step toward this general formulation, we consider the bi-objective case, which allows for a clear and tractable analysis of the routing problem and the associated Pareto frontier. The objective is to determine routing paths between a source node and a destination node that jointly optimize criteria related to network performance and trustworthiness.

Let $G = (V, E)$ denote the directed graph representing the network, where $V$ is the set of nodes and $E$ is the set of directed edges. For each edge $(i, j) \in E$, two cost components are associated representing the values of two routing objectives.

For a given path $p$, the objective functions are defined as

$$f_1(p) = \sum_{(i,j)\in p} c_{ij}^{(1)} \tag{5.1}$$

$$f_2(p) = \sum_{(i,j)\in p} c_{ij}^{(2)} \tag{5.2}$$

where $c_{ij}^{(1)}$ and $c_{ij}^{(2)}$ denote the edge costs associated with the first and second objective respectively.

The routing problem therefore consists of determining paths that minimize the vector objective

$$f(p) = (f_1(p), f_2(p)). \tag{5.3}$$

### 5.2.1 Pareto optimality

In multi-objective optimization problems the solution is generally not unique. Instead, there exists a set of solutions representing different trade-offs between the considered objectives. Improving the value of one objective typically leads to the degradation of another, which means that no single solution can simultaneously minimize all objectives. For this reason, the goal is to identify the set of *Pareto-optimal* paths, also referred to as efficient solutions, which correspond to routing configurations for which no objective can be improved without worsening at least one of the others.

Given two feasible paths $p$ and $p'$, path $p'$ is said to dominate path $p$ if

$$f_1(p') \leq f_1(p), \qquad f_2(p') \leq f_2(p), \tag{5.4}$$

with at least one strict inequality.

A path $p$ is considered Pareto-optimal (or efficient) if there exists no other feasible path $p'$ that dominates it. The set of objective vectors corresponding to all efficient paths defines the Pareto frontier of the routing problem.

### 5.2.2 Bi-objective Dijkstra algorithm

To compute the set of Pareto-optimal routing paths, CASTOR employs a bi-objective extension of the classical Dijkstra shortest-path algorithm.

In contrast to the single-objective version, which stores a single distance value per node, the bi-objective version maintains sets of non-dominated labels representing alternative trade-offs between the objective functions.

Each label associated with node $i$ is defined as

$$\ell_i = (f_1, f_2, \pi) \tag{5.5}$$

where $f_1, f_2$ are the accumulated objective values of the path from the source node to node $i$, and $\pi$ denotes the predecessor node in the corresponding path.

Let $L(i)$ denote the set of labels associated with node $i$. A newly generated label $\ell = (f_1, f_2)$ is accepted only if it is not dominated by any existing label in $L(i)$. Formally, the label is discarded if there exists $\ell' \in L(i)$ such that

$$f_1(\ell') \leq f_1(\ell), \qquad f_2(\ell') \leq f_2(\ell). \tag{5.6}$$

If the new label dominates existing labels, those labels are removed from $L(i)$.

The algorithm iteratively expands labels starting from the source node. If a label

$$\ell_i = (f_1, f_2) \tag{5.7}$$

is associated with node $i$, then for every successor node $j$ connected through edge $(i, j)$ a new label is generated as

$$\ell_j = \left(f_1 + c_{ij}^{(1)},\ f_2 + c_{ij}^{(2)}\right). \tag{5.8}$$

The candidate label is then subjected to the dominance test and retained only if it represents a non-dominated partial path.

The algorithm continues expanding labels until no additional non-dominated labels can be generated. The set of labels associated with the destination node $t$

$$L(t) \tag{5.9}$$

contains the objective vectors corresponding to all Pareto-optimal routing paths. These solutions represent the complete set of efficient trade-offs between the considered routing objectives.

## 5.3 Simulated Bifurcation

Simulated Bifurcation (SB) is a quantum-inspired optimization algorithm derived from the classical simulation of adiabatic evolutions in nonlinear Hamiltonian systems exhibiting bifurcation phenomena. The method is inspired by quantum adiabatic optimization with nonlinear oscillators, but it operates entirely within a classical-mechanical framework. SB is designed to solve binary quadratic optimization problems, which can be expressed either in the form of a Quadratic Unconstrained Binary Optimization (QUBO)

problem or equivalently as an Ising model. In the Ising formulation, the objective is to find the spin configuration $s_i \in \{-1, +1\}$ minimizing the Ising energy

$$E_{\text{Ising}} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} J_{ij} s_i s_j. \tag{5.10}$$

In SB, each spin variable is represented by a continuous dynamical variable $x_i(t)$ evolving under a time-dependent Hamiltonian system. During the evolution, bifurcations destabilize the trivial equilibrium and drive the variables toward the boundaries $x_i = \pm 1$, which correspond to the discrete spin states of the Ising model. The final solution is obtained through the projection $s_i = \text{sgn}(x_i)$, yielding a candidate low-energy configuration of the target Ising Hamiltonian.

A detailed description of the Simulated Bifurcation algorithm and its variants has been presented in Deliverable D4.1. In the present deliverable we summarize the main principles relevant to the CASTOR implementation and highlight the specific variant adopted in our experiments.

Depending on how the underlying dynamical evolution is implemented and controlled, SB admits several algorithmic variants that trade off adiabaticity, convergence speed, and robustness. In particular, three main flavors are commonly considered: adiabatic Simulated Bifurcation (aSB) [14], ballistic Simulated Bifurcation (bSB), and discrete Simulated Bifurcation (dSB) [13].

In the context of the CASTOR project, the solver implementation is based on ballistic Simulated Bifurcation enhanced with thermal fluctuations. The addition of a controlled heating term increases the dynamical energy of the system and helps the evolution escape shallow local minima of the Ising energy landscape. This mechanism improves the exploration capability of the algorithm while preserving the highly parallel structure of the SB dynamics.

**Definition of Simulated Bifurcation parameters** The dynamics of both ballistic and discrete Simulated Bifurcation are governed by three global parameters: $a_0$, $a(t)$, and $c_0$.

The parameter $a(t)$ is a time-dependent control parameter that is increased from zero during the evolution. Its role is to induce bifurcations in the system by destabilizing the trivial equilibrium at $x_i = 0$, thereby driving the dynamical variables toward the discrete attractors corresponding to Ising spin states.

The parameter $a_0 > 0$ is a constant that sets the characteristic time scale of the Hamiltonian dynamics and determines the coupling between the position variables $x_i$ and their conjugate momenta $y_i$. In the implementation considered in this work, $a_0$ is fixed to a constant value.

The parameter $c_0 > 0$ scales the interaction term and encodes the strength of the Ising couplings into the Simulated Bifurcation dynamics. It determines the relative influence of the coupling matrix $J_{ij}$ on the system evolution.

**Ballistic Simulated Bifurcation (bSB)** Ballistic Simulated Bifurcation is a nonadiabatic variant designed to improve convergence speed and solution quality. Instead of slowly tracking bifurcating minima, the system undergoes rapid, momentum-driven dynamics that push variables toward discrete boundaries. Perfectly inelastic constraints force variables to settle at their binary limits, ensuring convergence to stable local minima of the Ising energy.

In the implementation used in this work, ballistic SB is augmented with a heating term that introduces controlled thermal fluctuations in the momentum dynamics. This modification increases the kinetic energy of the system and helps avoid trapping in local minima during the evolution.

**Hamiltonian formulation** The Hamiltonian of the ballistic SB system is defined as

$$H_{\mathrm{bSB}} = \frac{a_0}{2} \sum_{i=1}^{N} y_i^2 + V_{\mathrm{bSB}}, \tag{5.11}$$

with the potential term

$$V_{\mathrm{bSB}} = -\frac{a_0 - a(t)}{2} \sum_{i=1}^{N} x_i^2 - \frac{c_0}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} J_{i,j}\, x_i x_j, \quad \text{for } |x_i| \leq 1. \tag{5.12}$$

Outside this domain, the potential is defined as

$$V_{\mathrm{bSB}} = \infty, \tag{5.13}$$

corresponding to perfectly inelastic walls at $x_i = \pm 1$.

**Continuous-time dynamics with heating** For each spin $i = 1, \ldots, N$, the equations of motion are given by

$$\dot{x}_i = a_0\, y_i, \tag{5.14}$$

$$\dot{y}_i = -\big[a_0 - a(t)\big] x_i + c_0 \sum_{j=1}^{N} J_{i,j}\, x_j + \gamma y_i. \tag{5.15}$$

Here $\gamma > 0$ is a constant heating rate that injects energy into the momentum variables and enhances exploration of the energy landscape.

**Discrete-time update** The numerical integration proceeds in two stages. First, the Hamiltonian part of the dynamics is updated using the symplectic Euler scheme:

$$\tilde{y}_i = y_i^{(k)} + \left( -\big[a_0 - a(t_k)\big] x_i^{(k)} + c_0 \sum_{j=1}^{N} J_{i,j}\, x_j^{(k)} \right) \Delta t, \tag{5.16}$$

$$\tilde{x}_i = x_i^{(k)} + a_0\, \tilde{y}_i\, \Delta t. \tag{5.17}$$

If $|\tilde{x}_i| > 1$, perfectly inelastic boundary conditions are enforced:

$$x_i^{(k+1)} = \begin{cases} \tilde{x}_i, & |\tilde{x}_i| \leq 1 \\ \mathrm{sgn}(\tilde{x}_i), & |\tilde{x}_i| > 1 \end{cases} \tag{5.18}$$

$$\hat{y}_i = \begin{cases} \tilde{y}_i, & |\tilde{x}_i| \leq 1 \\ 0, & |\tilde{x}_i| > 1 \end{cases} \tag{5.19}$$

Finally, the heating contribution is applied to the momentum variables:

$$y_i^{(k+1)} = \hat{y}_i + \gamma y_i^{(k)} \Delta t. \tag{5.20}$$

**Extraction of the spin configuration** Ballistic Simulated Bifurcation operates on continuous dynamical variables $x_i(t) \in [-1, 1]$, whose evolution is governed by the SB dynamical equations and subject to inelastic boundary constraints. The design of the SB potential ensures that stable attractors of the dynamics are located at the boundaries $x_i = \pm 1$, which correspond to discrete Ising spin states. The design of the SB potential ensures that stable attractors of the dynamics are located at the boundaries $x_i = \pm 1$, which correspond to discrete Ising spin states.

