A Formal Cryptanalysis of the Universal Digital Identity Framework (UDIF)

John G. Underhill

Quantum Resistant Cryptographic Solutions Corporation

Abstract. The Universal Digital Identity Framework (UDIF) is a post–quantum identity and object–custody architecture built on hierarchical certificates, capability–bounded authorization, canonical TLV state containers, Merkle–committed registries, anchored audit logs, and a PQ–authenticated transport. This paper presents a formal cryptanalysis of UDIF under standard game–based definitions and the cryptographic assumptions underlying its constituent primitives, including ML–DSA or SLH–DSA signatures, ML–KEM or McEliece key encapsulation, SHA3–family hashes, KMAC–based MACs, and RCS–based AEAD. We formalize the UDIF system model, including identity chains, capability tokens, registry and object commitments, anchor–chain propagation, transport handshakes, epoch–ratcheting, and the predicate–based query semantics that enforce minimal disclosure across domains. For each subsystem we provide precise security definitions covering certificate–chain authenticity, capability soundness, registry and object integrity, anchor–chain consistency, session confidentiality and integrity, forward secrecy of branch trunks, minimal–disclosure query privacy, and treaty–scoped cross–domain containment. Under the stated assumptions, we prove that any adversary forging certificates or escalating capabilities reduces to breaking the underlying signature or MAC; any attempt to equivocate object or registry state, or to fork anchor histories, reduces to finding hash collisions or forging signatures; any successful attack on the transport channel reduces to breaking the KEM, AEAD, or authenticated–header construction; and any adversary exceeding authorized query visibility contradicts Merkle commitment binding or capability unforgeability. The combined results yield explicit adversarial advantage bounds showing that UDIF achieves authenticity, integrity, confidentiality, forward secrecy, auditability, and minimal disclosure within negligible advantage relative to the primitive assumptions. This establishes UDIF as a cryptographically rigorous framework for post–quantum identity, authorization, and cross–domain provenance.

1 Introduction

The Universal Digital Identity Framework (UDIF) is a post–quantum identity and provenance architecture designed to provide a unified, cryptographically sound substrate for representing entities, administering authority, managing object custody, and enforcing auditability across diverse domains. UDIF addresses a long–standing gap in digital infrastructure: the absence of a universal, tamper–evident, post–quantum secure foundation for identity binding, ownership tracking, cross–domain verification, and minimal–disclosure queries. Existing identity and object registries, whether governmental, financial, enterprise, or supply–chain, typically rely on siloed databases, classical cryptography, and non–portable semantics. UDIF replaces these ad hoc structures with a rigorously specified system built on hierarchical certificates, deterministic canonical encodings, Merkle–committed state containers, anchored audit logs, and a PQ–secure transport layer.

At a structural level, UDIF organizes participants into a rooted hierarchy of authorities. A Root Authority defines the cryptographic suite of the domain and issues certificates to Branch Controllers. Branches operate either in branch–administration mode, where they manage subordinate branches, or in group–administration mode, where they act as Group Controllers responsible for User Agents. UAs are the leaf nodes of the hierarchy and the sole owners of objects. Every entity is represented by a canonical TLV–encoded certificate signed by its parent, forming a verifiable chain to the Root. Capabilities and access masks constrain each node's authority by binding explicit permissions into certificate and token fields. Objects, registries, membership logs, and transaction logs are maintained as Merkle–committed containers, with periodic Anchor Records relayed upstream to provide continuous, tamper–evident auditability. Communication between any UDIF nodes occurs over TCP tunnels protected by a post–quantum authenticated handshake, a cSHAKE–based key schedule, RCS–based AEAD with authenticated headers, strict sequence and time controls, and an epoch–based asymmetric ratchet for long–lived branch trunks. Query operations follow a predicate–based minimal–disclosure model, where entities may learn only Boolean answers or capability–restricted proofs, and cross–domain requests are governed by bilateral treaties.

