



SAFE-6G

A Smart and Adaptive Framework for Enhancing Trust in 6G Networks

Deliverable D4.4: User-centric Security Function

Date: 06/10/2026

Version: V1.0

DISCLAIMER

This document contains confidential information reserved for the partners of the SAFE-6G (“A Smart and Adaptive Framework for Enhancing Trust in 6G Networks”) Consortium and is subject to the confidentiality obligations set out in the Grant Agreement 101139031 and to the SAFE-6G Consortium Agreement.

This document contains information, which is proprietary to the SAFE-6G Consortium, that is subject to the rights and obligations and the terms and conditions applicable to the Grant Agreement. The action of the SAFE-6G Consortium is funded by the European Commission.

Neither this document nor the information contained herein shall be used, copied, duplicated, reproduced, modified, or communicated by any means to any third party, in whole or in parts, except with prior written consent of the SAFE-6G Consortium. In such case, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced. In the event of infringement, the consortium reserves the right to take any legal action it deems appropriate.

This document reflects only the authors’ view and does not necessarily reflect the view of the European Commission. Neither the SAFE-6G Consortium as a whole, nor a certain party of the SAFE-6G Consortium warrant that the information contained in this document is suitable for use, nor that the use of the information is accurate or free from risk and accepts no liability for loss or damage suffered by any person using this information.

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Grant Agreement	101139031
Document number	D4.4
Document title	User-centric Security Function
Lead Beneficiary	IQB
Editor(s)	Theodoros Ioannidis (IQB) Apostolos Garos (SHG) Guillaume Hébert (KEY)
Author(s)	Theodoros Ioannidis (IQB) Kushal Mehta (IQB) Orestis Panos (IQB) Athanasia Sampazioti (IQB) Konstantinos Zimpounis (IQB) Guillaume Hébert (KEY) Henry Faures Geors (KEY) Van Hoan Hoang (KEY)
Dissemination level	Public
Contractual date of delivery	30/09/2026
Status	Final
File name	SAFE-6G_D4.4_V1.0.pdf

Revision History

Version	
V0.1	Initial ToC
V0.2	First Round of Contributions (KEY)
V0.3	Second Round of Contributions (IQB, KEY)
V0.4	Internal Review & Edits (SHG, IQB)
V0.5	Chapter 7 Update & Edits (IQB)
V1.0	Final Version

GLOSSARY

Abbreviations/Acronym	Description
3GPP	3rd Generation Partnership Project
5G	Fifth Generation Mobile Network
6G	Sixth Generation Mobile Network
ABAC	Attribute-Based Access Control
AI	Artificial Intelligence
API	Application Programming Interface
ARF	Architecture and Reference Framework
CAPIF	Common API Framework
cLoTw	Calibrated Level of Trustworthiness
CoCo	Cognitive Coordinator
CPU	Central Processing Unit
DataOps	Data Operations
DCQL	Digital Credentials Query Language
DID	Decentralised Identifier
DLT	Distributed Ledger Technology
EC	European Commission
eIDAS	Electronic Identification, Authentication and Trust Services
EUDI	European Digital Identity
FN-DSA	FFT over NTRU-Lattice-Based Digital Signature Algorithm
GA	Grant Agreement
gRPC	gRPC Remote Procedure Call
IAM	Identity and Access Management
JSON	JavaScript Object Notation
JWT	JSON Web Token
KEM	Key Encapsulation Mechanism
KMS	Key Management System
KPI	Key Performance Indicator
KVI	Key Value Indicator
LoTw	Level of Trustworthiness
ML	Machine Learning
ML-DSA	Module-Lattice-Based Digital Signature Algorithm
ML-KEM	Module-Lattice-Based Key-Encapsulation Mechanism
mTLS	Mutual Transport Layer Security
nApp	Network Application
NIST	National Institute of Standards and Technology
nLoTw	Non-calibrated Level of Trustworthiness
OID4VCI	OpenID for Verifiable Credential Issuance
OID4VP	OpenID for Verifiable Presentations
OIDC	OpenID Connect
OpenCAPIF	Open-source Common API Framework

PEP	Policy Enforcement Point
PQC	Post-Quantum Cryptography
PromQL	Prometheus Query Language
RBAC	Role-Based Access Control
REST	Representational State Transfer
SAFE-6G	A Smart and Adaptive Framework for Enhancing Trust in 6G Networks
SAML	Security Assertion Markup Language
SIOPv2	Self-Issued OpenID Provider v2
SLH-DSA	Stateless Hash-Based Digital Signature Algorithm
SSI	Self-Sovereign Identity
TF	Trust Function
TOSCA	Topology and Orchestration Specification for Cloud Applications
TPS	Transactions Per Second
vApp	Vertical Application
VC	Verifiable Credential
VCDM	Verifiable Credentials Data Model
VDR	Verifiable Data Registry
VP	Verifiable Presentation
WP	Work Package

EXECUTIVE SUMMARY

This deliverable describes the User-centric Security Trust Function designed and implemented in the context of SAFE-6G Work Package 4 (Task 4.3). The Security Trust Function contributes to the broader SAFE-6G trustworthiness management framework delivering an adaptive, security control plane that addresses identity sovereignty, continuous authentication, fine-grained authorisation, zero-trust policy enforcement, and immutable audit trails in dynamic 6G and metaverse environments.

Following the WP4 layered reference model, the Local AI Agent serves as the security intelligence component, continuously monitoring cluster telemetry and executing a multi-stage machine learning inference pipeline to assess edge infrastructure viability. In coordination with the Cognitive Coordinator, it dynamically recommends an operational security trust flavour (low, mid, high), balancing requested security guarantees against real-time continuum operational capacity.

The Vertical Applications layer provides a decentralised identity service layer aligned with the emerging European Digital Identity Wallet ecosystem. It supports all three core Self-Sovereign Identity roles within the framework, including a Credential Issuer preparing and signing attribute attestations via OpenID for Verifiable Credential Issuance (OID4VCI), a Credential Verifier validating presentations via OpenID for Verifiable Presentations (OID4VP) and SIOPv2 across both OpenID Connect and Webhook integration modes, and a user-centric SSI Vault functioning as a secure cloud-wallet with post-quantum cryptographic capabilities. The Network Applications layer defines a dedicated Policy Enforcement Point (PEP), which enforces zero-trust access controls, tokenises operational actions and authorises secure ledger and smart contract access.

The Security Trust Function features an infrastructure layer utilising Hyperledger Fabric for its distributed ledger and specialised smart contracts (Verifiable Data Registry, Security Audits) for native did:safe6g identifier method and tamper-evident security audit logs.

The work described in this document builds on previous SAFE-6G outputs, in particular D4.1, which introduced the user-centric Trust Functions and the initial architecture of the Security Function, and on technical contributions related to SSI integration, use case metaverse authentication and authorisation, DID management, and credential lifecycle management, establishing the architectural and implementation baseline for empirical benchmarking and KPI validations conducted in Work Package 5.

KEYWORDS

SAFE-6G, Security Function, Self-Sovereign Identity, Verifiable Credentials, Decentralised Identifiers, OID4VCI, OID4VP, SIOPv2, OIDC, Webhook, Blockchain, Smart contract, Hyperledger Fabric, User-centric Trust, Credential Lifecycle, Access Control, Post-Quantum Cryptography, Zero-Trust Architecture

TABLE OF CONTENTS

1	Introduction	1
2	Related Work	3
3	Trust Function Definition & Role in SAFE-6G.....	5
3.1	Requirements Mapping	5
3.2	Conceptual Definition & Role	6
3.3	Trust Signals & Adaptive Deployments	6
3.4	Level Of Trustworthiness (LoTw)	7
3.5	Operational Trust Flavours.....	8
3.6	Role of Self-Sovereign Identity (SSI)	8
4	Reference Architecture	10
5	Trust Function Layers.....	12
5.1	Local AI Agent.....	12
5.1.1	Event-Driven Runtime & Messaging via Kafka	13
5.1.2	System Telemetry Ingestion Via Thanos	13
5.1.3	Machine Learning Inference Pipeline via KServe	13
5.1.4	Meta-Orchestration Interactions via aerOS	14
5.2	Vertical Applications (vApps)	14
5.2.1	SSI Roles and Decentralised Identifiers.....	15
5.2.2	Credential Issuance & Verification	16
5.2.3	Credential Lifecycle Management.....	19
5.2.4	Metaverse Credential Data Model.....	20
5.2.5	Post-Quantum Cryptographic Operations	21
5.3	Network Applications (nApps)	22
5.3.1	Zero-Trust Policy Enforcement Point (PEP).....	23
5.3.2	Dynamic Flavour Enforcement & Ledger Auditing.....	24
6	Trust Evaluation & Enforcement Logic	25
7	Future Research Pathways	27
7.1	Advanced Adaptive Trust Orchestration	28
7.2	Evolution of Crypto-Agile and Post-Quantum Trust.....	28
7.3	Extended Identity and Trust Interoperability	28
7.4	Trustworthy Security at Larger Scale and in Diverse Environments	29
7.5	Reuse and Evolution of SAFE-6G Technology Assets.....	29

8	Conclusions	30
9	REFERENCES	31
Annex 1:	Technical Specifications	33
A1.1	The did:safe6g DID Method	33
A1.2	Ledger Audit Model	34
A1.3	Credential Format Examples	35

List of FIGURES

Figure 1:	Security Function Architectural Components	11
Figure 2:	Local-AI Agent Operational Lifecycle	12
Figure 3:	Vertical Applications (vApps) SSI Architecture	15
Figure 4:	OID4VCI Credential Issuance Sequence	17
Figure 5:	OID4VP Credential Verification via OIDC Flow	18
Figure 6:	Credential Verification via Webhook Flow	19
Figure 7:	VDR-Invoker Operational Architecture & Transaction Lifecycle	23

List of TABLES

Table 1:	User-Centric Security Function Requirements	6
Table 2:	Available Cryptographic Algorithms	22
Table 3:	Security Trust Function Component Maturity	27
Table 4:	did:safe6g Method Operations	33
Table 5:	did:safe6g DID Document Example (High Flavour)	34
Table 6:	SecurityAudits Record Structure	34
Table 7:	Flavour-Dependent Audit Sequence	35
Table 8:	SecurityAudits Record Example	35
Table 9:	Bitstring Status List Credential JSON Example	36
Table 10:	MetaverseCredential JSON Example	36

1 INTRODUCTION

The SAFE-6G project aims to define and validate a smart and adaptive framework for enhancing trust across sixth generation (6G) mobile networks. In this context, Work Package 4 (WP4) focuses on cognitive coordination mechanisms, Local AI Agents, and user-centric Trust Functions capable of translating high-level trust requirements into adaptive, trust-aware behaviours across the computing continuum. Deliverable D4.4 reports specifically on the design and implementation of the User-centric Security Trust Function (Task 4.3), whose purpose is to provide an adaptive, closed-loop security control plane delivering sovereign identity management, continuous authentication, fine-grained authorisation, zero-trust policy enforcement, and immutable auditability.

At the core of the SAFE-6G vision is a shift toward user-centric trustworthiness. While 4G and 5G networks rely heavily on operator-controlled trust anchors, such as SIM cards and centralised subscriber databases, SAFE-6G places control back in the hands of users and enterprise tenants. In this model, users maintain sovereignty over their cryptographic keys and verifiable credentials through a secure personal vault, revealing only the attributes necessary for a given interaction. By expressing their required Level of Trustworthiness (LoTw), users directly shape the system's security posture. The Security Trust Function bridges this user intent with the underlying network, dynamically mapping requirements into operational security trust flavours (low, mid, high) while accounting for real-time infrastructure constraints.