Upon termination of the dynamical evolution, the final Ising spin configuration is obtained through a deterministic projection given by

$$s_i = \mathrm{sgn}(x_i), \tag{5.21}$$

where $s_i \in \{-1, +1\}$ denotes the Ising spin associated with variable $i$. This projection maps the continuous SB state to a discrete spin configuration that defines a candidate solution of the Ising Hamiltonian introduced in Section 5.3.3.

In ballistic Simulated Bifurcation, the inelastic boundary conditions enforce $|x_i| = 1$ at convergence, guaranteeing that the extracted spin configuration corresponds to a stable local minimum of the Ising energy. When the heating term is included in the momentum dynamics, the additional energy injected into the system helps the trajectories escape shallow local minima during the evolution, while the final projection rule remains unchanged.

### 5.3.1 Binary Optimization Problem Formulation

In the CASTOR framework, trusted path routing is formulated as a multi-objective combinatorial optimization problem defined over a network topology. The communication network is represented as a directed graph

$$G = (V, E) \tag{5.22}$$

where $V$ denotes the set of network nodes and $E$ the set of directed communication links between nodes. Each edge $(i, j) \in E$ is associated with network and trust-related attributes that characterize the quality and security properties of the link.

The routing task consists of determining a path between a source node $S$ and a destination node $D$ that satisfies routing constraints while optimizing multiple performance and trust objectives.

Routing decisions are represented using binary variables

$$x_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \text{ belongs to the selected path} \\ 0 & \text{otherwise.} \end{cases} \tag{5.23}$$

The objective functions can therefore be written as

$$f_1(x) = \sum_{(i,j)\in E} c_{ij}^{(1)} x_{ij} \tag{5.24}$$

$$f_2(x) = \sum_{(i,j)\in E} c_{ij}^{(2)} x_{ij}. \tag{5.25}$$

**Routing Constraints:**The routing constraints ensuring a valid path between the source node $S$ and the destination node $D$ can be expressed through flow conservation conditions

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1 & i = S \\ -1 & i = D \\ 0 & i \in V \setminus \{S, D\}. \end{cases} \tag{5.26}$$

The feasible set of routing configurations is therefore

$$x_{ij} \in \{0, 1\}, \quad (i, j) \in E. \tag{5.27}$$

These equations express the standard feasibility conditions, ensuring that the selected edges form a valid path between the source node $S$ and the destination node $D$. However, the compact formulation in Eq. (5.26) does not explicitly enforce degree constraints at the source and destination nodes. In particular, it does not prevent configurations where additional edges enter the source node or leave the destination node. To simplify the formulation and ensure that the selected edges form a proper path between the source and destination, we explicitly impose degree restrictions on these nodes. Specifically, the source node is constrained to have exactly one outgoing edge and the destination node exactly one incoming edge. Under these assumptions, the routing constraints can be written as follows.

For the source node:

$$\sum_{j} x_{S,j} = 1 \tag{5.28}$$

For the destination node:

$$\sum_{i} x_{i,D} = 1 \tag{5.29}$$

For all intermediate nodes:

$$\sum_{j} x_{i,j} - \sum_{k} x_{k,i} = 0 \tag{5.30}$$

These constraints ensure that exactly one edge leaves the source node, exactly one edge enters the destination node, and that the number of incoming and outgoing edges is balanced for every intermediate node. As a result, the selected edges form a continuous path between the source and destination nodes.

### 5.3.2 Transformation to QUBO

To enable the use of quantum-inspired optimization algorithms such as Simulated Bifurcation, the constrained routing problem must be reformulated as a Quadratic Unconstrained Binary Optimization (QUBO) problem.

In the QUBO formulation, all constraints are incorporated into the objective function through quadratic penalty terms. The resulting QUBO objective function is defined over binary variables $x_i \in \{0, 1\}$ as

$$H(x) = \sum_{i \neq j} Q_{ij} x_i x_j + \sum_{i} q_i x_i \tag{5.31}$$

where $Q_{ij}$ represents pairwise interaction coefficients between variables and $q_i$ denotes linear coefficients.

To enforce routing feasibility, penalty terms corresponding to the routing constraints are added to the objective function. The resulting QUBO formulation becomes

$$H(x) = f(x) + P_1 \left( \sum_j x_{S,j} - 1 \right)^2 + P_2 \left( \sum_i x_{i,D} - 1 \right)^2 + P_3 \sum_{i \neq S,D} \left( \sum_j x_{i,j} - \sum_k x_{k,i} \right)^2 \tag{5.32}$$

where $P_1$, $P_2$, and $P_3$ are positive penalty coefficients enforcing the routing constraints. In this work, we consider equal penalty weights for all routing constraints, i.e.

$$P_1 = P_2 = P_3 = P. \tag{5.33}$$

The penalty coefficient $P$ is selected proportional to the total cost of the network in order to ensure that violations of routing constraints are sufficiently penalized. Specifically, we define

$$P = \sum_{(i,j) \in E} c_{ij}. \tag{5.34}$$

With this choice, infeasible routing configurations incur a penalty that dominates the contribution of the objective function, ensuring that valid routing paths correspond to lower-energy states of the optimization problem.

### 5.3.3 Mapping from QUBO to the Ising Model

The QUBO formulation can be directly mapped to an equivalent Ising model representation, which is widely used in physics-inspired optimization algorithms.

In the Ising formulation, binary variables $x_i \in \{0, 1\}$ are transformed into spin variables

$$\sigma_i \in \{-1, +1\} \tag{5.35}$$

using the transformation

$$\sigma_i = 2x_i - 1. \tag{5.36}$$

Substituting this relation into the QUBO energy function yields an equivalent Ising Hamiltonian

$$H(\sigma) = -\frac{1}{2} \sum_{i,j} J_{ij} \sigma_i \sigma_j + \sum_i h_i \sigma_i. \tag{5.37}$$

Expanding the QUBO formulation in terms of spin variables results in the following relations between the QUBO coefficients and the Ising parameters.

The coupling coefficients are given by

$$J_{ij} = -\frac{Q_{ij}}{2} \tag{5.38}$$

while the local bias terms are

$$h_i = \frac{q_i}{2} + \frac{1}{4}\sum_{j \neq i}(Q_{ij} + Q_{ji}). \tag{5.39}$$

The optimal solution of the routing problem corresponds to the ground state of this Ising Hamiltonian. Once the spin configuration minimizing the energy is obtained, the original binary variables can be recovered through the inverse transformation.

### 5.3.4 Scalarization Strategy for Multi-Objective QUBO/Ising Optimization

Although QUBO and Ising formulations provide a powerful and unified framework for combinatorial optimization, they are inherently designed to address single-objective optimization problems, where the goal is to minimize a single scalar energy function. In contrast, the routing problem considered in CASTOR involves multiple objectives related to network performance and trust, which must be optimized simultaneously. Such multi-objective optimization problems typically lead to a set of trade-off solutions rather than a single optimal configuration.

To address this challenge within the QUBO framework, we adopt a scalarization strategy that transforms the multi-objective routing problem into a single-objective formulation compatible with QUBO-based solvers. In this context, the routing decision must simultaneously account for multiple network and trust attributes. Let $f_k(x)$ denote the cost associated with the $k$-th objective function evaluated over the routing configuration $x$, where $x$ represents the binary routing variables introduced in the previous section.

The multiple objectives are combined into a single scalar objective function using a weighted-sum formulation

$$f(x) = \sum_{k=1}^{K} w_k f_k(x), \tag{5.40}$$

where $w_k$ denotes the weight associated with the $k$-th objective function and the weights satisfy

$$\sum_{k=1}^{K} w_k = 1. \tag{5.41}$$

This scalarization enables the multi-objective routing problem to be expressed as a single-objective optimization problem compatible with the QUBO formulation, while preserving the relative importance of the individual objectives. The resulting optimization problem consists of minimizing the scalar objective function $f(x)$ subject to the routing constraints defined above.

## 5.4 Standalone Evaluation

This section presents the experimental evaluation of the proposed routing optimization framework. The objective of the evaluation is to assess the performance of the considered optimization approaches, namely the bi-objective Dijkstra algorithm and the ballistic Simulated Bifurcation (bSB) algorithm, in identifying feasible routing configurations and approximating the Pareto-optimal solution set of the routing problem.

Experiments were conducted on two categories of network topologies: synthetic grid graphs and real-world network topologies. This combination allows us to analyze the behavior of the algorithms under both controlled structural conditions and realistic network environments.

The evaluation follows a twofold methodology. First, the bi-objective Dijkstra algorithm is employed as an exact reference method to characterize the structure of the routing problem on the considered network topologies. Since this algorithm computes the complete Pareto-optimal solution set, it enables a quantitative analysis of the intrinsic properties of the routing landscape and the trade-offs between the considered objectives. In particular, this analysis examines statistical characteristics of the Pareto fronts, including their size distribution, the typical path lengths of Pareto-optimal routes, and the relative participation of network nodes across Pareto solutions. These properties provide insight into the complexity and diversity of feasible optimal routing configurations.

In the second stage of the evaluation, the ballistic Simulated Bifurcation (bSB) algorithm is assessed by comparing the solutions it produces with the exact Pareto-optimal solutions obtained through the bi-objective Dijkstra algorithm. This comparison allows us to evaluate the ability of bSB to identify valid routing configurations and to approximate the Pareto-optimal solution set under the same network conditions.

### 5.4.1 Datasets

The performance of both the bi-objective Dijkstra algorithm and the ballistic Simulated Bifurcation (bSB) method is evaluated using a collection of synthetic and real-world network topologies. The datasets were selected to provide both controlled structural properties and realistic network characteristics.