The goal of this paper is to give a formal cryptanalysis of UDIF under standard game–based definitions and the security properties induced by its construction. We present a complete functional and adversarial model of UDIF, formalize its certificate and capability system, canonical data model, registry and object commitments, anchored audit chains, transport handshake and ratchet, and predicate–based query semantics. For each subsystem we define the precise security goals, including certificate–chain authenticity, capability soundness and non–escalation, registry and object integrity, anchor–chain consistency, transport confidentiality and integrity, forward secrecy and post–compromise recovery, minimal–disclosure privacy, and cross–domain containment. We prove that these properties hold up to negligible advantage under the cryptographic assumptions underpinning the UDIF primitives, including ML–DSA or SLH–DSA signatures, ML–KEM or McEliece encapsulation, RCS or AES–GCM authenticated encryption, SHA3–family hashing, KMAC authentication, and the indifferentiability of sponge functions.

UDIF builds on design principles shared with other QRCS protocols, including QSTP, QSMP, MPDC, RCS, and DKTP, while extending them into the identity and object–provenance domain. Like QSTP, UDIF uses a PQ–secure handshake and AEAD channel with authenticated headers, strict sequencing, and epoch ratcheting. Like MPDC, it applies canonical TLV encodings, deterministic Merkle commitments, and capability masks for authorization. Like QSMP, it uses a post–quantum KEM and signature suite fixed at compile time to avoid downgrade and parser ambiguity. UDIF's contribution is to unify these cryptographic components into a coherent identity and auditability framework suitable for multi–domain deployments.

The remainder of this paper proceeds as follows. Section 2 fixes notation and cryptographic

preliminaries. Section 3 gives a complete formalization of the UDIF system model, including entities, certificates, capabilities, registries, logs, anchors, transport, and query semantics. Section 4 defines the security properties targeted by UDIF. Section 5 states the cryptographic assumptions used in the analysis. Sections 6 through 8 provide security proofs for certificates and capabilities, registry and anchor integrity, the transport layer, and the query and treaty model. Section 9 gives the composition theorem establishing UDIF's end–to–end guarantees. Section 10 discusses adversarial work factors and recommended parameter sets. Section 11 analyzes limitations and implementation considerations. Section 12 concludes.

1.1 Background and Motivation

Modern digital infrastructure lacks a universal, cryptographically enforced identity substrate capable of representing individuals, institutions, object registries, and asset provenance with strong authenticity, minimal disclosure, and tamper–evident auditability. Identity systems such as SSNs, KYC registries, directory services, and certificate–based authentication are fragmented, centralized, or limited in scope, and they lack cryptographic binding between entities and the objects they own or manage. Provenance systems in finance, supply chains, and digital assets rely on mutable databases or ledger–like constructs that expose too much information or lack formal guarantees. As digital interactions increasingly span jurisdictions and independent administrative domains, a need arises for a verifiable identity and custody system that preserves privacy, enforces least privilege, and remains secure against classical and quantum adversaries.

UDIF addresses this need by defining a polymorphic identity and object container model equipped with post–quantum certificates, capability–scoped authorization, canonical encodings, Merkle–committed registries, anchored audit propagation, and a PQ–secure transport. The system is designed for deployment in adversarial and cross–domain environments where passive observation, active interference, compromised nodes, treaty misuse, and cryptanalytic threats must all be assumed.

1.2 Overview of the UDIF Construction

UDIF organizes participants into a hierarchical trust model rooted at a single Root Authority. Every entity holds a certificate containing a suite identifier, role, serial number, issuer serial, validity window, public key, capability and access mask fields, and a parent's signature. Capabilities are implemented as KMAC–tagged bitmaps that constrain allowable verbs and scopes. All structures in UDIF, including certificates, objects, registries, logs, anchors, queries, and treaties, are encoded canonically using TLV encodings with strict ordering and deterministic minimal–length encodings.

Objects are represented as canonical records containing serials, type codes, creator bindings, attribute Merkle roots, ownership assignments, and signatures. Each UA maintains a registry of object digests organized into a Merkle tree, with the registry root periodically committed to membership logs. GCs and BCs maintain membership and transaction logs, which are folded into periodic Anchor Records containing Merkle roots and counters. Anchors propagate upward through the hierarchy, forming a tamper–evident audit chain to the Root.