To support the overall objectives of WP4, the Security Trust Function adheres to the common three-tier reference model, distributing intelligence and enforcement capabilities across three cooperating layers:

- The **Local AI Agent** serves as the security intelligence component, continuously ingesting cluster telemetry (CPU, memory, request rates) and executing a multi-stage machine learning inference pipeline to assess edge infrastructure viability and recommend operational flavours in coordination with the Cognitive Coordinator (CoCo).
- The **Vertical Applications** (vApps) layer realises a decentralised identity service layer aligned with the European Digital Identity Wallet ecosystem, supporting Credential Issuance (OID4VCI), Presentation Verification (OID4VP/SIOPv2 across OpenID Connect and Webhook modes), and an SSI Vault (cloud-wallet) with post-quantum cryptographic agility.
- The **Network Applications** (nApps) layer deploys a dedicated Policy Enforcement Point (VDR-Invoker) to enforce zero-trust policies, tokenise operational actions, and mediate interactions with external service meshes and OpenCAPIF APIs.

Furthermore, trust and auditability are anchored onto a Hyperledger Fabric distributed ledger through specialised smart contracts, including an immutable Verifiable Data Registry supporting the native did:safe6g identifier method and an auditable Security Audits log recording all critical authorisation and state transitions.

The scope of this deliverable covers the complete architectural design, software implementations, and trust evaluation protocols developed under Task 4.3. To maintain clear boundaries within the project,

several aspects are intentionally designated as out of scope: companion trust functions (Safety, Privacy, Resilience, and Reliability) are specified in sibling deliverables (D4.2, D4.3, D4.5, and D4.6); physical layer radio access encryption and user plane packet processing are assumed to be provided by underlying 3GPP network infrastructure; and empirical performance benchmarking, live stress testing, and formal KPI validation are deferred to Work Package 5 (WP5) activities.

2 RELATED WORK

Security and trust management in 6G environments require mechanisms that go beyond static identity, perimeter-based security, and centralised authentication. D4.1 [1] highlights that 6G systems are expected to operate across heterogeneous infrastructures, edge-cloud environments, distributed services, and metaverse applications, making continuous trust evaluation, contextual enforcement, and transparency necessary design properties.

Traditional identity federation mechanisms such as OpenID Connect [12] and SAML provide mature building blocks for authentication and authorisation, but they remain centralised and depend on fixed identity providers and static trust relationships. In the context of SAFE-6G, these models show limitations when applied to dynamic environments involving virtualised entities, distributed services, digital twins, avatars, industrial operators, and metaverse users.

Zero-Trust Architecture, as defined in NIST SP 800-207 [15], requires that no entity be implicitly trusted regardless of its network location. Every access request must be authenticated, authorised, and validated before granting access to resources. SAFE-6G adopts this principle at the network enforcement tier, where a dedicated Policy Enforcement Point intercepts and validates all transactions before they reach the trust anchoring infrastructure. NIST SP 800-207 does not, however, prescribe how zero-trust controls should interact with decentralised identity systems or how enforcement rigour should adapt to runtime infrastructure conditions. This is the gap addressed by the Security Trust Function's flavour-based enforcement model.

Self-Sovereign Identity (SSI) takes a different approach: identities, attributes, and permissions are represented through Decentralised Identifiers (DIDs) [6], Verifiable Credentials [7], and Verifiable Presentations. This supports cryptographic verification, selective disclosure, portability of attributes, and user-controlled identity data. Within SAFE-6G, SSI is investigated as a mechanism to support strong authentication, contextual authorisation, and privacy-preserving access control for metaverse-oriented use cases.

The regulatory direction for SSI in Europe is set by the revised eIDAS Regulation (eIDAS 2.0) [16] and the associated Architecture Reference Framework (ARF) [17]. eIDAS 2.0 mandates that EU Member States offer citizens a European Digital Identity (EUDI) Wallet capable of storing and presenting electronic attestations of attributes. The ARF specifies the protocols and components for wallet-to-verifier interactions, credential issuance, and trust list management. The SAFE-6G SSI layer follows this direction by implementing wallet-compatible issuance and presentation flows, although the current implementation targets a project-specific private trust anchor rather than a public EUDI trust infrastructure.

The vApp layer relies on emerging OpenID protocols, including OpenID for Verifiable Credential Issuance (OID4VCI) [9] and OpenID for Verifiable Presentations (OID4VP) [10] with SIOPv2 [11]. These protocols allow credentials to be issued to a wallet, stored under the control of the holder, and later presented to a verifier during an authentication or authorisation flow. However, these protocols are still evolving, and their integration into existing systems requires careful alignment between wallet

behaviour, verifier expectations, credential formats, presentation definitions, DID resolution, and claim mapping.

A specific challenge in SAFE-6G is the use of a private Hyperledger Fabric [13] deployment as a trust anchor. Standard DID methods such as did:web or did:key do not directly address the requirements of a project-specific private blockchain, particularly when the registry is exposed through a custom gateway and chaincode. For this reason, a dedicated did:safe6g method was specified and integrated to support DID creation, resolution, update, and deactivation in the SAFE-6G environment.

A further important aspect is credential lifecycle management. Verifiable Credentials may need to be revoked, suspended, or re-issued depending on changes in role, authorisation status, compromise, expiration, or policy evolution. The Credential Lifecycle Manager addresses this by implementing the W3C Bitstring Status List specification [8] and the IETF Token Status List [14], while maintaining compatibility with both internal and external lifecycle registries.

High-assurance deployments must also account for cryptographic longevity. The NIST Post-Quantum Cryptography standardisation effort [18] has standardised ML-KEM (Kyber) for key encapsulation and ML-DSA (Dilithium) and SLH-DSA (SPHINCS+) for digital signatures, and has selected Falcon for standardisation as FN-DSA (FIPS 206, in development). As quantum computing matures, classical signature algorithms used in current DID and Verifiable Credential implementations will become vulnerable. The Security Trust Function addresses this by incorporating post-quantum primitives in its high trust flavour, so the system can adopt quantum-resistant operations without redesigning the identity architecture.

The gaps identified above, namely the rigidity of centralised identity models, the absence of adaptive zero-trust enforcement tied to infrastructure conditions, the immaturity of wallet-based OpenID flows, the missing DID method for private blockchain anchoring, the need for flexible credential lifecycle controls, and the future risk posed by quantum-capable adversaries, directly motivated the architectural decisions described in the remaining chapters of this deliverable.

3 TRUST FUNCTION DEFINITION & ROLE IN SAFE-6G

The SAFE-6G architecture defines five Trust Functions, each responsible for a specific domain: Safety, Security, Privacy, Resilience, and Reliability. The Cognitive Coordinator (CoCo) acts as the central cognitive authority, translating user intent into trust policies and coordinating the activation of these functions. The Security Trust Function confines its scope to identity management, cryptographic assurance, access gating, and auditability. It does not monitor physical safety hazards, calculate radio-link retransmission reliability, or manage multi-tenant privacy budgeting, which are handled by the corresponding sibling functions. The Security Trust Function receives trust signals and user intent from CoCo, assesses telemetry and executes machine learning inference to determine whether the requested security posture is feasible, and enforces the resulting decisions across identity gateways, policy enforcement points, and distributed ledgers.

3.1 REQUIREMENTS MAPPING

Deliverable D4.4 sits within the core technical components of SAFE-6G Work Package 4, specifying and implementing the User-Centric Security Trust Function. Backwards, the architecture and implementation in this deliverable satisfy the security requirements defined in deliverable D2.1 [2]. The following table maps each requirement to its realising D4.4 component and the relevant deliverable sections.

Requirement ID	Short Title & Description	Realisation in D4.4 Component	Section
REQ-FSEC-NET-F-M-01	6G Component Interaction	OpenCAPIF-aligned API exposure & invoker interfaces for Core/MetaOS interaction.	4, 5.3.2
REQ-FSEC-INT-F-M-02	Self-Sovereign Identity (SSI)	Decentralised identity layer with W3C VCs, OID4VCI, and OID4VP.	3.6, 5.2.1
REQ-FSEC-COMP-NF-M-03	Smart Contract Security	VerifiableDataRegistry & SecurityAudits Hyperledger Fabric chaincodes.	5.3.2
REQ-FSEC-COMP-NF-M-04	Comprehensive Audit Logging	Immutable audit logging on DLT via the SecurityAudits chaincode (mid and high flavours).	5.3.2, 6, A1.2
REQ-FSEC-INT-F-M-05	Decentralised Identifiers (DIDs)	Creation, management, and resolution of did:safe6g identifiers.	5.2.1
REQ-FSEC-COMP-F-M-06	Secure Audit Storage	Hyperledger Fabric distributed ledger for immutable audit trail recording.	5.3.2, A1.2
REQ-FSEC-INT-F-M-07	Tokenisation through VCs	Action tokenisation via bearer tokens mapped from verified VCs.	5.2.2, 5.3.1
REQ-FSEC-COMP-NF-M-08	Data Encryption	Encrypted private-key storage in the SSI Vault/KMS & mTLS-secured gRPC channels to the ledger.	4, 5.2.2
REQ-FSEC-COMP-NF-M-09	Identity Access Control	Zero-Trust PEP (VDR-Invoker) with bearer-token authentication, plus role- and attribute-based authorisation from verified VC claims.	5.2.4, 5.3.1, 6

REQ-FSEC-PER-F-M-10	Real-Time Security Processing	Real-time telemetry ingestion via Thanos & KServe ML inference pipeline.	5.1.2, 5.1.3
REQ-FSEC-PER-NF-M-11	Scalable Security Handling	Bitstring Status List v1.0 credential status checks & stateless, horizontally scalable PEP.	5.2.3, 5.3.1
REQ-FSEC-PER-F-R-12	Customisable Security Policies	Operational security trust flavours (low, mid, high) mapped from cLoTw.	3.5, 5.3.2
REQ-FSEC-PER-NF-M-13	High Reliability	Stateless PEP microservices with Kubernetes multi-replica pod failover.	5.3.1
REQ-FSEC-PER-NF-M-14	Low-Latency Performance	Cached credential status & fast-path execution under the low flavour.	5.2.3, 5.3.1
REQ-FSEC-COMP-NF-M-15	Highest Security Standards	Alignment with W3C VC/DID, OpenID (OIDC, OID4VCI, OID4VP), NIST SP 800-207, and NIST PQC standards.	2, 5.2.1, 5.2.5
REQ-FSEC-INT-NF-R-16	Legacy Interoperability	OIDC/OAuth2 enterprise bridge for legacy 4G/5G system integration.	5.2.2, 5.3.1

Table 1: User-Centric Security Function Requirements

3.2 CONCEPTUAL DEFINITION & ROLE

Within the SAFE-6G architecture, security and trust are related but distinct concepts. Security covers the cryptographic, algorithmic, and operational mechanisms deployed to protect network assets, communication channels, and transmitted data from unauthorised access or malicious interference. Trust represents a measurable degree of confidence that an entity, service, or underlying edge-cloud infrastructure will reliably satisfy defined operational, privacy, and security expectations under changing conditions.

The User-centric Security Trust Function, developed under Task 4.3, is the autonomous control-plane component responsible for evaluating and enforcing security-related trust decisions within SAFE-6G. Its role is to enable secure, privacy-preserving access to 6G network services and vertical applications, including industrial and educational metaverse environments, by coupling user-declared security requirements with infrastructure telemetry, verifiable identity credentials, and zero-trust policy enforcement. Rather than imposing a fixed security perimeter that ignores computational constraints, the Security Trust Function maintains an adaptive, closed-loop security posture. The function continuously balances user-demanded protection against the current capacity of edge hosting nodes, avoiding security under-provisioning on one hand and resource exhaustion on the other.

3.3 TRUST SIGNALS & ADAPTIVE DEPLOYMENTS

The Security Trust Function processes several categories of trust signals to derive and execute security decisions. These signals arrive from user-facing channels and from internal infrastructure monitoring.

At the user and tenant tier, the function receives declared security requirements, normalised into intent scores, alongside identity attributes contained in Verifiable Credentials. These credentials carry claims such as organisational affiliations, access roles, security clearances, and operational certifications. The function also evaluates cryptographic and resolution signals, including digital

signature validity, proof-of-possession verification, and the real-time resolution status of Decentralised Identifiers (DIDs) across distributed ledger anchors.

The function also inspects credential lifecycle signals on an ongoing basis, querying cryptographic status lists to verify that presented credentials have not been revoked or suspended by their issuing authority. At the infrastructure tier, the function samples real-time telemetry from edge-cloud hosting nodes: container CPU usage, memory consumption, packet drop rates, and load spikes that could degrade cryptographic processing.