**Grid Graphs.** Synthetic grid graphs are used to analyze the behavior of the algorithms under controlled conditions. In a grid graph, nodes are arranged in a two-dimensional lattice and each node is connected to its neighboring nodes. Such graphs provide a structured topology where the network size can be systematically increased while maintaining regular connectivity.

In our experiments, square grid graphs of increasing size were considered, with side lengths equal to $5$, $7$, $10$, and $15$. The structural characteristics of these instances are summarized in Table 5.1. In the table, $|V|$ and $|E|$ denote the number of vertices and directed edges of the graph, respectively, while $\langle k \rangle$ represents the average node degree, defined as the average number of incident edges per vertex, accounting for both incoming and outgoing connections. The use of grid graphs allows us to study the scalability of the optimization algorithms as the network size increases.

| Instance | $\|\mathbf{V}\|$ | $\|\mathbf{E}\|$ | $\langle \mathbf{k} \rangle$ |
|---|---|---|---|
| Grid $5 \times 5$ | 25 | 40 | 3.20 |
| Grid $7 \times 7$ | 49 | 84 | 3.43 |
| Grid $10 \times 10$ | 100 | 180 | 3.60 |
| Grid $15 \times 15$ | 225 | 420 | 3.73 |

Table 5.1: Structural characteristics of the synthetic grid datasets.

**Real-world Network Topologies.**

In addition to synthetic graphs, the algorithms are evaluated on real network topologies which do not typically have the predictable properties of grids. Active measurements and the relevant extraction of intra-domain ISP topologies was one important threat of networking research during the 2000s and the 2010s; lately, importance has been shifted towards the network edge with content distribution networks (driven by giants such as Akamai, Google, Netflix) while software-defined networking suggest a network dynamicity that goes beyond traditional static topologies.

To evaluate the considered algorithms, we have used three sets of intradomain topologies of diverse characteristics (e.g., size, degree distribution). Two of them relate to measurement projects while a

third one contains capacitated topologies at the router level, collected directly by network operators of academic and research networks. Their characteristics are briefly discussed below:

- *CAIDA topologies*: Those datasets were collected in autumn 2011 by CAIDA [5] using traceroute probes to randomly-chosen destinations from 54 monitors worldwide. Parsing the generated file that heuristically assigns an AS to each node found, the router-to-AS ownership is determined and subsequently topologies of the nodes operated by certain ASes have been extracted out of the raw data files.
- *Rocketfuel topologies*: The Rocketfuel technique [22] has been shown to collect high-fidelity router-level maps of ISPs and therefore has been widely used despite its relatively old publication. The considered dataset includes measurements from 800 vantage points serving as traceroute sources. Innovative techniques such as BGP directed probing and IP identifiers, have been applied to reduce the number of probes and tackle the alias resolution (i.e., discover the different interface IP addresses that belong to the same router), respectively.
- *Topology Zoo topologies*: Whereas previous measurement studies employ a number of route discovery tools to reveal the Internet connectivity the Topology Zoo gathers the maps of more than 140 real-world topologies directly from the network operators [17]. As the resulting maps (topologies and associated attributes) come from the owner and/or manager of the network, they are claimed to reflect an accurate network view circumventing any errors due to biases of measurement techniques. We have selected a subset of the largest router-levels snapshots retrieved during 2008-11.

Table 5.2 summarizes the structural properties of the real network instances considered in this study.

Table 5.2: Real network topologies used in the evaluation

| Dataset | Topology | $\|V\|$ | $\|E\|$ | $\langle k \rangle$ |
|---|---|---|---|---|
| CAIDA | AS1299 | 3820 | 11784 | 3.09 |
| CAIDA | AS1557 | 6598 | 16290 | 2.47 |
| CAIDA | AS174 | 14413 | 44666 | 3.10 |
| CAIDA | AS4134 | 81121 | 321846 | 3.97 |
| CAIDA | AS6181 | 1831 | 4362 | 2.38 |
| CAIDA | AS701 | 18281 | 50672 | 2.77 |
| CAIDA | AS786 | 2259 | 5110 | 2.26 |
| RocketFuel | AS1221 | 2515 | 6084 | 2.42 |
| RocketFuel | AS1239 | 7303 | 19786 | 2.71 |
| RocketFuel | AS1755 | 295 | 1087 | 3.69 |
| RocketFuel | AS2914 | 4607 | 15115 | 3.28 |
| RocketFuel | AS3257 | 411 | 1306 | 3.18 |
| RocketFuel | AS3356 | 1620 | 13483 | 8.33 |
| RocketFuel | AS3967 | 353 | 1640 | 4.65 |
| RocketFuel | AS4755 | 41 | 136 | 3.32 |
| RocketFuel | AS7018 | 9418 | 23344 | 2.48 |
| Topology Zoo | Belnet2003 | 23 | 101 | 4.39 |
| Topology Zoo | Belnet2006 | 23 | 105 | 4.57 |
| Topology Zoo | Bren | 37 | 113 | 3.06 |

| Dataset | Topology | $\|V\|$ | $\|E\|$ | $\langle k \rangle$ |
|---|---|---|---|---|
| Topology Zoo | Carnet | 44 | 130 | 2.96 |
| Topology Zoo | Geant2009 | 34 | 138 | 4.06 |
| Topology Zoo | Janetlense | 20 | 88 | 4.40 |
| Topology Zoo | KentmanJan2011 | 38 | 114 | 3.00 |
| Topology Zoo | Myren | 37 | 115 | 3.11 |
| Topology Zoo | Niif | 36 | 118 | 3.28 |
| Topology Zoo | Renater2010 | 43 | 155 | 3.61 |
| Topology Zoo | Sanet | 43 | 133 | 3.10 |
| Topology Zoo | SwitchL3 | 42 | 168 | 4.00 |
| Topology Zoo | Uninett2010_max | 74 | 276 | 3.73 |
| Topology Zoo | Uninett2010_mean | 74 | 276 | 3.73 |
| Topology Zoo | Uninett2010_min | 74 | 276 | 3.73 |
| Topology Zoo | Uninett2011_max | 69 | 261 | 3.79 |
| Topology Zoo | Uninett2011_mean | 69 | 261 | 3.79 |
| Topology Zoo | Uninett2011_min | 69 | 261 | 3.79 |

### 5.4.2 Edge Weight Generation

Following the definition of the network datasets, we now describe the generation of the edge attributes associated with the two objectives considered in the routing problem. For each edge in the network, two cost values are assigned corresponding to the two optimization objectives.

The first objective edge-weight metrics were sampled from a Poisson distribution with parameter $\lambda = 20$. To ensure that all weights remain strictly positive, the sampled values are shifted by one unit.

The second objective metrics were generated using a mixture of two Beta distributions in order to introduce heterogeneous edge characteristics. Specifically, with probability $0.20$ the weight is drawn from a Beta$(2.5, 7.5)$ distribution, while with probability $0.80$ it is drawn from a Beta$(90, 10)$ distribution. This mixture produces a distribution in which most edges exhibit high reliability values, while a smaller fraction represents lower-quality links.

Together, these weight distributions create heterogeneous edge costs and induce meaningful trade-offs between the routing objectives. This setup allows us to evaluate the ability of the considered algorithms to explore the resulting Pareto-optimal solution space across the different network topologies described above.

### 5.4.3 Pareto Landscape Analysis Using the Bi-objective Dijkstra Algorithm

The first stage of the experimental evaluation focuses on analyzing the routing problem using the bi-objective Dijkstra algorithm as an exact reference method. Since this algorithm computes the complete set of Pareto-optimal paths between a given source–destination pair, it provides a reliable basis for conducting a quantitative study of the routing landscape across the considered network topologies.

For each generated problem instance, twenty source–destination routing queries are defined and solved using the bi-objective shortest-path algorithm. Each query produces a Pareto front containing all non-dominated routing alternatives between the selected endpoints. Rather than focusing on a single optimal path, the analysis examines the structural properties of these Pareto fronts in order to characterize the solution space of the routing problem.

In particular, the study evaluates the size of the Pareto fronts, defined as the number of non-dominated routing paths, which indicates how many alternative routing solutions exist under the two considered objectives. The lengths of the Pareto-optimal paths are also analyzed to understand how path complexity varies across different solutions and topologies. In addition, the frequency with which network nodes appear in Pareto-optimal paths is computed to identify nodes that play a central role in multi-objective routing.

By performing this analysis across networks of different sizes and structures, the experiments provide a quantitative characterization of how the number of routing alternatives and the properties of Pareto-optimal paths evolve with the size and connectivity of the underlying topology. This study therefore provides a reference description of the routing landscape on the considered datasets, which is subsequently used to evaluate the performance of the ballistic Simulated Bifurcation (bSB) algorithm in approximating the Pareto-optimal solution set.

Figure 5.1 summarizes the main statistical characteristics of the Pareto fronts obtained from the routing queries. The left panel shows the distribution of Pareto front sizes, indicating the number of non-dominated routing paths identified for each routing query. The middle panel presents the distribution of path lengths among the Pareto-optimal routes, providing insight into the complexity of the routing paths that form the Pareto frontier. Finally, the right panel reports the ranked frequency of node visits across all Pareto-optimal paths, highlighting the relative importance of network nodes in the set of efficient routing configurations.

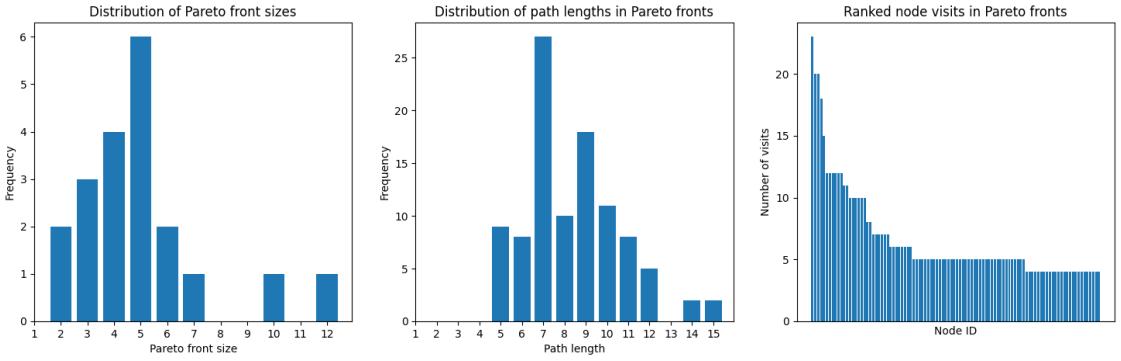



Figure 5.1: Indicative results obtained using the CAIDA-nets AS174 topology.

The distribution of Pareto front sizes shown in the left panel indicates that most routing queries produce relatively small sets of non-dominated solutions. In the majority of cases, the number of efficient routing paths lies between three and six, suggesting that although multiple routing alternatives exist under the considered objectives, the size of the efficient solution set remains limited for most source–destination pairs. This observation implies that the routing decision space is typically manageable even when multiple objectives are considered simultaneously.

The middle panel illustrates the distribution of path lengths among the Pareto-optimal routes. The results show that most efficient routing paths fall within a relatively narrow range of path lengths, with a moderate concentration around typical routing distances in the network. This behavior suggests that Pareto-optimal solutions tend to correspond to paths that are structurally similar in terms of hop count while still offering different trade-offs between the considered objectives.

Finally, the right panel reports the ranked frequency of node participation in Pareto-optimal paths. The distribution reveals that a small subset of nodes appears significantly more frequently in efficient routing paths, while most nodes participate only occasionally. This pattern highlights the presence of structurally important nodes that act as central transit points for multi-objective routing solutions within the network topology.

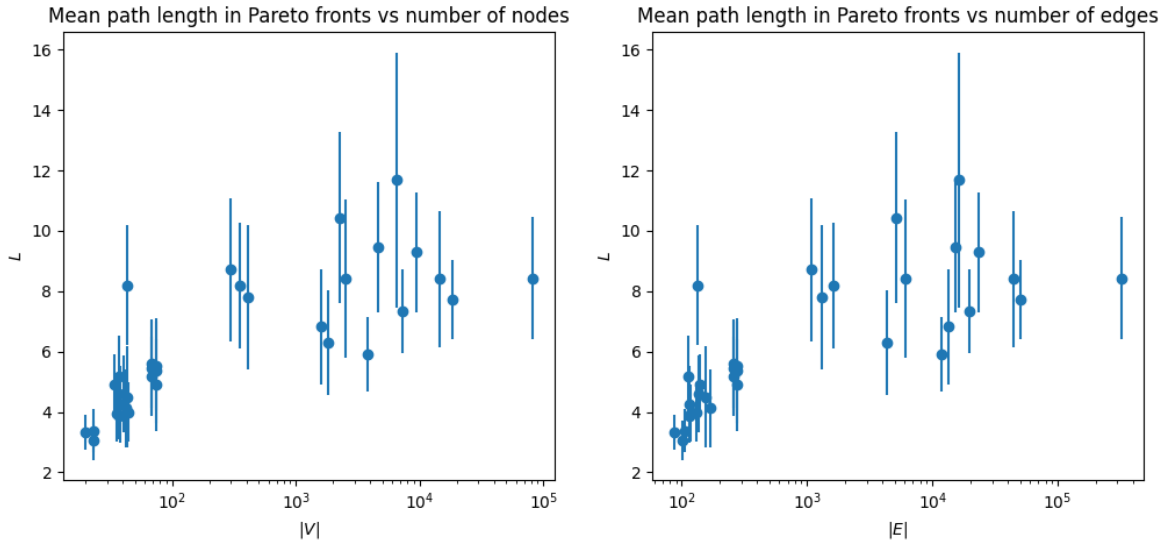



Figure 5.2: Mean path length as a function of nodes (left panel) and edges (right panel)

Figures 5.2 and 5.3 extend the previous analysis by examining how the structural characteristics of Pareto-optimal routing solutions evolve with the size and connectivity of the underlying network topology. In particular, Figure 5.2 shows the mean path length observed within Pareto fronts as a function of the number of nodes (left panel) and the number of edges (right panel). The results indicate that the average length of Pareto-optimal paths tends to increase with the size of the topology for smaller networks, but gradually stabilizes as the network grows larger, typically remaining within a range of approximately six to twelve hops. This behavior suggests that although larger networks provide more routing alternatives, Pareto-optimal paths do not grow proportionally in length, indicating that efficient trade-off solutions remain relatively compact in terms of hop count.

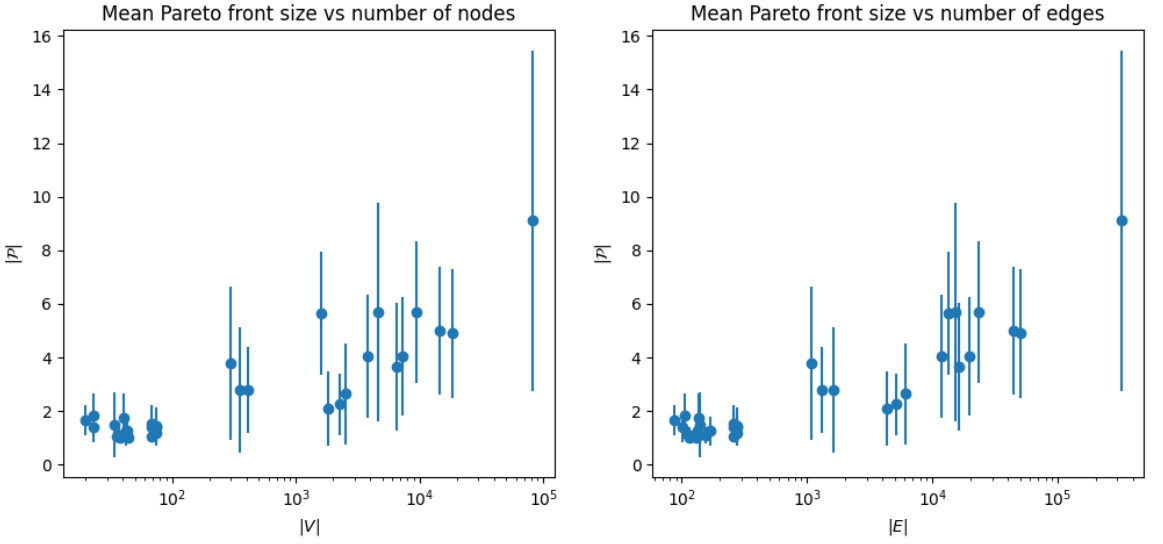



Figure 5.3: Mean Pareto front size as a function of nodes (left panel) and edges (right panel)

Figure 5.3 presents the corresponding evolution of the mean Pareto front size with respect to the number of nodes (left panel) and edges (right panel). The results show that the number of non-dominated routing alternatives generally increases with network size and connectivity, as larger and more densely connected topologies provide more possible routing combinations. Nevertheless, this growth remains moderate, with most instances producing relatively small Pareto fronts.

Taken together, these results indicate that while the number of routing alternatives tends to increase with the size and connectivity of the network, the Pareto fronts remain compact and structurally interpretable. Combined with the earlier observations on the distribution of path lengths and node participation frequencies, this suggests that multi-objective routing in the considered topologies typically yields a manageable set of efficient routing configurations. Such behavior is particularly desirable in practical routing scenarios, as it allows the optimization engine to explore meaningful trade-offs between performance and trust objectives without facing an excessively large solution space.

### 5.4.4 Evaluation of the Ballistic Simulated Bifurcation Algorithm

This subsection presents the experimental evaluation of the ballistic Simulated Bifurcation (bSB) algorithm for the considered routing optimization problem. The results are obtained using both synthetic grid-based network topologies and real-world network datasets in order to assess the performance and robustness of the method across different network structures.

#### 5.4.4.1 bSB Parameter Configuration

The parameters of the bSB algorithm are adapted to each problem instance in order to account for differences in graph size and edge density across the considered datasets. While the parameter $a_0$ remains fixed across all experiments, the integration step $\Delta t$, the final simulation time $t_{\text{final}}$, and the damping coefficient $\gamma$ are adjusted depending on the instance size. The values used in the experiments are summarized in Table 5.3.

The time-dependent parameter $a(t)$ appearing in the bSB Hamiltonian is implemented using the linear schedule

$$a(t) = \frac{t}{t_{\text{final}}},$$

which increases monotonically during the simulation horizon.

The parameter $a_0$ is fixed to $a_0 = 1$ in all experiments. The scaling parameter $c_0$ is chosen according to the coupling structure of the QUBO matrix as

$$c_0 = \frac{\kappa}{\langle J \rangle \sqrt{n}},$$

where $n$ denotes the number of spin variables (equal to the number of edges in the graph) and

$$\langle J \rangle = \sqrt{\frac{\sum_{i,j} J_{ij}^2}{n(n-1)}}.$$

The coefficient $\kappa$ depends on the dataset and is selected as

$$\kappa = \begin{cases} 0.8, & \text{for grid and RocketFuel topologies,} \\ 0.7, & \text{for Topology Zoo networks.} \end{cases}$$

Finally, the initial conditions for the spin variables are generated as

$$x_i(0) = 0, \qquad y_i(0) \sim \mathcal{U}(-0.01, 0.01),$$

for all spins $i$. The following table summarizes the parameter values used for each problem instance considered in the experiments.

| Dataset / Graph | $\Delta t$ | $t_{\text{final}}$ | $\gamma$ | $\kappa$ |
|---|---|---|---|---|
| Grid $5 \times 5$ | 0.5 | 1000 | 0.005 | 0.8 |
| Grid $7 \times 7$ | 0.5 | 5000 | 0.01 | 0.8 |
| Grid $10 \times 10$ | 1 | 9000 | 0.01 | 0.8 |
| Grid $15 \times 15$ | 1 | 10000 | 0.1 | 0.8 |
| RocketFuel AS1755 | 0.5 | 3000 | $10^{-5}$ | 0.8 |
| RocketFuel AS3257 | 0.5 | 3000 | $10^{-5}$ | 0.8 |
| RocketFuel AS3967 | 0.5 | 3000 | $10^{-5}$ | 0.8 |
| RocketFuel AS4755 | 0.25 | 1300 | $10^{-4}$ | 0.8 |
| Topology Zoo | 0.25 | 1300 | $10^{-4}$ | 0.7 |

Table 5.3: Instance-dependent parameters used in the bSB experiments.