From these inputs the function produces three outputs. First, a flavour recommendation indicating the highest security level the hosting platform can sustain without resource failure. Second, the calibrated trust commitment, confirming the final security contract agreed with CoCo. Third, actuation directives dispatched to the execution plane to configure identity verification modes, set ledger audit frequencies, and enforce policy boundaries.

3.4 LEVEL OF TRUSTWORTHINESS (LoTw)

To formalise the negotiation and enforcement of trust, SAFE-6G introduces the Level of Trustworthiness (LoTw) paradigm, which operates across two sequential stages: the Non-calibrated Level of Trustworthiness (*nLoTw*) and the Calibrated Level of Trustworthiness (*cLoTw*).

The Non-calibrated Level of Trustworthiness is the initial security trust level that the Cognitive Coordinator derives from user or tenant intent. When an application or tenant requests network access, CoCo maps the user's security preferences into an initial *nLoTw* score and queries the Security Trust Function via an asynchronous message bus. Upon receiving the *nLoTw* query, the Security Trust Function evaluates its local hosting environment before committing to the requested security level. The function gathers instantaneous and trended telemetry metrics from Prometheus and Thanos, which feed into a multi-stage machine learning inference pipeline consisting of Gatekeeper, Predictor, and Decider models. A deterministic Logic Bridge then combines the inference outputs to determine whether deploying the cryptographic primitives demanded by *nLoTw* is computationally viable or whether edge resource constraints require a lower security tier. The Security Trust Function then reports back to CoCo with a proposed operational flavour.

The Calibrated Level of Trustworthiness is the final, binding trust contract established by CoCo. Because each Trust Function has visibility only over its own domain, a high security request that demands heavy post-quantum handshakes could consume compute cycles needed by the Safety or Reliability Trust Functions running on the same physical edge nodes. To resolve these conflicts, CoCo collects the flavour proposal from the Security Trust Function together with evaluations from the other four Trust Functions (Safety, Privacy, Resilience, and Reliability), detects cross-functional collisions, and reconciles competing resource demands against platform-wide policies. The resulting value is the confirmed *cLoTw*. CoCo dispatches this resolved score and the definitive flavour back to the Security Trust Function, which triggers service provisioning and policy configuration through the aerOS meta-orchestrator.

3.5 OPERATIONAL TRUST FLAVOURS

The resolved *cLoTw* maps into three discrete operational flavours: low, mid, and high. Each flavour activates a coordinated configuration across the three architectural layers of the Security Trust Function: the cognitive decision layer managed by the Local AI Agent, the application identity layer realised through Vertical Applications (vApps), and the network-facing enforcement tier realised through Network Applications (nApps). The specific cryptographic algorithms, protocol selections, and enforcement policies associated with each flavour are described in detail in Chapter 5. At a conceptual level, the three flavours are characterised as follows:

- **Low Flavour** ($cLoTw \leq 33$): A lightweight profile intended for resource-constrained edge environments or situations where sibling Trust Functions require low-latency precedence. The system minimises overhead by relying on classical cryptographic schemes, locally resolvable identifiers (did:key), and cached status checks, without resolving identifiers against the distributed ledger. Any registry operation that still reaches the ledger is forwarded by the Policy Enforcement Point without audit anchoring. If a policy violation occurs, the system terminates access without triggering escalation sequences.
- **Mid Flavour** ($33 < cLoTw \leq 66$): The standard baseline for enterprise-grade 6G environments. It implements full Self-Sovereign Identity verification using OpenID for Verifiable Presentations (OID4VP) with Digital Credentials Query Language (DCQL) queries. Identifiers are anchored to a private Hyperledger Fabric network via the did:safe6g method, and all ledger access is intermediated by the VDR-Invoker Policy Enforcement Point, which anchors audit records after execution on a best-effort basis. Policy deviations trigger soft enforcement actions, such as stepping up authentication or restricting tenant permissions, rather than terminating the session outright.
- **High Flavour** ($cLoTw > 66$): Targets mission-critical applications, high-value transactions, and regulated vertical slices. It requires quantum-resistant cryptography, combined with selective disclosure credentials that release only the claims required by the verifier. Inter-service communications are strictly gated, and all ledger transactions are preceded by mandatory audit anchoring to the SecurityAudits chaincode. Any policy anomaly or credential invalidity triggers hard enforcement: active sessions are revoked, identity attestations suspended, and security notifications sent to administrators.

This flavour model ensures that the Security Trust Function adapts its resource consumption and assurance level to the conditions determined by CoCo's cross-functional calibration, rather than imposing a uniform security overhead across all workloads and network slices.

3.6 ROLE OF SELF-SOVEREIGN IDENTITY (SSI)

Within this architecture, the Self-Sovereign Identity (SSI) layer acts as a bridge between traditional telecommunication IT systems and the emerging European Digital Identity (EUDI) Wallet ecosystem. The SSI design decouples identity verification from centralised providers, giving users and service components direct control over their credentials while giving relying parties verifiable attestations.

The SSI layer is organised around three roles: the Credential Issuer, Holder and Verifier. The Credential Issuer validates identity claims during onboarding, signs electronic attribute attestations, and exposes OID4VCI endpoints compatible with digital wallets. The Holder is the subject of the credentials and manages them within a digital wallet or a dedicated vault. The Holder receives credentials from trusted issuers, stores them securely, and presents cryptographic proofs to verifiers when required. Each credential binds contextual or personal claims to the Holder's cryptographic keypair, allowing the holder to prove possession later without exposing private keys or requiring connectivity to the issuer.

The Credential Verifier manages authentication sessions according to the active trust flavour. Depending on the negotiated security posture, the Verifier presents authentication challenges to the Holder through standard OpenID Connect redirections or through OID4VP exchanges using DCQL queries. During presentation processing, the Verifier validates the digital signatures, confirms proof-of-possession tokens, and queries distributed status lists to confirm the credential has not been revoked or suspended.

In SAFE-6G, this SSI architecture supports both user-facing metaverse applications and internal infrastructure protection. Within metaverse deployments, verifiable credentials carry role-based access entitlements, clearance levels, and operational privileges required for interaction within collaborative virtual spaces and industrial digital twins. Users authenticate through mobile or desktop digital wallets, with privacy preserved through selective disclosure. The Security Trust Function itself uses the same SSI primitives to authenticate administrative operations, protect inter-service APIs, validate tenant configurations, and produce audit logs anchored to the distributed ledger.

After a verification session completes, the Credential Verifier extracts the confirmed attributes and produces a standard authentication token. This token can be consumed by cloud-native micro-services or processed through OpenID Connect identity provider interfaces, maintaining backward compatibility with existing enterprise Identity and Access Management (IAM) systems.

4 REFERENCE ARCHITECTURE

The reference architecture of the Security Trust Function follows the general WP4 structure described in D4.1 [1] and the overall SAFE-6G reference architecture defined in D2.2 [3]. It is organised around three cooperating layers: the Local AI Agent, the Vertical Application (vApp) layer, and the Network Application (nApp) layer. These layers cooperate with the Cognitive Coordinator and other SAFE-6G components to evaluate security needs and enforce the appropriate level of protection. The conceptual responsibilities, trust signals, and flavour negotiation lifecycle are defined in Chapter 3; the detailed protocols, credential flows, and enforcement policies are described in Chapter 5. This chapter presents the structural organisation and inter-layer communication paths that connect these components into a unified security control plane.

The Local AI Agent sits at the cognitive decision tier. D4.1 describes the Security Function as combining local AI classification and forecasting with verifiable credentials and Hyperledger Fabric blockchain support for access control and auditability. In the current design, the Local AI Agent consumes trust-score messages from the Cognitive Coordinator via Kafka/Redpanda and operational metrics from the Prometheus/Thanos monitoring layer. It relies on forecasting and classification models served through KServe to determine whether the requested security posture can be sustained and which trust flavour should be activated. A deterministic Logic Bridge applies ceiling and veto constraints to the model outputs before producing the final recommendation. Once CoCo confirms the calibrated flavour, the agent forwards the deployment specification to the aerOS meta-orchestrator, which instantiates or reconfigures the corresponding vApp and nApp components on the target edge cluster.

The Vertical Application layer contains the security and identity orchestration logic exposed to higher-level services, including metaverse applications and identity-aware backends. From the SSI perspective, this layer manages the interaction between the Metaverse Manager or client application, the SSI Verifier, the OIDC provider or Webhook endpoint, the user's digital wallet, and the application backend. It is responsible for triggering authentication sessions, validating Verifiable Presentations, extracting confirmed claims, and returning usable authorisation information to the consuming application. The vApp layer also includes the DID Manager, the Credential Issuer, the Credential Verifier, the Credential Lifecycle Manager, and the wallet, which together provide the full SSI operational stack. In the implementation, these services are delivered by a single configurable component, the SSI Agent, whose role (Credential Issuer, Credential Verifier, or wallet) and enabled capabilities are set at deployment time according to the active operational flavour.

The Network Application layer translates security decisions into network-facing enforcement actions. In the current architecture, this layer is realised through the VDR-Invoker component, which operates as a Zero-Trust Policy Enforcement Point (PEP). The PEP intercepts all transactions targeting the Verifiable Data Registry, enforces policy constraints according to the active deployment flavour, records audit trails through the SecurityAudits Fabric chaincode, and proxies authorised requests to the ledger gateway. Integration with CAPIF-exposed APIs and 5G/6G control-plane interfaces remains part of the broader SAFE-6G integration path and is subject to validation against the final platform architecture.

These three layers share a common trust anchoring backend composed of the Fabric Gateway REST service, the Hyperledger Fabric peer and orderer nodes, the VerifiableDataRegistry chaincode, and the SecurityAudits chaincode. The high-level interaction follows a layered chain: the Security Trust Function interacts with the SSI Agent for DID and credential operations, the SSI Agent calls the Fabric Gateway, and the Fabric Gateway submits or queries transactions against Hyperledger Fabric over mTLS-secured gRPC channels.

The following figure illustrates the high-level architecture of the Security Function and shows how the Local AI Agent, the SSI Agent, the Verifiable Data Registry, the Fabric Gateway, and Hyperledger Fabric interact to provide identity, credential, and audit capabilities within SAFE-6G.

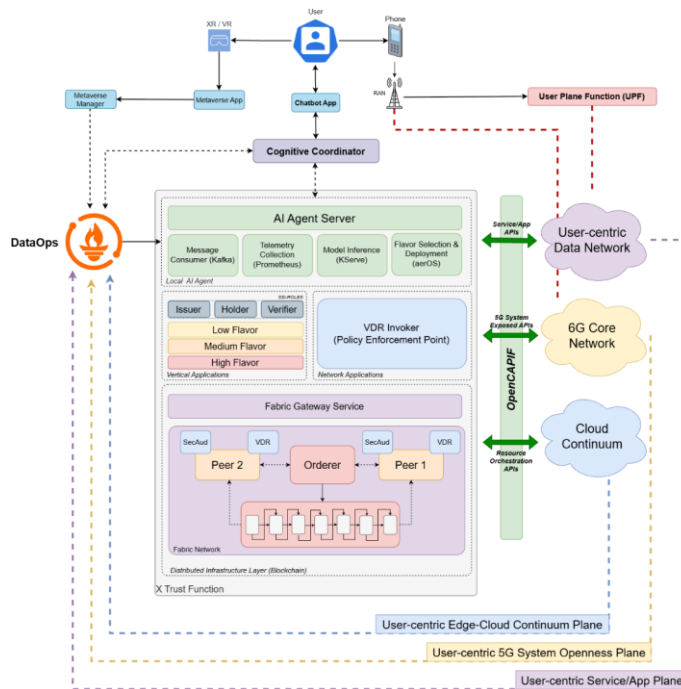


Figure 1: Security Function Architectural Components

The DID Manager provides an extensible framework for creating, storing, and resolving Decentralised Identifiers. It separates responsibilities between the DID service, the method provider, and the method resolver, allowing new DID methods to be introduced without modifying the overall SSI architecture. The `did:safe6g` provider extends this mechanism to create and resolve DIDs against the SAFE-6G Hyperledger Fabric chaincode. The Credential Lifecycle Manager handles credential status lists, lifecycle profiles, and status update operations. It supports multiple lifecycle providers and status list formats, including the W3C Bitstring Status List and the IETF Token Status List, allowing the system to revoke or suspend credentials without delegating all lifecycle logic to an external registry.