#### 5.4.4.2 bSB Results on Grid Graphs

We begin the experimental evaluation by analyzing the behavior of bSB on synthetic grid graphs. As a first step, we study the effect of scalarization on the solutions returned by bSB. To assess the performance of the algorithm across all experiments, we employ three evaluation metrics defined as follows.

Let $N$ denote the total number of algorithm runs for a given scalarization tuple, $N_f$ the number of feasible solutions returned by bSB, and $P^\star$ the set of Pareto-optimal solutions obtained using the exact bi-objective Dijkstra algorithm. Furthermore, let $P_{\text{bSB}}$ denote the set of Pareto solutions identified by bSB.

- **Feasibility probability**

$$P_{\text{feasible}} = \frac{N_f}{N},$$

  which measures the probability that bSB returns a feasible source-to-destination path.

- **Pareto hit probability**

$$P_{\text{Pareto}} = \frac{N_p}{N},$$

  where $N_p$ denotes the number of runs that produce a Pareto-optimal solution. This metric quantifies how often bSB reaches a Pareto point.

- **Pareto coverage**

$$\text{PC} = \frac{|P_{\text{bSB}} \cap P^\star|}{|P^\star|},$$

  which measures the fraction of the exact Pareto front recovered by the solutions returned by bSB.

**Scalarization effect:** Using these metrics, we investigate the effect of scalarization on the solutions produced by bSB. The experiment was conducted on a $7 \times 7$ grid graph instance. To this end, we generated a set of $19$ scalarization tuples in the range $(0, 1)$ with step size $0.05$. For each tuple pair, bSB was executed $5000$ times. The obtained solutions were first classified as *feasible* or *infeasible*. A solution is considered feasible if it corresponds to a continuous path from the source node $S$ to the destination node $D$, whereas infeasible solutions correspond to broken paths that do not connect the two endpoints.

Feasible solutions were further classified into Pareto-optimal and dominated solutions, using the exact Pareto set obtained from the bi-objective Dijkstra algorithm as reference.

The results are summarized in Figure 5.4. The upper row of the figure reports the three metrics as a function of the scalarization weights, while the bottom row illustrates the distribution of the obtained solutions in the objective space. In panels (c) and (d), the orange disks correspond to the exact Pareto-optimal solutions obtained using the bi-objective Dijkstra algorithm, blue crosses denote Pareto solutions identified by bSB, and gray points correspond to dominated solutions returned by bSB. The plots correspond to two representative scalarization tuples: (c) $(0.05, 0.95)$ and (d) $(0.5, 0.5)$.

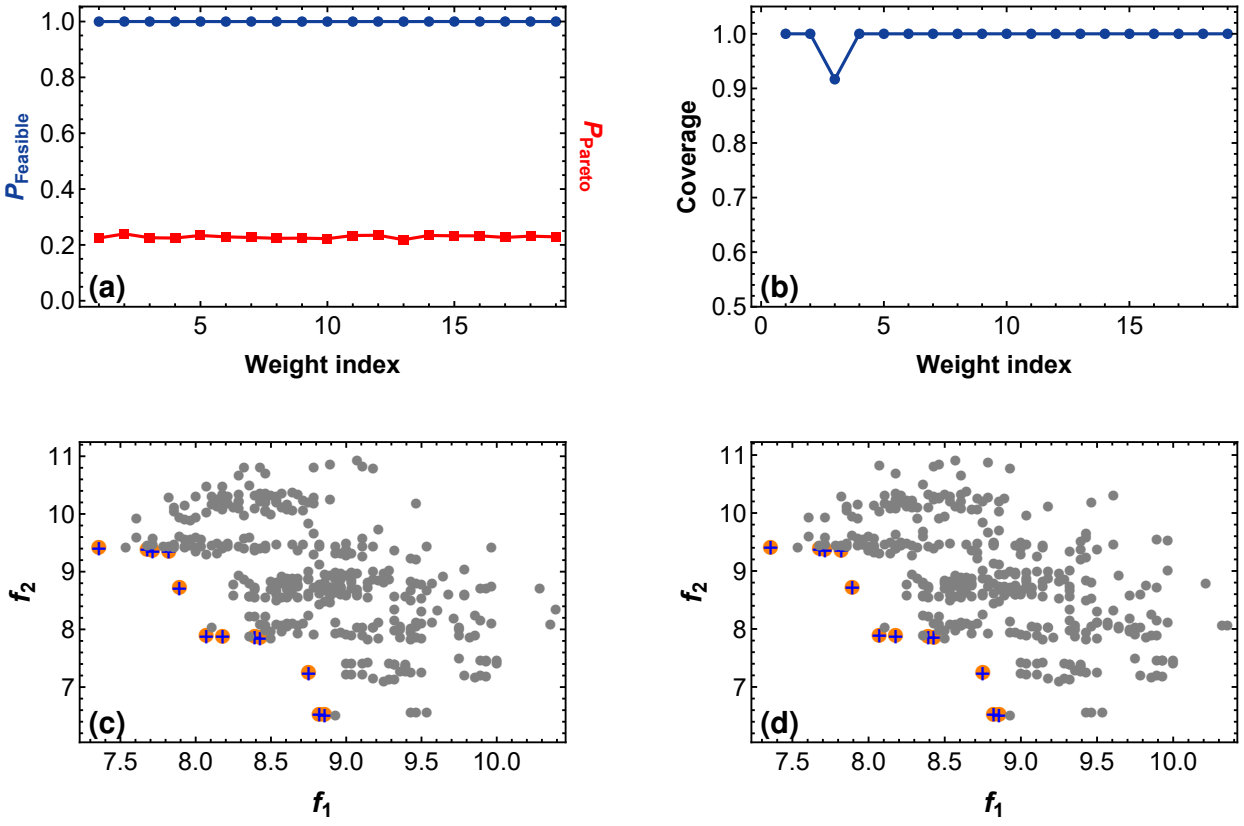



Figure 5.4: Effect of scalarization on bSB performance for the $7 \times 7$ grid instance. The upper panels report feasibility probability, Pareto hit probability, and Pareto coverage as a function of the scalarization weights, while the lower panels show the resulting solution distributions in the objective space.

Figure 5.4 highlights that the scalarization weights have only a limited impact on the ability of bSB to identify Pareto-optimal solutions. As shown in panel (a), the probability of obtaining a feasible path remains close to one across all scalarization tuples, indicating that the algorithm consistently produces valid source-to-destination routes. At the same time, the probability of reaching a Pareto-optimal solution remains approximately constant, suggesting that the scalarization weights do not significantly bias the search toward specific regions of the objective space. Panel (b) further confirms this observation: the Pareto coverage remains very close to unity for almost all scalarization tuples, indicating that the set of solutions returned by bSB is able to recover nearly the entire exact Pareto front obtained using the bi-objective Dijkstra algorithm. Only minor fluctuations are observed for a small number of weight indices. The scatter plots in panels (c) and (d) illustrate the distribution of solutions in the objective space for two representative scalarization tuples. In both cases, the Pareto-optimal points identified by bSB (blue crosses) coincide with the exact Pareto solutions (orange disks), while the remaining samples correspond to dominated solutions (gray points). These results indicate that the scalarization scheme does not hinder the ability of bSB to discover Pareto-optimal paths for individual weight tuples.

These observations indicate that the scalarization weights have only a limited influence on the capability

of bSB to identify Pareto-optimal solutions. In particular, the feasibility probability and the Pareto hit rate remain nearly constant across the examined weight tuples, while the Pareto coverage remains close to unity. This behavior can be attributed to the structure of the QUBO formulation used to encode the routing problem, where the terms enforcing path feasibility dominate the energy landscape explored by the bSB dynamics. As a result, the algorithm primarily drives the system toward feasible source-to-destination paths, while the scalarized objective only weakly perturbs the relative ordering of these solutions. Based on this observation, the remaining experiments are conducted using a single representative scalarization tuple, since varying the weights does not significantly affect the ability of bSB to recover Pareto-optimal paths.

**Statistical evaluation.**

Following the observations regarding the limited impact of scalarization, the remaining experiments are conducted using a single representative weight tuple $(0.5, 0.5)$. To evaluate the statistical behavior of the algorithm across different graph sizes, we perform a set of experiments on grid graphs of increasing size.

For each grid topology ($5 \times 5$, $7 \times 7$, $10 \times 10$, and $15 \times 15$), we generate $10$ independent problem instances. Each instance is constructed by sampling the edge weights from the distributions described in the previous section. For every problem instance, the bSB algorithm is executed $2000$ times.

To compute the performance metrics, all solutions obtained from the $2000$ runs are aggregated per problem instance. From this aggregated set we estimate the feasibility probability, the Pareto hit probability, and the Pareto coverage as defined previously. The resulting statistics provide an estimate of the expected performance of the algorithm for each grid topology while accounting for variability in the randomly generated edge weights. In addition to the above evaluation, we analyze how the algorithmic performance evolves as the problem size increases. In particular, the metrics are examined as a function of the number of edges in the graph, which provides a convenient measure of the problem complexity. Finally, we investigate the influence of the number of algorithm runs on the discovery of Pareto-optimal solutions. Since bSB is a stochastic algorithm, the set of Pareto solutions recovered by the solver depends on the number of independent executions performed. To study this effect, we conduct an additional experiment in which the number of runs is treated as a free parameter. For each problem instance, the Pareto coverage is evaluated for increasing numbers of runs, allowing us to assess how the probability of discovering the complete Pareto front improves as the sampling effort increases. Each experiment is repeated 50 times, and the average Pareto coverage for each number of runs is reported. The error bars represent the variability across these repetitions. This analysis provides insight into the trade-off between computational effort and solution quality, revealing how many runs are typically required for the algorithm to recover a substantial portion of the Pareto-optimal solutions as the size of the graph grows.