5 TRUST FUNCTION LAYERS

This section provides a structured analysis of the Security Trust Function across the three WP4 layers: the Local AI Agent, Vertical Applications (vApps), and Network Applications (nApps). Consistent with the WP4 reference design introduced in D4.1, trust decisions and enforcement logic are distributed across these three tiers, enabling security controls to be interpreted, evaluated, and enforced at different levels of abstraction. The three layers operate in a coordinated, closed-loop workflow:

- The **Local AI Agent** acts as the function's intelligence engine, evaluating real-time cluster telemetry against user trust intents to determine the system's operational security posture.
- The **Vertical Applications (vApps)** layer implements application-facing security services, introducing a decentralised identity framework for identity proofing, credential issuance, presentation verification, and key management.
- The **Network Applications (nApps)** layer serves as the network-facing enforcement tier, deploying a Policy Enforcement Point (PEP) to govern ledger access, tokenise operational actions, and interact with 6G network exposure APIs.

5.1 LOCAL AI AGENT

The Local AI Agent is the cognitive intelligence and decision-support component of the Security Trust Function. Operating at the edge-cloud control plane, it processes incoming trust requirements from the Cognitive Coordinator (CoCo), continuously monitors computing continuum telemetry, and executes machine learning inference to recommend and actuate operational configurations.

The following figure illustrates the complete end-to-end operational lifecycle of the Local AI Agent across its two phases; the subsequent subsections detail each stage of this lifecycle.

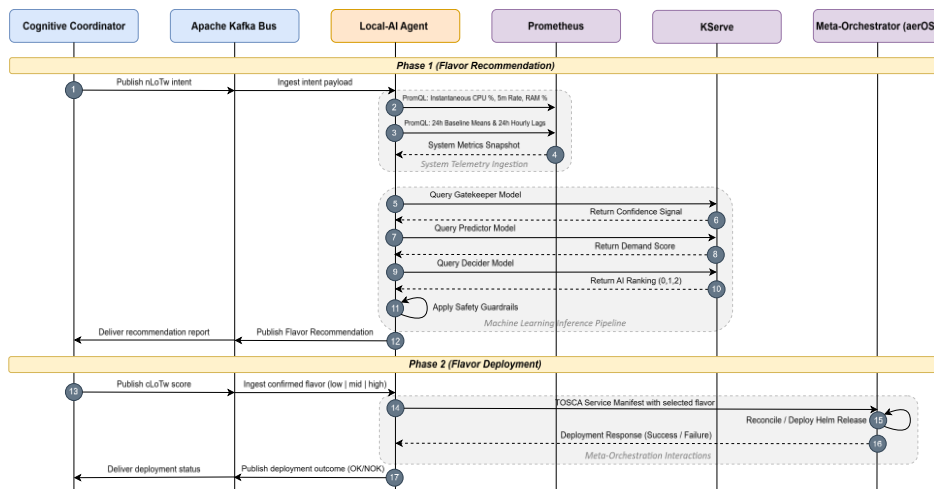


Figure 2: Local-AI Agent Operational Lifecycle

5.1.1 EVENT-DRIVEN RUNTIME & MESSAGING VIA KAFKA

The Local AI Agent is implemented as an asynchronous, event-driven service in Python, interfacing with the Cognitive Coordinator (CoCo) and cluster services over an Apache Kafka message bus. The interaction follows a two-phase protocol:

- **Phase 1 (Flavour Recommendation):** Upon receiving a user trust intent payload from CoCo containing the desired (non-calibrated) Level of Trustworthiness for security, the agent ingests real-time cluster telemetry, executes its internal machine learning inference, and publishes a structured evaluation report containing the recommended security flavour back to CoCo.
- **Phase 2 (Flavour Deployment):** After CoCo harmonises requirements across all active Trust Functions and returns the final (calibrated) Level of Trustworthiness, the agent initiates service reconfiguration or deployment via the aerOS meta-orchestrator and reports the final execution outcome back to CoCo.

5.1.2 SYSTEM TELEMETRY INGESTION VIA THANOS

To ensure that trust decisions are rigorously grounded in empirical infrastructure conditions, the agent gathers multi-dimensional telemetry by executing concurrent PromQL queries against a Thanos and Prometheus monitoring endpoint within the deployed cluster. This telemetry harvesting captures both instantaneous compute loads and memory saturation metrics, specifically polling the real-time average cluster CPU utilisation percentage, the 5-minute CPU rate-of-change percentage, and the cluster-wide allocation of physical memory.

Alongside real-time measurements, the agent ingests historical baselines and temporal lag series to contextualise current observations. It evaluates 24-hour moving averages across CPU ratios, rate-of-change metrics, and memory utilisation to establish baseline operating profiles, complemented by a sequential 24-hour hourly lag series of CPU measurements. Correlating instantaneous system state with long-term historical trajectories allows the agent to distinguish between transient, short-lived workload spikes and sustained resource exhaustion, thereby preventing premature security downgrades and safeguarding cluster stability.

5.1.3 MACHINE LEARNING INFERENCE PIPELINE VIA KSERVE

The ingested telemetry feed is processed by a multi-stage decision engine combining statistical trend inference with deterministic safety guardrails. The ML models are hosted on a Kubernetes-managed KServe model server, while the safety rules are enforced locally via a Deterministic Logic Bridge:

1. **Gatekeeper Model:** Functions as an admission controller evaluating instantaneous system resources and platform readiness. It ingests current CPU, 5-minute CPU rate, memory saturation, 24-hour baselines, and the user's requested score. It outputs a resource confidence signal for the selected flavour.
2. **Predictor Model:** Analyses the 24-hour hourly CPU lag series, 12-hour rolling averages, and temporal calendar features (hour, day of week, month start) to forecast cluster workload trends over the subsequent hour, outputting a projected resources demand score.

3. **Decider Model:** Combines the Gatekeeper's confidence signal, the Predictor's demand score, instantaneous CPU utilisation, and memory used percentage to generate an unconstrained AI recommendation: an operational rank index (ai_{rank} : 0 for low, 1 for mid, 2 for high).

To uphold user sovereignty and protect edge nodes from saturation, the unconstrained AI recommendation is filtered through two inviolable deterministic guardrails. The user's intent score (n_{score} , ranging from 0 to 100) dictates an intended ceiling rank:

$$intended_{rank} = \begin{cases} 0, & \text{if } n_{score} \leq 33 - (low) \\ 1, & \text{if } 33 < n_{score} \leq 66 - (mid) \\ 2, & \text{if } n_{score} > 66 - (high) \end{cases}$$

The final rank calculation is capped using the intended rank, ensuring that the system never over-provisions security resources beyond what was requested:

$$final_{rank} = \min(ai_{rank}, intended_{rank})$$

If continuum resources fall below acceptable operating parameters, this is reflected in the Gatekeeper's confidence signal. The system then overrides the decision to low, shedding high-overhead workloads during severe resource strains.

The resulting rank is mapped to the target operational security flavour string (low, mid, or high), which is published as the recommended flavour in the Phase 1 evaluation report. The flavour that is finally deployed is the one confirmed by CoCo in Phase 2 after cross-functional calibration, which the Local AI Agent then actuates through aerOS.

5.1.4 META-ORCHESTRATION INTERACTIONS VIA AEROS

Once CoCo has confirmed the calibrated flavour in Phase 2, the Local AI Agent actuates the operational decision by invoking the aerOS meta-orchestrator client. The agent parameterises an OASIS TOSCA Simple Profile (v1.3) service manifest, binding the containerised *security-function-vapps* Helm chart and the selected flavour configuration values. aerOS then reconciles the micro-services across the target edge Kubernetes cluster, completing the autonomous decision loop.

5.2 VERTICAL APPLICATIONS (VAPPS)

The Vertical Application (vApp) layer serves as the application-level trust tier of the Security Trust Function. While the Local AI Agent evaluates telemetry and recommends the operational security flavour, and the Network Application (nApp) layer enforces perimeter policies at the blockchain boundary, the vApp tier implements decentralised identity services, user-centric credential lifecycles, and policy enforcement within 6G service verticals. In contrast to legacy approaches that confine identity to static, network-operator managed credentials or centralised identity databases, the vApp layer places control directly in the hands of the user. Built in alignment with European Digital Identity (EUDI) architectural principles, the vApp ecosystem provides modular Self-Sovereign Identity (SSI) services spanning user-facing wallets, higher-level enterprise systems (such as the Metaverse Manager and OpenID Connect identity providers), and distributed ledger registries. By translating

cognitive trust decisions into concrete application-level access policies, the vApp layer ensures that security flavour changes are reflected in real time across user sessions, resource permissions, and operational privileges.

5.2.1 SSI ROLES AND DECENTRALISED IDENTIFIERS

The vApp layer is the primary integration point for SSI-based authentication and authorisation flows within SAFE-6G. It allows vertical services, including industrial metaverse environments, digital twin platforms, and edge microservices, to require authentication based on Verifiable Credentials and to receive cryptographically verified claims that can be translated into application-level access decisions. The layer is structured around three functional roles:

- **Credential Issuer:** Operated by authorised administrative entities, the Credential Issuer manages the creation, cryptographic signing, and distribution of Verifiable Credentials. It processes credential requests, generates signed attestations using standardised issuance protocols (OID4VCI), and registers the corresponding lifecycle status entries on the ledger.
- **Credential Verifier:** Embedded within vertical service gateways or identity mediation layers, it handles authentication challenges on behalf of consuming applications. It validates cryptographic proofs submitted by wallets, checks ledger-anchored credential status lists, resolves issuer identifiers, and extracts verified claims for downstream access control.
- **Holder / SSI Vault:** Acting as the client-side wallet, the SSI Vault manages private key storage and credential presentation. Whether running within a browser-based portal or an edge runtime, the Vault ensures that private keys remain exclusively under user custody. It requests explicit user consent before disclosing identity data, supports selective claim presentation to limit unnecessary data exposure, and submits signed Verifiable Presentations to verifiers.

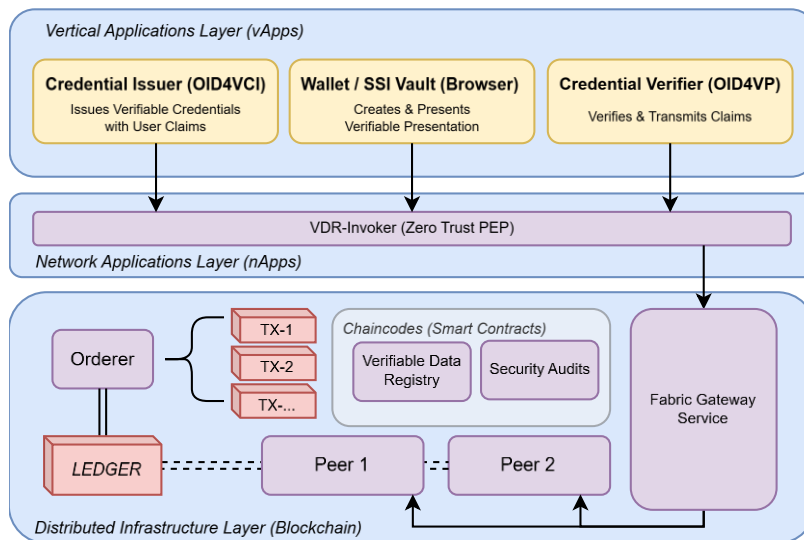


Figure 3: Vertical Applications (vApps) SSI Architecture

Identity subjects and services within this ecosystem are anchored using the did:safe6g Decentralised Identifier method, a blockchain-backed DID scheme tailored for 6G multi-tenant networks. Creating a did:safe6g identifier requires generating a cryptographic key pair, constructing a W3C-compliant DID Document, and registering it on the Hyperledger Fabric ledger through the SAFE-6G Gateway. By default, identifiers are bound to high-performance Edwards-curve keys (Ed25519), but the method supports crypto-agile key binding, including post-quantum assertion keys such as ML-DSA-65 for long-term identity protection. Resolution of did:safe6g identifiers is handled by querying the VerifiableDataRegistry chaincode to retrieve the authoritative DID Document, while deactivation updates the on-chain record to retire the identifier.

To maintain alignment with the multi-tier trust model, the vApp layer instantiates the calibrated flavour confirmed by CoCo and actuated by the Local AI Agent. Under the low flavour, the tier activates a lightweight profile suited to latency-critical edge deployments: credentials are signed with classical keys, identifiers use locally resolvable methods such as did:key, and status checks rely on cached status lists, so no interaction with the distributed ledger is required. Under the default mid flavour, the layer enables the did:safe6g method and the OID4VCI and OID4VP protocols, resolving DIDs on the ledger at verification time and checking credential status against the current status list. Under the high flavour, the layer applies maximum cryptographic assurance: dual classical and post-quantum signature validation, selective claim disclosure, and a fresh, synchronous status check for every presentation.

Decision coordination at the vApp layer follows a graduated enforcement model. Soft enforcement is applied when policy discrepancies are minor or trust scores fluctuate within acceptable bounds. This includes step-up authentication challenges, user notifications, requests for additional credential presentation, or temporary restriction to reduced privilege tiers. Hard enforcement is triggered by cryptographic validation failure, revoked credential status, or critical trust anomalies. In these cases, the vApp tier terminates the session, denies access, and flags the event across ledger-backed audit channels.

5.2.2 CREDENTIAL ISSUANCE & VERIFICATION

The operational lifecycle of decentralised identity within the vApp tier centres on two complementary workflows: credential issuance and credential verification. Both follow OpenID for Verifiable Credential Issuance (OID4VCI) and OpenID for Verifiable Presentations (OID4VP) specifications, avoiding proprietary protocol lock-in and ensuring interoperability with EUDI-aligned architectures.

5.2.2.1 CREDENTIAL ISSUANCE VIA OID4VCI

Credential issuance is the process by which an authoritative entity binds identity and role attributes to a user's decentralised identifier. The flow begins when a user submits a credential request through the issuer portal operated by the Credential Issuer. The request specifies the credential type, such as a metaverse operator clearance, and the relevant subject metadata. After validating the request, the Credential Issuer generates an OID4VCI credential offer containing the issuer endpoint, available credential configurations, and pre-authorised transaction parameters.

The credential offer is presented to the user through the portal interface, either as scannable QR code for cross-device sessions or as a deep link for same-device execution. Activating the offer redirects the interaction to the SSI Vault. After the user authenticates and provides consent, the Vault initiates the OID4VCI protocol exchange with the Credential Issuer, proving possession of its private key and binding its did:safe6g identifier to the credential. Before signing the attestation, the Credential Issuer resolves the subject's did:safe6g DID Document from the VerifiableDataRegistry smart contract via the VDR-Invoker PEP to validate key binding and policy adherence. Once verified, the Issuer signs the credential and delivers it through the portal session to the Vault, where it is stored in encrypted local storage for subsequent authentication sessions.

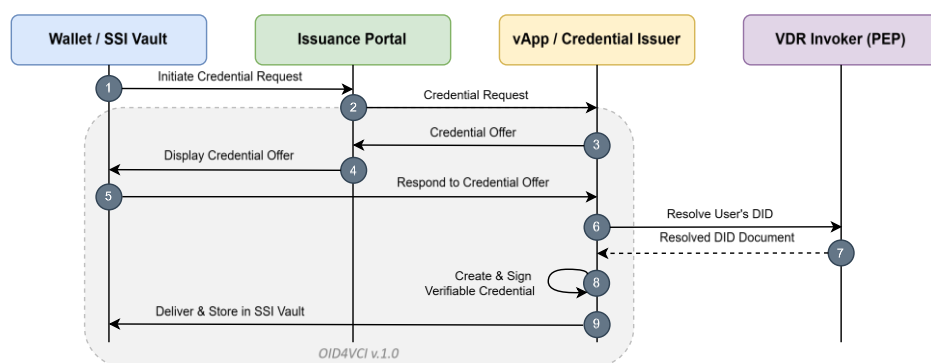


Figure 4: OID4VCI Credential Issuance Sequence

5.2.2.2 CREDENTIAL VERIFICATION VIA OID4VP

Credential verification governs the presentation and validation of identity claims when a user or automated agent requests access to a vertical resource. The Credential Verifier constructs an OID4VP authorisation request bound to an ephemeral session, the wallet compiles a Verifiable Presentation matching the requested claims, and the verifier evaluates the cryptographic proof before returning extracted claims to the consuming application. In both integration modes described below, the SSI Vault acts as the wallet-side component: it checks the authenticity of the authorisation request using the verifier's decentralised identifier, evaluates the requested disclosures against user privacy preferences, prompts for consent, and submits the signed presentation. To accommodate different application architectures, the vApp layer supports two integration modes: OpenID Connect (OIDC) and direct Webhook:

- OpenID Connect:** In this mode, the client application, for example the Metaverse Manager, delegates authentication to an OpenID Connect provider such as Keycloak, indicating which attestation is required for the session. The identity provider forwards the SSI-specific authentication request to the Credential Verifier, which generates an OID4VP authorisation request and displays it to the user as a QR code or same-device redirect. The SSI Vault then presents the required credential.

The Credential Verifier validates the submitted presentation: it checks the cryptographic signature, resolves the issuer's did:safe6g document, queries the ledger-anchored revocation list, and confirms policy compliance. On success, the client receives a signed OIDC ID token containing the validated claims extracted from the credential, enabling the application to create a user session and enforce role-based or attribute-based access decisions through standard OIDC mechanisms.

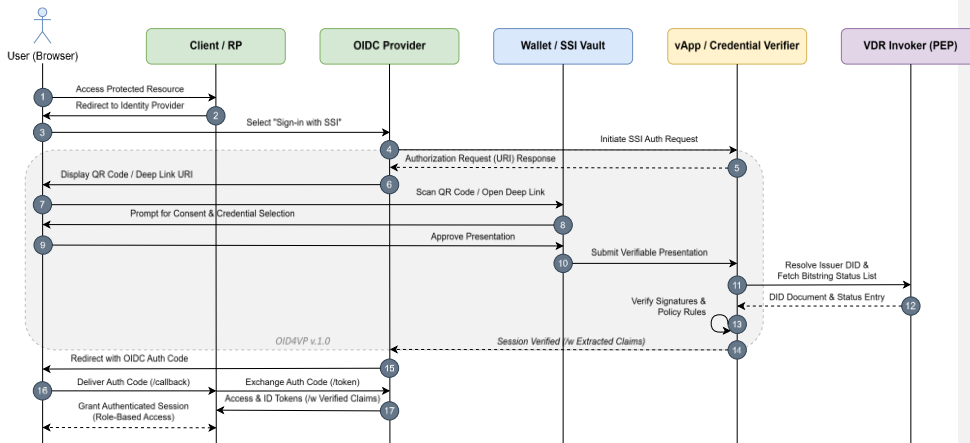


Figure 5: OID4VP Credential Verification via OIDC Flow

- Webhook:** In this mode, the client application initiates an authentication session directly with the Credential Verifier by issuing an API call with a predefined scope. The scope reflects the active security flavour and the type of attestation required. The verifier returns a session identifier and an OID4VP authorisation request, which the client displays to the user as a QR code or same-device interaction. The user then opens the SSI Vault and consents to present the requested credential.

After the SSI Vault submits the presentation, the Credential Verifier validates the proof against ledger state. On success, the verifier exposes the result under the session identifier. The client application polls or retrieves the outcome, receiving a minimised set of verified claims, such as subject identifier, organisational role, and use-case-specific permissions. The client can then create the application session or retrieve the current user role from the Metaverse Manager. This dual-mode design allows SAFE-6G to support both standardised enterprise federations and lightweight, low-latency edge integrations.

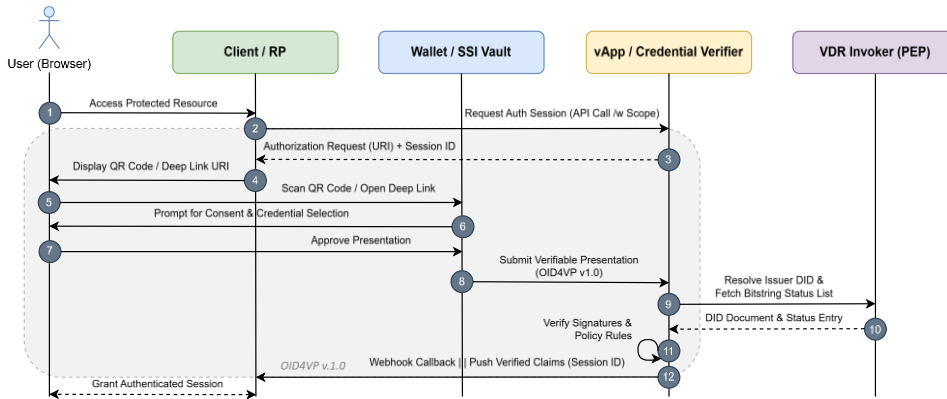


Figure 6: Credential Verification via Webhook Flow

5.2.3 CREDENTIAL LIFECYCLE MANAGEMENT

The integrity of decentralised trust in 6G environments depends on the ability to monitor, update, and invalidate credentials throughout their operational lifetime. The Credential Lifecycle Manager tracks credential states and enforces validity rules during every verification exchange. Rather than treating credentials as static tokens, the architecture models them as dynamic security assets subject to suspension, revocation, and temporal expiration.

To achieve scalable status tracking without degrading edge performance or leaking privacy metadata, the architecture implements the W3C Bitstring Status List specification. Credential validity is encoded as a compact, bit-indexed array where each bit position maps to a single credential's validity flag. Status lists support separate operational purposes, distinguishing definitive revocation from temporary suspension. These bitstrings are compressed using gzip encoding, allowing thousands of credential states to be conveyed in payloads of only a few kilobytes. This reduces bandwidth overhead during high-density edge verifications and prevents external observers from correlating individual subjects through public status queries.

Authoritative status lists are anchored on the Hyperledger Fabric ledger via the VerifiableDataRegistry smart contract. When an administrative authority or an automated security monitor detects credential compromise, anomalous behaviour, or role reassignment, the Credential Lifecycle Manager submits a transaction updating the target bit index. The distributed ledger provides a consistent and verifiable record of all status changes, ensuring that every verifier observes the same credential state.

During verification, the Credential Verifier extracts the status list pointer embedded in the presented credential and checks the referenced bit index against the ledger. The verification depth is tuned to the active security flavour: under the low flavour, verifiers check status against locally cached copies of the status list; under the mid flavour, verifiers check status at verification time against the current status list, accepting a cached copy only within its validity period; and under the high flavour, verifiers require a fresh, synchronous status query for every presentation to eliminate any risk of operating on stale revocation data.

If a credential is found to be suspended, for example pending administrative re-verification, the system applies the soft enforcement measures described in Section 5.2.1, such as step-up authentication or temporary privilege restriction. If a credential is expired, corrupted, or definitively revoked, hard enforcement is engaged: the verifier terminates the authentication flow, rejects the session, and logs the event to the network audit channel.

The SSI Agent wraps the compressed bitstring in a Bitstring Status List Credential, which is secured as a JWT and published by the Credential Issuer at a stable status list URL from which verifiers retrieve it. The credential subject declares the list type and its status purpose (revocation). A complete example is provided in Annex A1.3 (Table 9).