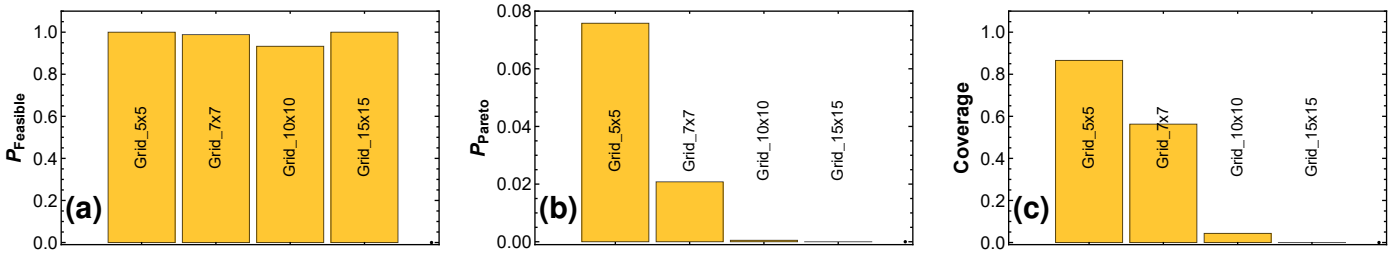



Figure 5.5: Comparison of routing coverage across the four grid topologies.

The statistical results obtained from the grid experiments are illustrated in Figure 5.5 and Figure 5.6. The bar plots in Figure 5.5 summarize the average performance of the algorithm across the $10$ problem instances for each grid topology.

Panel (a) reports the feasibility probability. The results show that bSB consistently produces valid source-to-destination paths across all grid sizes, with feasibility remaining close to unity. This indicates that the algorithm reliably satisfies the routing constraints even as the problem size increases. Building on

this observation, Panel (b) examines the Pareto hit probability, which measures how often the algorithm identifies at least one Pareto-optimal solution. In contrast to the high feasibility levels observed in Panel (a), this metric decreases as the grid size grows. While Pareto-optimal solutions are found relatively frequently in smaller grids, their discovery becomes increasingly rare for larger graphs. This behavior reflects the rapid expansion of the search space and the growing number of feasible routing configurations as the grid network size increases. Finally, Panel (c) complements the previous results by reporting the Pareto coverage, i.e., the fraction of the exact Pareto front recovered by the algorithm. Consistent with the trend observed in Panel (b), the coverage decreases significantly as the graph size increases. For the smallest grid, a large portion of the Pareto set is recovered, whereas for the largest grid the coverage becomes very limited. These results highlight that although feasible solutions are consistently identified, recovering the complete set of Pareto-optimal paths becomes progressively more challenging as the grid network grows.

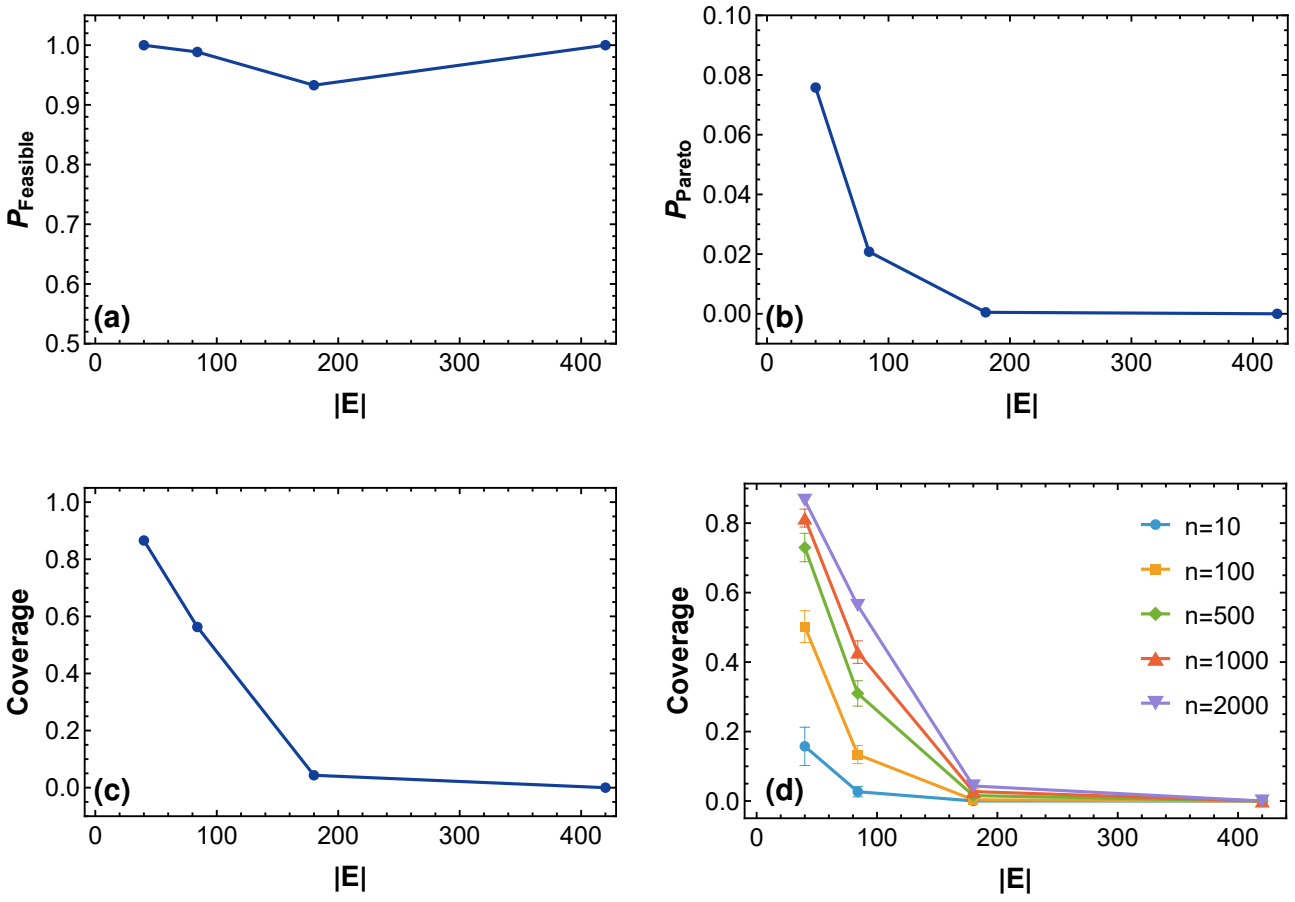



Figure 5.6: Dependence of the evaluation metrics on the number of edges of grid graphs. Panels (a)–(c) present feasibility probability, Pareto hit probability, and Pareto coverage, while panel (d) depicts the dependence of Pareto coverage on the number of algorithm runs.

A complementary view of these results is provided in Figure 5.6, where the metrics are plotted as a function of the number of edges in the graph. The results confirm the trends observed in the bar plots: feasibility remains high even for large graphs, whereas both the Pareto hit probability and the Pareto coverage decrease as the problem size increases.

Finally, panel (d) in Figure 5.6 illustrates the effect of the number of algorithm runs on the Pareto coverage. As expected, increasing the number of runs improves the probability of discovering Pareto-optimal solutions. However, the improvement becomes progressively smaller for larger graphs, indicating that substantially more sampling effort is required to recover a significant portion of the Pareto front as the problem size grows.

**Statistical evaluation on real network topologies.**

To complement the analysis on synthetic networks, we also evaluate the algorithm on real-world communication topologies obtained from the Topology Zoo and RocketFuel datasets. These datasets provide realistic representations of network infrastructures with diverse structural characteristics. By testing the algorithm on these networks, we assess its performance in practical scenarios where connectivity patterns are more realistic and closely reflect those encountered in operational network environments.

A similar statistical analysis is conducted for these real network topologies. In contrast to the grid experiments, where multiple instances were generated by sampling different edge weights, here the variability is introduced through the selection of different source–destination pairs.

For each real topology, $10$ independent problem instances are generated by selecting different source ($S$) and destination ($D$) node pairs. For each problem instance, the bSB algorithm is executed $500$ times in order to collect the solutions required for the statistical analysis. The solutions obtained across all runs are aggregated per topology, and the performance metrics introduced earlier are computed from the aggregated data.

The results are summarized through bar charts reporting the average values of the three metrics for each topology. These plots provide a compact comparison of the algorithmic performance across networks with different sizes and connectivity patterns.

In addition, the behavior of the metrics is examined as a function of the number of edges in the topology, allowing us to analyse how the algorithm scales with increasing network complexity. Similar to the Grid topologies, we also investigate how the number of algorithm runs affects the discovery of Pareto-optimal solutions. For this purpose, the Pareto coverage is evaluated for increasing numbers of runs, up to a maximum of $500$ executions per problem instance. Each experiment is repeated $50$ times, and the reported values correspond to the average performance.

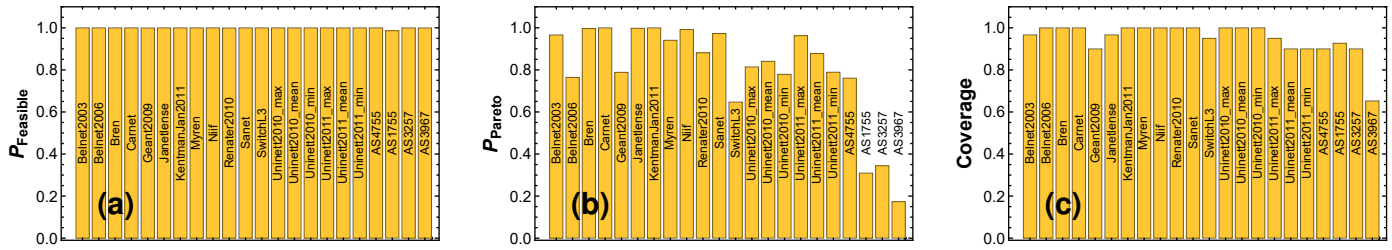



Figure 5.7: Statistical performance of bSB across real network topologies. Panels (a)– (c) report the feasibility probability, Pareto hit probability, and Pareto coverage averaged over ten source–destination instances for each topology.