5.2.4 METAVERSE CREDENTIAL DATA MODEL

To support fine-grained access control across industrial 6G use cases, the vApp layer defines a standardised credential schema designated as the MetaverseCredential. Structured in compliance with the W3C Verifiable Credentials data model, this schema encapsulates both permanent identity attributes and dynamic operational clearances. Credentials across the SAFE-6G SSI framework are issued as JWT-secured Verifiable Credentials (VC-JWT) through OID4VCI endpoints. As the signature is carried by the JWT envelope, the credential payload contains no embedded proof object; it combines the Verifiable Credentials Data Model (VCDM) 1.1 base context expected by JWT-based verifiers with the VCDM 2.0 context that defines the Bitstring Status List terms. The credential links cryptographic proofs to application-level access decisions, allowing vertical applications to parse verified claims directly or map them into standard token structures during federated OpenID Connect sessions.

The data model accommodates the operational roles required by smart factory and industrial metaverse use cases defined in D2.3 [4]. It supports five standardised roles: line-manager, production-director, factory-digital-twin, instructor, and technician. By binding these roles within cryptographically signed subject claims, the credential allows authorisation engines to grant contextual access to industrial telemetry, robotics actuation interfaces, and virtual twin environments without relying on centralised user databases.

Beyond static identity attributes, the credential includes authorisation constraints and lifecycle metadata. Role designations are paired with explicit clearance levels and permissible zones, such as specific production lines or restricted sub-plant sectors. Every issued credential also embeds a reference to its lifecycle registry via a Bitstring Status List entry pointer, enabling verifiers to check validity against the ledger-backed status lists described in Section 5.2.3. An example credential payload is provided in Annex A1.3 (Table 10).

When the Credential Verifier processes these attributes during an authentication session, they are mapped into the consuming application's authorisation logic. In the OIDC integration mode, the verifier transforms `primaryRole`, `clearanceLevel`, and `permittedZones` into standard JWT claims, allowing the Metaverse Manager to apply role-based access control natively. In the Webhook mode, the verified claims are returned directly over the API boundary, enabling edge microservices to validate device and zone clearances with minimal latency.

5.2.5 POST-QUANTUM CRYPTOGRAPHIC OPERATIONS

To protect 6G identity infrastructure against harvest-now-decrypt-later threats posed by future quantum computers, the vApp layer integrates a post-quantum cryptographic subsystem. Rather than deferring quantum resistance to a future migration, the architecture implements a modular, crypto-agile Key Management System (KMS) plugin that allows services to switch between classical and quantum-safe primitives based on the active security flavour.

The KMS supports quantum-resistant digital signatures and Key Encapsulation Mechanisms (KEM) aligned with current NIST Post-Quantum Cryptography standards. For digital signatures, the system implements NIST FIPS 204 ML-DSA (Module-Lattice-Based Digital Signature Algorithm, at parameter sets ML-DSA-44, ML-DSA-65, and ML-DSA-87), Falcon (Falcon-512 and Falcon-1024), which NIST has selected for standardisation as FN-DSA (FIPS 206, in development), and NIST FIPS 205 SLH-DSA (Stateless Hash-Based Digital Signature Algorithm). Falcon offers compact signature sizes suited to bandwidth-constrained 6G edge links, while ML-DSA serves as the primary signing algorithm for Verifiable Credentials. For key agreement and credential encryption, the system implements NIST FIPS 203 ML-KEM (ML-KEM-512, ML-KEM-768, and ML-KEM-1024).

To reduce the risks inherent in adopting newly standardised lattice-based cryptography, the implementation also supports hybrid schemes. These include X-Wing, a hybrid key encapsulation mechanism combining ML-KEM-768 with classical X25519, and ML-KEM-768-P256, which pairs ML-KEM-768 with NIST P-256. Hybrid schemes ensure that cryptographic channels retain classical security guarantees even if a future vulnerability is found in lattice-based constructions.

Within the SAFE-6G operational model, post-quantum capabilities are tied to the active security flavour. The low and mid flavours use Edwards-curve signatures (Ed25519) to maximise throughput and minimise verification latency on constrained edge devices. Transitioning to the high flavour activates the post-quantum subsystem: newly registered did:safe6g DID Documents include post-quantum verification methods (such as the project-defined ML-DSA-65-VerificationKey2024 type) under the assertionMethod relationship, and the Credential Issuer applies dual classical and quantum-safe signatures to issued credentials. This design ensures that high-assurance services achieve post-quantum protection without altering the OID4VCI or OID4VP protocol workflows.

Algorithm	Standard / Specification	Primitive Type	Flavour(s)	Role in SAFE-6G
Ed25519	RFC 8032	Classical Signature	Low, Mid, High	Baseline for VC/VP signing; ultra-low latency verification on constrained edge devices; classical leg of dual-signature high flavour.
ML-DSA-65	NIST FIPS 204	Lattice-based Signature	High	Primary quantum-safe digital signature for Verifiable Credentials and did:safe6g assertion methods.

Falcon-512	NIST FN-DSA (FIPS 206, in development)	Lattice-based Signature	High (Optional)	High-performance compact signature profile suited to bandwidth-constrained 6G radio channels.
SLH-DSA	NIST FIPS 205	Hash-based Signature	High (Fallback)	Conservative stateless hash signature ensuring resilience against future algebraic weaknesses in lattice primitives.
ML-KEM-768	NIST FIPS 203	Lattice-based KEM	High	Post-quantum key encapsulation for encrypted credential exchange and secure session establishment.
X-Wing	IETF Draft	Hybrid KEM (ML-KEM-768 + X25519)	High	Primary hybrid key agreement, combining post-quantum security with classical elliptic-curve guarantees.
ML-KEM-768-P256	Hybrid Profile	Hybrid KEM (ML-KEM-768 + NIST P-256)	High	FIPS/enterprise-compliant hybrid key encapsulation for federated identity and enterprise vertical integrations.

Table 2: Available Cryptographic Algorithms

Under the high security flavour, the Credential Issuer generates composite verifiable credentials carrying both a classical Ed25519 signature and a quantum-resistant ML-DSA-65 signature; the post-quantum signature uses a project-defined proof type, as no ML-DSA suite has yet been standardised for Verifiable Credentials. Verifiers resolve both public verification keys from the authoritative did:safe6g DID Document registered on Hyperledger Fabric, requiring both cryptographic assertions to succeed before granting access.

5.3 NETWORK APPLICATIONS (NAPPS)

The Network Application (nApp) layer serves as the network-facing enforcement tier of the Security Trust Function. While the vApp layer manages application-level credential semantics, the nApp layer operationalises security decisions across network sessions, service exposure interfaces, and blockchain interactions. It is realised through the VDR-Invoker, a high-performance network application that functions as a Zero-Trust Policy Enforcement Point (PEP).

The following sequence illustrates the operational architecture and transaction lifecycle of the VDR-Invoker, showing how external calls are authenticated, handled across the active security flavour, and audited onto the distributed ledger before execution.

Commented [GH1]: Not sure it's relevant to quote some PQC algo that are not standard like Falcon and W-Wing

Commented [GH2]: same thing. Better to deal with standard ?

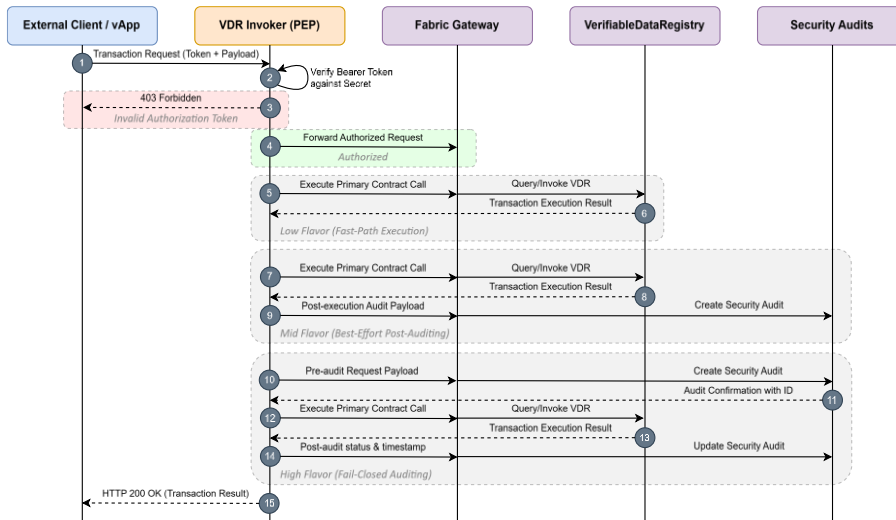


Figure 7: VDR-Invoker Operational Architecture & Transaction Lifecycle

5.3.1 ZERO-TRUST POLICY ENFORCEMENT POINT (PEP)

The VDR-Invoker is implemented to satisfy strict 6G low-latency response requirements while establishing a secure defence perimeter. Deployed as a network boundary gateway, it protects the underlying Hyperledger Fabric blockchain infrastructure from direct exposure to external application clients. In this topology, external services, such as credential issuers, metaverse managers, and edge microservices, are strictly prevented from establishing direct communications with Hyperledger Fabric peer nodes, orderers, or internal gRPC channels. Instead, all ledger interactions must traverse the Policy Enforcement Point, which functions as an authoritative intermediary proxying validated requests to the upstream Fabric Gateway service.

To uphold zero-trust principles at the network perimeter, every incoming service request is systematically intercepted by an authentication gateway filter before reaching any blockchain primitives. This filter enforces continuous tokenised authentication by validating incoming cryptographic bearer tokens against hashed security credentials maintained at the gateway tier. Unauthorised or malformed caller requests are immediately terminated at the edge with an explicit access denial, effectively thwarting unauthorised ledger traversal and preventing computing exhaustion on consensus nodes.

Beyond perimeter defence and identity mediation, the Policy Enforcement Point provides comprehensive transaction payload abstraction. It ingests standard REST-based service requests from application clients and dynamically marshals them into structured Hyperledger Fabric transaction envelopes. By encapsulating channel configurations, cryptographic endorsements, and chaincode invocation semantics within the enforcement layer, the architecture substantially insulates application developers from ledger-specific complexities while ensuring consistent cryptographic and policy invariants across all distributed ledger interactions.

Because the enforcement point keeps no session state between requests, it can be replicated horizontally behind a Kubernetes service: additional replicas absorb load peaks, and the remaining replicas take over transparently if an individual pod fails, so the enforcement layer does not become a single point of failure.

5.3.2 DYNAMIC FLAVOUR ENFORCEMENT & LEDGER AUDITING

The primary responsibility of the VDR-Invoker is to enforce the active operational security flavour (low, mid, or high) as calibrated by CoCo and actuated by the Local AI Agent. The enforcement engine dynamically adapts its operational behaviour and blockchain auditing rigour based on this active profile:

- **Low Flavour (Fast-Path Ledger Execution):** For resource-constrained or latency-critical edge environments, the enforcement point completely bypasses the SecurityAudits smart contract. Authorised queries and invocations are forwarded directly to the VerifiableDataRegistry without the overhead of blockchain audit state writes, maximising transaction throughput.
- **Mid Flavour (Best-Effort Post-Execution Auditing):** In the default operational profile, the enforcement point executes the primary transaction against the VerifiableDataRegistry first to eliminate pre-flight latency. Following successful execution, it anchors the audit trail to the SecurityAudits smart contract in a single post-execution transaction. To ensure application resilience, audit anchoring failures in this profile generate operational warnings without blocking the client's business transaction.
- **High Flavour (Strict, Fail-Closed Auditing):** Prior to executing any transaction against the VerifiableDataRegistry smart contract, the Policy Enforcement Point constructs an audit record and writes a pre-execution entry to the SecurityAudits chaincode. If this initial audit write fails for any reason, the entire transaction is aborted immediately under a strict policy violation, returning an access denial to the caller. If the audit succeeds, the primary registry transaction is executed, after which the enforcement point invokes the SecurityAudits contract again to record the final execution status, completion score, and timestamps.

To ensure seamless service integration across heterogeneous 6G administrative domains, the nApp layer is designed to expose these registry and verification interfaces in alignment with the 3GPP Common API Framework (CAPIF). Key endpoints, including decentralised identifier resolution and audit query APIs, are intended for registration with the SAFE-6G OpenCAPIF Core Function, subject to validation against the final platform architecture. This would allow the Cognitive Coordinator (CoCo) and meta-orchestration frameworks to consume standardised interfaces for real-time security telemetry and cross-domain policy coordination, bridging high-level cognitive intent with low-level distributed ledger execution.

6 TRUST EVALUATION & ENFORCEMENT LOGIC

The Security Trust Function operates as a closed-loop control system in which trust evaluation and enforcement are distributed coherently across its three cooperating layers. Trust is not evaluated at a single point but emerges from the interaction between the cognitive intelligence of the Local AI Agent, the identity and credential assurance provided by the vApp layer, and the ledger-anchored policy rigour enforced by the nApp layer. Trust evaluation draws on a combination of user intent, contextual system metrics, identity evidence, credential validity, and policy rules, producing an adaptive security posture that aligns operational constraints with declared trust requirements.

Evaluation is initiated when the Cognitive Coordinator forwards a user trust intent payload to the Local AI Agent, encoding the requested non-calibrated Level of Trustworthiness for the security dimension. The Local AI Agent ingests this intent alongside real-time and historical cluster telemetry, executes its three-stage machine learning inference pipeline, and applies two deterministic guardrails: the Intent Ceiling, which prevents the system from provisioning a higher security tier than the user requested, and the Gatekeeper Veto, which overrides the AI recommendation to the low flavour under severe resource exhaustion. The resulting flavour recommendation is returned to the Cognitive Coordinator as a Phase 1 evaluation report. After the Cognitive Coordinator harmonises requirements across all active Trust Functions and returns the calibrated Level of Trustworthiness, the Local AI Agent actuates the corresponding deployment via the aerOS meta-orchestrator, completing the autonomous decision loop.

Once the operational flavour is active, SSI-based enforcement at the vApp tier follows a defined verification sequence. The user or service initiates an access request, and the Security Trust Function selects the required authentication profile according to the active flavour and trust context. If SSI-based authentication is required, the Credential Verifier generates an OID4VP presentation request. The SSI Vault presents the required Verifiable Credential, and the Credential Verifier checks the cryptographic proof, resolves the issuer's did:safe6g DID Document from the VerifiableDataRegistry, evaluates the Bitstring Status List entry, and confirms policy compliance. The verified claims, including roles, clearance levels, and permitted operational zones, are then returned to the consuming application or identity provider to support authorisation decisions. The cryptographic depth of this verification sequence is governed by the active flavour: the low flavour applies lightweight validation with locally resolved identifiers and cached status checks; the mid flavour enforces full ledger-anchored DID resolution and status verification; and the high flavour additionally requires dual classical and post-quantum signature validation and synchronous revocation queries.

At the nApp tier, the same flavour governs the auditing rigour applied to every distributed ledger interaction via the VDR-Invoker. Under the low flavour, the enforcement point bypasses the SecurityAudits smart contract entirely to maximise throughput at the edge. Under the mid flavour, primary registry transactions execute without pre-flight overhead, with audit records anchored after execution on a best-effort basis, so that audit failures raise warnings without blocking the transaction. Under the high flavour, the enforcement point operates in fail-closed mode: a pre-execution audit

record must be committed to the SecurityAudits chaincode before any VerifiableDataRegistry transaction proceeds, and audit failure results in immediate transaction abort and access denial.

Across both tiers, enforcement actions follow a graduated model based on the severity of the policy violation. Soft enforcement is applied when credential status is suspended or trust scores fluctuate within acceptable bounds, and the system issues step-up authentication challenges, requests additional credential presentations, or restricts the session to a reduced privilege tier while maintaining service continuity. Hard enforcement is triggered when cryptographic validation fails, credentials are definitively revoked or expired, or a pre-execution audit fails at the nApp tier: the session is terminated, access is denied, and the event is recorded as an immutable entry on the Hyperledger Fabric ledger via the SecurityAudits chaincode. This graduated model ensures that the Security Trust Function remains resilient under partial trust degradation while maintaining strict correctness guarantees during cryptographic validation.

7 FUTURE RESEARCH PATHWAYS

All core architectural components, cryptographic primitives, smart contracts, and microservices specified under Task 4.3 have been implemented and component-tested within WP4, with the source code publicly available in the project repository [19]. Extended end-to-end multi-cell testbed validation, large-scale stress testing, and official project KPI measurements (as specified in D2.4 [5]) are formally allocated to Work Package 5 (WP5). The following table outlines the maturity level of each D4.4 component at the M33 handover.

Component	Status	Description	Validation (WP5)
Local AI Agent	Implemented	Standalone ML model evaluation & synthetic telemetry testing	Real-time telemetry ingestion & inference latency in D5.2/D5.3
SSI Vault / Wallet	Implemented	Credential issuance and presentation flow validation	Multi-user edge wallet performance & storage testing
DID Manager (did:safe6g)	Implemented	Unit-tested DID creation, resolution, and key-rotation flows	On-chain resolution latency (<100 ms) in testbed
Credential Lifecycle Manager	Implemented	Bitstring Status List v1.0 encoding & status check testing	Revocation query evaluation under high concurrency
VDR-Invoker (PEP)	Implemented	REST API & OpenCAPIF invoker endpoint verification	Zero-Trust access decision throughput (TPS) in D5.3
VerifiableDataRegistry Chaincode	Implemented	Hyperledger Fabric local peer deployment & CRUD testing	Multi-node DLT storage & ledger query benchmarking
SecurityAudits Chaincode	Implemented	Audit transaction generation & chaincode execution testing	End-to-end audit trail throughput & block finalisation latency

Table 3: Security Trust Function Component Maturity

The User-centric Security Trust Function developed within SAFE-6G establishes a comprehensive technological foundation for adaptive security management in distributed 6G environments. Its architecture combines infrastructure-aware security assessment, machine-learning-supported trust decisions, Self-Sovereign Identity, decentralised credential management, zero-trust policy enforcement, distributed auditability, and crypto-agile security mechanisms within a common operational framework. A particular strength of the implemented approach lies in the coordination of these capabilities rather than their deployment as isolated security mechanisms. Through the operational security flavour model, SAFE-6G links user-defined trust requirements and runtime infrastructure conditions with corresponding identity, cryptographic, access-control, and auditing mechanisms. This creates a flexible foundation capable of accommodating different security requirements and deployment conditions while preserving a common trust-management architecture.

Building on these results, several pathways emerge to extend the concepts and technological assets developed within SAFE-6G beyond the objectives and implementation scope of the project. These pathways concern both the evolution of the integrated Security Trust Function and the reuse of individual components as building blocks for broader research and innovation activities.

7.1 ADVANCED ADAPTIVE TRUST ORCHESTRATION

The adaptive trust-management mechanisms introduced in SAFE-6G provide a natural starting point for more sophisticated forms of security orchestration. By combining user-defined trust requirements, infrastructure telemetry, machine-learning-supported assessment, deterministic safeguards, and coordinated security flavours, the architecture already supports security configurations that respond to both user expectations and operational conditions. This foundation can be extended by incorporating richer contextual information into trust decisions, including application characteristics, service criticality, deployment-specific policies, and additional operational conditions. Such information could support increasingly fine-grained adaptation of security configurations while preserving the core project principle of balancing security assurance with available infrastructure resources. The same approach can also be expanded towards multi-dimensional trust orchestration, where security considerations are coordinated with other trustworthiness dimensions such as privacy, resilience, reliability, and safety. In this way, the trust negotiation and flavour-based mechanisms can serve as the basis for broader orchestration models across heterogeneous and dynamic edge-cloud environments.

7.2 EVOLUTION OF CRYPTO-AGILE AND POST-QUANTUM TRUST

The crypto-agile foundation established within the Security Trust Function introduces support for classical, post-quantum, and hybrid cryptographic mechanisms within a common security architecture. Its combination with decentralised identity and Verifiable Credentials creates a particularly relevant basis for the evolution of quantum-resistant trust infrastructures. A promising direction is the development of adaptive cryptographic policies in which algorithm selection and assurance levels respond to operational requirements and deployment conditions. This could enable different cryptographic profiles to be applied according to service criticality, device capabilities, security requirements, or resource availability. The existing support for classical, post-quantum, and hybrid mechanisms also creates a suitable environment for investigating transition strategies between cryptographic generations. Hybrid classical/post-quantum schemes, policy-driven algorithm selection, and crypto-agility across heterogeneous devices could be examined together with their implications for computational overhead, communication cost, and latency.

7.3 EXTENDED IDENTITY AND TRUST INTEROPERABILITY

The SAFE-6G vApp layer provides a modular identity ecosystem covering credential issuance and verification, Decentralised Identifiers, credential lifecycle management, standard OpenID-based interactions, and multiple application integration mechanisms. This modularity provides a strong basis for extending the reach of the Security Trust Function towards additional identity ecosystems and operational domains. Broader interoperability could encompass different wallet implementations,

identity providers, credential profiles, trust registries, and vertical platforms. At the same time, the extensible design of the DID Manager enables alternative identifier methods and trust-anchor configurations to be incorporated while maintaining the common identity-management structure established within SAFE-6G. These capabilities also create a pathway towards cross-domain trust relationships. The same identity and credential mechanisms could be applied across telecommunications infrastructures, enterprise services, edge environments, digital twins, and other distributed platforms, allowing these principles to support trust establishment across increasingly heterogeneous administrative and technological environments.

7.4 TRUSTWORTHY SECURITY AT LARGER SCALE AND IN DIVERSE ENVIRONMENTS

The distributed and modular character of the Security Trust Function makes the architecture suitable for exploration across a broader range of deployment conditions. The mechanisms developed for adaptive security, decentralised identity, credential-status management, trust anchoring, and distributed auditing can be applied to scenarios involving increasing numbers of users, services, edge nodes, administrative domains, and concurrent interactions. Extending the architecture to such environments would broaden the applicability of the developed mechanisms across a wider range of operational characteristics, including latency, throughput, resource utilisation, and workload diversity. Of particular interest is the behaviour of the operational security flavour model as network and service conditions become increasingly dynamic. This extends the architecture beyond the project-specific scenarios towards a wider class of distributed, resource-aware, and trust-sensitive environments.

7.5 REUSE AND EVOLUTION OF SAFE-6G TECHNOLOGY ASSETS

Beyond the integrated Security Trust Function, the modular organisation of the implementation creates an additional pathway centred on the reuse and evolution of individual technological assets. The architecture has been structured around components with clearly defined responsibilities, allowing them to contribute either as part of the complete trust-management framework or as independent building blocks in other research architectures. This flexibility enables two complementary forms of technological continuation. The complete Security Trust Function can be incorporated as an integrated baseline where adaptive trust management is required, while selected components can be integrated independently into architectures addressing more specialised security objectives. Consequently, the resulting technological assets can support subsequent research and innovation activities without constraining their reuse to the original project architecture or use cases. New functionality, deployment contexts, integration models, and technology combinations can therefore be developed on top of these assets, building on the implementation and integration experience already acquired within the project.

8 CONCLUSIONS

The User-centric Security Trust Function provides a key building block of the SAFE-6G trustworthiness framework. Its role is to support adaptive security by combining trust flavours, user intent, identity evidence, Verifiable Credentials, Decentralised Identifiers, access policies, and auditability mechanisms across the computing continuum.

The SSI contribution addresses the decentralised identity dimension of the Security Trust Function. It provides mechanisms for credential issuance, credential verification, OIDC integration, Webhook integration, DID management, did:safe6g support, and credential lifecycle management. These elements enable user-centric, verifiable, and privacy-preserving authentication and authorisation within SAFE-6G metaverse use cases and security workflows, without requiring all applications to adopt the same integration pattern.

The current design establishes clear responsibilities across the three cooperating layers. The Local AI Agent recommends the appropriate operational security flavour based on trust requirements and real-time resource conditions, and actuates the flavour calibrated by CoCo. The vApp layer implements SSI-based authentication and authorisation flows through the Credential Issuer, Credential Verifier, and SSI Vault. The nApp layer provides network-facing enforcement and mediation around the Verifiable Data Registry and the audit-enabled blockchain infrastructure. Remaining work primarily concerns end-to-end integration with the Cognitive Coordinator and sibling Trust Functions, performance and scalability validation under realistic 6G workloads, and fine-tuning of the thresholds that map trust levels to security flavours and enforcement actions, based on the WP5 measurements. These activities are planned for subsequent project phases and will be reported in the corresponding deliverables.