Figure 5.7 presents the average values of the three evaluation metrics across all real network topologies considered in the dataset. Panel (a) reports the feasibility probability, which remains very close to unity for all networks. This result indicates that the bSB algorithm consistently produces valid source-to-destination paths even in large and heterogeneous network structures.

Panel (b) shows the Pareto hit probability across the examined topologies. In contrast to the feasibility results, this metric exhibits greater variability across networks. While smaller topologies tend to yield high probabilities of discovering Pareto-optimal solutions, the probability decreases for larger networks where the number of feasible routing configurations grows significantly.

Panel (c) reports the Pareto coverage achieved by the algorithm. In most topologies, a large fraction of the Pareto front is recovered, although a moderate decrease in coverage can be observed for some of the larger networks. This behavior reflects the increasing complexity of the search space as the network size grows.

As in the case of the grid graphs, we further analyze the behavior of the evaluation metrics as a function of the number of edges in the topology. To avoid redundancy arising from networks with similar edge counts, only a representative subset of the topologies is included in this analysis. Panel (a) shows that the feasibility probability remains consistently high across all examined networks, indicating that the algorithm continues to produce valid source-to-destination paths even as the number of edges increases. In contrast, panel (b) reveals a gradual decrease in the Pareto hit probability with increasing edge count, suggesting that identifying Pareto-optimal solutions becomes more challenging as the routing search space grows.This trend is also reflected in panel (c), where the Pareto coverage is plotted as a function of the number of edges. Although a decreasing tendency can be observed for larger topologies, the coverage remains relatively high for most networks, indicating that a substantial portion of the Pareto front can still be recovered. Finally, panel (d) examines the influence of the number of algorithm runs on the Pareto coverage. As expected, increasing the number of runs improves the probability of discovering Pareto-optimal solutions. However, the improvement gradually diminishes for larger topologies, indicating that significantly more sampling effort is required to recover a large fraction of the Pareto front as the network complexity increases.

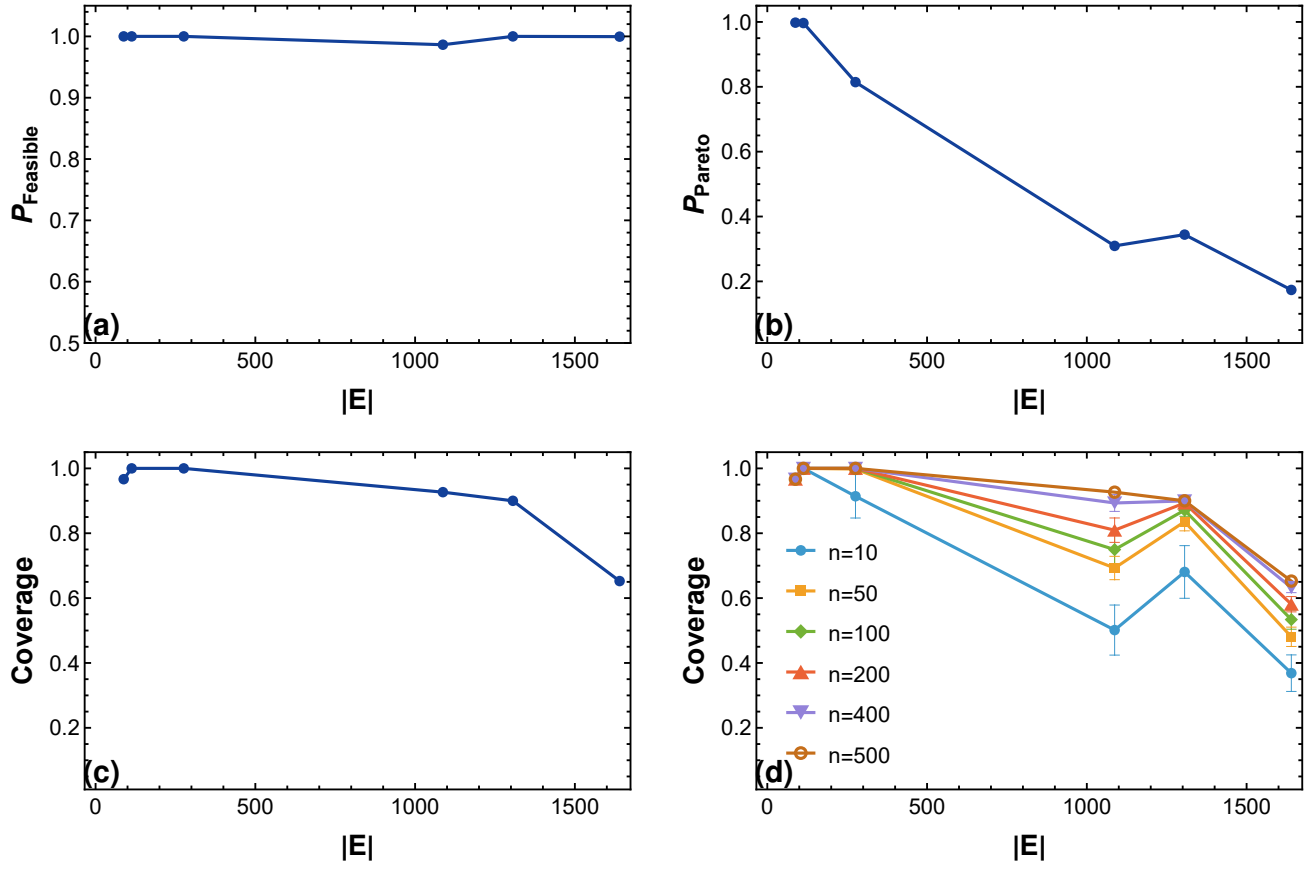



Figure 5.8: Dependence of the evaluation metrics on the number of edges in the topology. Panels (a)–(c) present feasibility probability, Pareto hit probability, and Pareto coverage, while panel (d) depicts the dependence of Pareto coverage on the number of algorithm runs.

A direct comparison between the results obtained on grid graphs and those observed on real network topologies provides additional insight into the behavior of the algorithm. In grid graphs, the size of the Pareto front tends to grow rapidly as the graph becomes larger. The regular structure of the grid creates a large number of alternative routing configurations with comparable objective values, leading to a substantial expansion of the Pareto set. As a result, recovering the complete Pareto front becomes increasingly challenging even for moderate grid sizes, such as those considered in this analysis.

In contrast, the real network topologies examined in this study typically exhibit smaller Pareto fronts. Due to the heterogeneous and more constrained connectivity patterns of these networks, the number

of routing paths that simultaneously satisfy the optimal objective conditions is considerably more limited. Consequently, the search space contains fewer Pareto-optimal solutions, allowing the bSB algorithm to explore the solution landscape more effectively and recover a larger fraction of the Pareto front even for significantly larger graphs.

However, for the largest graphs listed in Table 5.2, which contain substantially more edges than those considered in the previous analysis, the bSB algorithm was not always able to identify feasible solutions. As the graph size increases, the solution landscape becomes considerably more complex, making it increasingly difficult for the algorithm to effectively explore the space of routing configurations. This observation is consistent with the results presented earlier, where larger graphs required a significantly higher number of sampling iterations in order to recover Pareto-optimal solutions.

These findings indicate that although bSB demonstrates promising capabilities for solving certain classes of multi-objective routing problems, and compares favorably with several quantum-inspired and quantum approaches, its behavior is still not fully understood. In particular, the selection of algorithmic parameters and their interaction with the structural properties of the underlying graph—such as size, connectivity, and the resulting complexity of the Pareto front—remain open research questions. Moreover, due to the stochastic nature of the algorithm, a single execution does not guarantee that the returned solution belongs to the Pareto front. Consequently, a large number of independent runs may be required even to identify a single Pareto-optimal solution, and many more to recover a substantial portion of the Pareto front, as observed in the experiments presented above. While the algorithm can in principle benefit from parallel execution, this requirement reduces the practical advantage of parallelization, since multiple runs must be aggregated in order to obtain reliable Pareto-optimal solutions. These aspects highlight the need for further systematic investigation by the scientific community in order to better understand the scalability and parameterization of the method.

At its current stage of development, the method should therefore be regarded primarily as an exploratory optimization approach rather than a technique suitable for real-time routing applications. For the purposes of the CASTOR project, where the objective is to reliably compute the complete set of Pareto-optimal paths, exact approaches such as the multi-objective Dijkstra algorithm remain more appropriate, as they guarantee the recovery of the full Pareto front.

# Chapter 6

# Summary and Conclusions

This deliverable introduces the first version of the three core elements of the Trust Engineering process within the CASTOR Framework, namely the Trust Assessment Framework (TAF), the Risk Assessment Engine (RAE), and the Optimization Engine (OE). Together, these three pillars allow CASTOR to continuously collect and interpret trustworthiness evidence, quantify and propagate trust (and risk) across complex topologies, and compute network paths that are jointly optimal with respect to both network and trust metrics.

The first version of the **Trust Assessment Framework (TAF)** has been specified and implemented within this deliverable. It has been implemented as a federated system of trust assessment agents. These agents jointly provide topology-wide trust characterisation at the orchestration layer. The evaluation plan detailed in Chapter 3 validates core TAF behaviours related to trust propagation in federated, multi-agent systems, including the temporal evolution of these assessments. Preliminary evaluations demonstrate that the necessity of using the appropriate Subjective Logic operators when discounting or consolidating trust opinions in order to ensure an accurate trust characterization.