9 REFERENCES

- [1] SAFE-6G Consortium, "D4.1: Cognitive Coordination, AI Agents, and User-centric Trust Functions," SAFE-6G Project Deliverable.
- [2] SAFE-6G Consortium, "D2.1: Definition of Technical Requirements for User-centric 6G Trustworthiness," SAFE-6G Project Deliverable.
- [3] SAFE-6G Consortium, "D2.2: Overall SAFE-6G Framework and Reference Architecture Design," SAFE-6G Project Deliverable.
- [4] SAFE-6G Consortium, "D2.3: Metaverse Use-cases Definition with Virtual Assistant for User-centric Configuration," SAFE-6G Project Deliverable.
- [5] SAFE-6G Consortium, "D2.4: 6G Trustworthiness KPI & KVI Definition and Validation Methodology," SAFE-6G Project Deliverable.
- [6] W3C, "Decentralized Identifiers (DIDs) v1.0," W3C Recommendation, Jul. 2022. [Online]. Available: <https://www.w3.org/TR/did-core/>
- [7] W3C, "Verifiable Credentials Data Model v2.0," W3C Recommendation, May 2025. [Online]. Available: <https://www.w3.org/TR/vc-data-model-2.0/>
- [8] W3C, "Bitstring Status List v1.0," W3C Recommendation, May 2025. [Online]. Available: <https://www.w3.org/TR/vc-bitstring-status-list/>
- [9] OpenID Foundation, "OpenID for Verifiable Credential Issuance 1.0," Final Specification, Sep. 2025. [Online]. Available: https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html
- [10] OpenID Foundation, "OpenID for Verifiable Presentations 1.0," Final Specification, Jul. 2025. [Online]. Available: https://openid.net/specs/openid-4-verifiable-presentations-1_0.html
- [11] OpenID Foundation, "Self-Issued OpenID Provider v2 (SIOPv2)," Draft 13, Nov. 2023. [Online]. Available: https://openid.net/specs/openid-connect-self-issued-v2-1_0.html
- [12] OpenID Foundation, "OpenID Connect Core 1.0," Nov. 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html
- [13] Hyperledger Foundation, "Hyperledger Fabric Documentation," 2024. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/>
- [14] IETF, "Token Status List (TSL)," Internet-Draft, draft-ietf-oauth-status-list-21, Jun. 2026 (approved for publication as an RFC). [Online]. Available: <https://datatracker.ietf.org/doc/draft-ietf-oauth-status-list/>
- [15] NIST, "SP 800-207: Zero Trust Architecture," Aug. 2020. [Online]. Available: <https://csrc.nist.gov/publications/detail/sp/800-207/final>
- [16] European Parliament, "Regulation (EU) 2024/1183 amending Regulation (EU) No 910/2014 (eIDAS 2.0)," Apr. 2024. [Online]. Available: <https://eur-lex.europa.eu/eli/reg/2024/1183>
- [17] European Commission, "The European Digital Identity Wallet Architecture and Reference Framework (ARF)," v3.0.0, Jul. 2026. [Online]. Available: <https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework>
- [18] NIST, "Post-Quantum Cryptography Standardisation," FIPS 203 (ML-KEM), FIPS 204 (ML-DSA), and FIPS 205 (SLH-DSA), Aug. 2024; FIPS 206 (FN-DSA) in development. [Online]. Available: <https://csrc.nist.gov/projects/post-quantum-cryptography>



- [19] T. Ioannidis, "Security Function," InQbit Romania, 2026. [Online]. Available: <https://gitlab.com/safe-6g/development/security-function/>

ANNEX 1: TECHNICAL SPECIFICATIONS

This annex complements the architectural description of Chapter 5 with the technical specifications of the main ledger-facing artefacts of the Security Trust Function: the did:safe6g DID method (A1.1), the SecurityAudits record model and its flavour-dependent write sequence (A1.2), and reference examples of the credential formats issued by the vApp layer (A1.3).

A1.1 THE DID:SAFE6G DID METHOD

Syntax. A did:safe6g identifier has the form did:safe6g:<method-specific-id>, where the method-specific identifier is a randomly generated UUID (version 4) assigned at creation time, for example did:safe6g:5f8d04e2-7b19-4a92-9c3a-d6023f462194. Identifiers are opaque: they carry no personal data and cannot be correlated with the attributes of their subject.

Operations. All operations are executed against the VerifiableDataRegistry chaincode on Hyperledger Fabric. They are submitted through the Fabric Gateway and mediated by the VDR-Invoker Policy Enforcement Point, which applies the audit policy of the active flavour (A1.2). The did:safe6g method is enabled under the mid and high flavours; under the low flavour, the vApp layer relies on locally resolvable methods such as did:key.

Operation	Chaincode function	Behaviour
Create	createDID	Validates the DID syntax and the DID Document structure, rejects duplicate identifiers, and stores the document in the ledger world state.
Read (resolve)	readDID	Returns the authoritative DID Document for the given method and identifier. Executed as a query, without ledger writes.
Update	updateDID	Replaces the DID Document, for example to rotate keys or to add a post-quantum verification method. Previous versions remain in the ledger history.
Deactivate	deleteDID	Removes the DID Document from the world state so that the identifier no longer resolves, while the immutable ledger history retains prior versions for audit purposes.

Table 4: did:safe6g Method Operations

DID Document. Verification methods are encoded with publicKeyBase58. Under the mid flavour, a single Ed25519 key is registered for authentication and assertion. Under the high flavour, an additional ML-DSA-65 key is registered under assertionMethod. The ML-DSA verification method type is project-defined, since no post-quantum verification method type has yet been registered in the W3C DID specification registries. Table 5 shows a DID Document created under the high flavour.

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083",
  "controller": "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083",
  "verificationMethod": [
    {
      "id": "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083#key-1",
      "type": "Ed25519VerificationKey2018",

```

```

    "controller": "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083",
    "publicKeyBase58": "B12NYF8RrR3h41TDCTJoY59usg3mbtbnFs7Eud1Y6u"
  },
  {
    "id": "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083#key-2",
    "type": "ML-DSA-65-VerificationKey2024",
    "controller": "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083",
    "publicKeyBase58": "3yQsZ8...(1,952-byte key, truncated)...Lk9P"
  }
],
"authentication": [
  "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083#key-1"
],
"assertionMethod": [
  "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083#key-1",
  "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083#key-2"
]
}

```

Table 5: did:safe6g DID Document Example (High Flavour)

A1.2 LEDGER AUDIT MODEL

Every ledger interaction mediated by the VDR-Invoker under the mid and high flavours produces an audit record in the SecurityAudits chaincode. Each record is keyed by a UUID and has the structure summarised in Table 6.

Field	Description
id	Unique identifier (UUID) of the audit record.
issuer	Component on whose behalf the audited operation is executed (e.g., the SSI Agent).
auditor	Component producing the record (the VDR-Invoker PEP).
scope	Target of the operation, expressed as channel/chaincode (e.g., security/VerifiableDataRegistry).
auditType	Operation type: query or invoke.
riskLevel	Risk classification (low, medium, high, or critical). Policy violations intercepted by the PEP are recorded as critical.
status	Lifecycle state: created, completed, or failed.
result	Outcome of the audited operation: pass flag, score, vulnerability count, and details.
additionalDetails	Operation context, such as the invoked method, its arguments, and a description.
recommendations, evidence	Optional lists of recommendations and evidence references.
timestamps	Ordered list of status transitions with ISO 8601 timestamps.

Table 6: SecurityAudits Record Structure

The chaincode exposes the createAudit, readAudit, updateAudit, and deleteAudit operations. The order in which the VDR-Invoker invokes them depends on the active flavour, as summarised in Table 7.

Flavour	Before the registry transaction	After the registry transaction	Failure handling
Low	No audit	No audit	Not applicable: requests are forwarded directly to the VerifiableDataRegistry.

Mid	No audit	createAudit with the final status (completed or failed) and result	Fail-open: an audit write failure raises a warning and does not affect the client transaction.
High	createAudit with status created	updateAudit with the final status, result, and timestamps	Fail-closed: if the pre-execution audit fails, the transaction is aborted and access is denied.

Table 7: Flavour-Dependent Audit Sequence

Table 8 shows an audit record for a DID resolution executed under the high flavour, after the post-execution update.

```
{
  "id": "0b7e3c52-9a41-4f7e-b1d2-3f6a8c9e2d10",
  "issuer": "ssi-agent",
  "auditor": "vdr-invoker-pep",
  "scope": "security/VerifiableDataRegistry",
  "auditType": "query",
  "riskLevel": "medium",
  "status": "completed",
  "result": {
    "pass": true,
    "score": 100,
    "vulnerabilities": 0,
    "details": "Transaction completed successfully"
  },
  "additionalDetails": {
    "method": "readDID",
    "args": [
      "safe6g",
      "5f8d04e2-7b19-4a92-9c3a-d6023f462194"
    ],
    "description": "VDR DID operation: readDID for identifier: safe6g"
  },
  "timestamps": [
    {
      "status": "created",
      "date": "2026-03-15T08:00:15.112Z"
    },
    {
      "status": "completed",
      "date": "2026-03-15T08:00:15.348Z"
    }
  ]
}
```

Table 8: SecurityAudits Record Example

A1.3 CREDENTIAL FORMAT EXAMPLES

The following examples show the decoded payloads of the JWT-secured credentials issued by the vApp layer. Because the signature is carried by the JWT envelope, whose header references the issuer key (for example `did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083#key-1`), the payloads contain no embedded proof object. Both payloads combine the VCDM 1.1 base context expected by JWT-based verifiers with the VCDM 2.0 context that defines the Bitstring Status List terms. Table 9 shows the status list credential referenced in Section 5.2.3, and Table 10 shows the MetaverseCredential described in Section 5.2.4.

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/ns/credentials/v2"
  ],
  "id": "https://issuer.safe6g.eu/status/list/revocation-list-1",
  "type": [
    "VerifiableCredential",
    "BitstringStatusListCredential"
  ],
  "issuer": "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083",
  "issuanceDate": "2026-03-15T07:55:00Z",
  "credentialSubject": {
    "id": "https://issuer.safe6g.eu/status/list/revocation-list-1#list",
    "type": "BitstringStatusList",
    "statusPurpose": "revocation",
    "encodedList": "H4sIAAAAAAAAA-3BMQEAAADCoPVPbQwfoAAAAAAAAAAAAAAAAAIC3AYbSVKsAQAAA"
  }
}
```

Table 9: Bitstring Status List Credential JSON Example

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/ns/credentials/v2"
  ],
  "id": "urn:uuid:7c9e6679-7425-40de-944b-e07fc1f90ae7",
  "type": [
    "VerifiableCredential",
    "MetaverseCredential"
  ],
  "issuer": "did:safe6g:6ed04e2a-6d11-4a71-9c4b-d6023f351083",
  "issuanceDate": "2026-03-15T08:00:00Z",
  "expirationDate": "2027-03-15T08:00:00Z",
  "credentialSubject": {
    "id": "did:safe6g:5f8d04e2-7b19-4a92-9c3a-d6023f462194",
    "operatorId": "OP-98214",
    "displayName": "Industrial Systems Specialist",
    "primaryRole": "technician",
    "authorizedRoles": [
      "technician",
      "factory-digital-twin"
    ],
    "clearanceLevel": "level-3-operational",
    "permittedZones": [
      "Production-Bay-A",
      "Robotics-Cell-4",
      "Virtual-Twin-East"
    ]
  },
  "credentialStatus": {
    "id": "https://issuer.safe6g.eu/status/list/revocation-list-1#1042",
    "type": "BitstringStatusListEntry",
    "statusPurpose": "revocation",
    "statusListIndex": "1042",
    "statusListCredential": "https://issuer.safe6g.eu/status/list/revocation-list-1"
  }
}
```

Table 10: MetaverseCredential JSON Example