The first version of the **Risk Assessment Engine (RAE)** has been specified and implemented within this deliverable. It establishes the foundational framework for computing context-specific Required Trust Levels (RTLs) by shifting from static, device-centric vulnerability scoring to dynamic, topology-aware risk profiling. Crucially, Chapter 4 delineates the core parameters that dictate the stochastic propagation of an adversary from a compromised node to a neighbouring asset within the network topology. Building upon these parameters, the goal is to elevate the modeling of cascading attack analysis into a formal Markov Decision Process (MDP). This mathematical framing enables the systematic processing of both the transition probabilities inherent in multi-step threats and the degree of confidence regarding the resulting state after each attack step. Furthermore, through the introduction of Monte Carlo simulation capabilities (an initial version of which is presented in this deliverable) we aim to experiment with simulating probabilistic ”what-if” scenarios that will unlock the dynamic updating of the transition probabilities used to derive the topology-aware risk and, eventually, the RTL constraints. The case study evaluation presented in this chapter validates this methodology, demonstrating that the incorporation of cascading attacks can significantly impact the accuracy of the RTL requirements and, therefore, the network’s true security posture. Ultimately, this approach seamlessly bridges risk and trust within the CASTOR framework by identifying the exact security controls and runtime evidence required to accurately monitor the Actual Trust Level (ATL).

The **Optimization Engine (OE)** formulates the discovery of trusted communication paths as a multi-objective combinatorial optimization task over the network topology. As a starting point, the OE tackles the reduced bi-objective optimization problem formulated in D4.1 [6] to simultaneously accommodate network and trust objectives. In the first prototype, two distinct approaches are introduced to solve this initial routing problem. First, to mirror standard routing convergence processes, the engine employs an exact bi-objective variant of Dijkstra’s algorithm to establish a baseline Pareto front. Second, it implements

a heuristic approach based on the quantum-inspired Simulated Bifurcation (bSB) algorithm. Based on this first version of the OE, a preliminary scalability analysis using realistic topologies demonstrates how network size (i.e., specifically the number of nodes and edges) affects both path lengths and Pareto front sizes. Ultimately, these insights will inform the parametrization and scaling of the OE for larger deployment scenarios. In the next version, this foundational work will be expanded to evaluate more complex multi-objective instances of the problem.

The three pillars developed in this deliverable are designed to operate as a coherent pipeline for trusted path establishment within CASTOR. This integrated design ensures that trust decisions in CASTOR are not made in isolation. Instead, they are derived from a combination of evidence-centric trust assessment (from the TAF), topology-aware risk reasoning (from the RAE), and multi-objective path optimisation (from the OE). D4.2 represents a foundational milestone towards realising CASTOR's vision of enabling trust-aware traffic engineering provisioning. The current implementations of TAF, RAE, and OE described in this deliverable provide the necessary building blocks towards the evaluation of the first version of the CASTOR integrated framework, both in the context of the Proof-of-Concept (PoC) scenarios but most importantly in the context Use Case applications as detailed in D6.1 [9]. This validation will employ the three pillars jointly to assess their effectiveness in real-world deployments.

In conclusion, deliverable D4.2 provides the first description of TAF, RAE, and OE components within the CASTOR framework. The theoretical description is corroborated by experimental insights on the CASTOR's capabilities to trust-aware traffic engineering policies.

# List of Abbreviations

| Abbreviation | Translation |
|---|---|
| **API** | Application Programming Interface |
| **ATL** | Actual Trust(worthiness) Level |
| **BC** | Betweenness Centrality |
| **BCF** | Belief Constraint Fusion |
| **bSB** | Ballistic Simulated Bifurcation |
| **BW** | Bandwidth |
| **CIA** | Confidentiality, Integrity, Availability |
| **CPE** | Common Platform Enumeration |
| **CVE** | Common Vulnerabilities and Exposures |
| **CVSS** | Common Vulnerability Scoring System |
| **CWE** | Common Weakness Enumeration |
| **DLT** | Distributed Ledger Technology |
| **DoS** | Denial of Service |
| **dSB** | Discrete Simulated Bifurcation |
| **DSPG** | Directed Series-Parallel Graph |
| **EPSS** | Exploit Prediction Scoring System |
| **FIB** | Forwarding Information Base |
| **FSM** | Finite State Machine |
| **HTTP** | Hypertext Transfer Protocol |
| **IRL** | Individual Risk Level |
| **ISP** | Internet Service Provider |
| **MDP** | Markov Decision Process |
| **NDQ** | Node Discovery Query |
| **NIST** | National Institute of Standards and Technology |
| **NVD** | National Vulnerability Database |
| **OE** | Optimization Engine |
| **OIP** | Opinion Information Point |
| **PC** | Pareto Coverage |
| **PCEP** | Path Computation Element Protocol |
| **PoC** | Proof-of-Concept |
| **PPS** | Parallel-Path Subnetwork |

| | |
|---|---|
| **QUBO** | Quadratic Unconstrained Binary Optimization |
| **RA** | Risk Assessment |
| **RAE** | Risk Assessment Engine |
| **RCE** | Remote Code Execution |
| **RPKI** | Resource Public Key Infrastructure |
| **RTL** | Required Trust(worthiness) Level |
| **SB** | Simulated Bifurcation |
| **SL** | Subjective Logic |
| **SSH** | Secure Shell |
| **STN** | Subjective Trust Network |
| **TAF** | Trust Assessment Framework |
| **TAM** | Trust Assessment Manager |
| **TAQ** | Trust Assessment Query |
| **TAR** | Trustworthiness Assessment Request |
| **TAS** | Trust Assessment Service |
| **TD** | Trust Decision |
| **TDE** | Trust Decision Engine |
| **TE** | Traffic Engineering |
| **TLS** | Transport Layer Security |
| **TMI** | Trust Model Instance |
| **TMT** | Trust Model Template |
| **TMT-DB** | Trust Model Template Database |
| **TMT-ID** | Trust Model Template Identifier |
| **TNDE** | Trust Network Device Extensions |
| **TNDI** | Trust Network Device Interface |
| **TNDI-SP** | TNDI Security Protocol |
| **TN-DSM** | Trust Network Device Security Monitor |
| **TPL** | Trust Policy Language |
| **TSM** | Trust Sources Manager |
| **UC** | Use Case |
| **vRouter** | Virtualized Router |

# Bibliography

[1] 5G Automotive Association (5GAA). A framework for dynamic trustworthiness assessment in cooperative and automated vehicles. White paper, 5G Automotive Association (5GAA), 2019.

[2] Apache Software Foundation. Apache kafka. https://kafka.apache.org/, 2026. Accessed: 2026-03-24.

[3] 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.

[4] Harold Booth, David Rike, and Gregory Witte. The national vulnerability database (nvd): Overview. Technical report, National Institute of Standards and Technology, Gaithersburg, MD, 2013. Accessed December 15, 2025.

[5] CAIDA. The CAIDA UCSD Macroscopic Internet Topology Data Kit-[ITDK 2011-10]. http://www.caida.org/data/active/internet-topology-data-kit/. Accessed: February, 2026.

[6] CASTOR. Architectural specification of castor continuum-wide trust assessment framework. Deliverable 4.1, The CASTOR Consortium, 12 2025.

[7] CASTOR. Architectural specification of dynamic enforcement of trust-/network-aware path establishments. Deliverable 5.1, The CASTOR Consortium, 12 2025.

[8] CASTOR. Operational landscape, requirements and reference architecture - initial version. Deliverable 2.1, The CASTOR Consortium, 12 2025.

[9] CASTOR. Castor integrated framework, use case analysis & interim report on poc verification results. Deliverable 6.1, The CASTOR Consortium, 3 2026.

[10] CASTOR. Device-level & continuum-wide trust extensions and security controls (first release). Deliverable 3.2, The CASTOR Consortium, 03 2026.

[11] CASTOR. Implementation of castor policy enforcement in (cross-) domain continuum topologies. Deliverable 5.2, The CASTOR Consortium, 6 2026.

[12] Forum of Incident Response and Security Teams (FIRST). FIRST - Forum of Incident Response and Security Teams. https://www.first.org/. Accessed: March 23, 2026.

[13] Hayato Goto, Kotaro Endo, Masaru Suzuki, Yoshisato Sakai, Taro Kanao, Yohei Hamakawa, Ryo Hidaka, Masaya Yamasaki, and Kosuke Tatsumura. High-performance combinatorial optimization based on classical mechanics. *Science Advances*, 7(6):eabe7953, 2021.

[14] Hayato Goto, Kosuke Tatsumura, and Alexander R Dixon. Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems. *Science advances*, 5(4):eaav2372, 2019.

[15] Audun Jøsang. *Subjective logic*. Springer, 2016.

[16] Audun Jøsang, Stephen Marsh, and Simon Pope. Exploring different types of trust propagation. In *International Conference on Trust Management*, pages 179–192. Springer, 2006.

[17] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[18] MITRE Corporation. Common Attack Pattern Enumeration and Classification (CAPEC). https://capec.mitre.org/. Accessed: March 23, 2026.

[19] MITRE Corporation. Common Weakness Enumeration (CWE). https://cwe.mitre.org/. Accessed: March 23, 2026.

[20] National Institute of Standards and Technology. Common Platform Enumeration (CPE). https://nvd.nist.gov/products/cpe. Accessed: March 23, 2026.

[21] National Institute of Standards and Technology (NIST). NIST Special Publication 800-30 Rev. 1: Guide for Conducting Risk Assessments. Special Publication 800-30, NIST, 2012. Accessed: March 23, 2026.

[22] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, 2004.

[23] The CONNECT Consortium. CONNECT Trust & Risk Assessment and CAD Twinning Framework (InitialVersion) , 2024. Accessed: 2025-11-16.

[24] The Forum of Incident Response and Security Teams (FIRST). Common Vulnerability Scoring System (CVSS) Version 3.1. https://www.first.org/cvss/v3.1/specification-document, 2019. Accessed: March 23, 2026.