Sessions between UDIF nodes run over TCP tunnels protected by a PQ–authenticated handshake based on KEM encapsulation, digital signatures, nonces, and a transcript hash. Keys are derived using cSHAKE and used to drive an RCS–based AEAD with authenticated headers. Strictly monotonic sequence numbers, timestamp windows, and epoch counters enforce replay and reordering resistance, while long–term branch trunks employ periodic asymmetric ratchets to provide forward secrecy and post–compromise recovery.

Queries operate under a minimal–disclosure predicate model. Predicate families include existence checks, ownership binding, attribute–bucket membership, and registry membership proofs. Queries require explicit capabilities and are logged and anchored. Cross–domain queries flow through bilateral treaties that constrain which predicate families may be forwarded and audited in both domains.

1.3 Security Objectives of UDIF

UDIF targets the following informal security objectives:

- **Authenticity of certificates and identities**: Only valid parent–signed certificates may appear in the hierarchy, and no adversary can forge or elevate authority.
- **Capability soundness and non–escalation**: An entity's permissions cannot exceed those granted by its parent or authorized capability tokens.
- **Object and registry integrity**: Objects, registry states, and ownership histories cannot be equivocated or rolled back without detection.
- **Anchor–chain consistency and auditability**: Logs and anchor histories cannot be forked or manipulated without breaking cryptographic assumptions.
- **Transport confidentiality and integrity**: All messages exchanged between UDIF nodes remain confidential and tamper–evident under active attacks.
- **Forward secrecy and post–compromise recovery**: Long–lived branch trunks gain new entropy through asymmetric ratchets to limit compromise impact across epochs.
- **Minimal disclosure**: Adversaries learn only predicate outcomes authorized by capabilities and cannot infer hidden state or attributes.
- **Cross–domain containment**: Treaty–mediated interactions reveal nothing beyond the explicitly permitted predicate families.

These objectives motivate the formal definitions and proofs presented in the remainder of the paper.

1.4 Roadmap

The remainder of this paper is structured to isolate the cryptographic components of UDIF, define their security properties, and prove them under standard assumptions. Section 2 introduces notation, formal models, and the interfaces for the primitives that UDIF relies upon. Section 3 presents a complete formal system model of UDIF, including certificates, capabilities, registries, logs, anchor propagation, the transport channel, and query semantics. Section 4 defines the security goals for each subsystem. Section 5 outlines the hardness assumptions and models used in the proofs. Sections 6, 7, and 8 contain the main technical results, giving security proofs for the certificate and capability system, the registry and anchor architecture, the transport layer, and the query and treaty model. Section 9 demonstrates how these component results compose into end to end guarantees. Section 10 discusses parameter choices and adversarial work factors. Section 11 presents limitations and implementation considerations. Section 12 concludes.

2 Preliminaries and Notation

This section introduces notation, probability models, and cryptographic interfaces used throughout the analysis. All security statements are with respect to a security parameter $\lambda \in \mathbb{N}$ and adversaries with probabilistic polynomial time complexity unless otherwise stated.

2.1 Basic Notation

Let $\{0,1\}^n$ denote the set of all bit strings of length n. Let $\{0,1\}^*$ denote the set of all finite bit strings. For a string x, the notation $|x|$ denotes its bit length. For bit strings x and y, the symbol $x\|y$ denotes concatenation. For a probabilistic algorithm A, the notation $y \leftarrow A(x)$ denotes running A on input x and obtaining an output sampled from its internal randomness.
For a set S, the notation $x \xleftarrow{\$} S$ denotes sampling x uniformly at random from S. A function $\epsilon(\lambda)$ is negligible if for every polynomial $p(\lambda)$ there exists λ_0 such that $\epsilon(\lambda) < 1/p(\lambda)$ for all $\lambda > \lambda_0$.
For an adversary $\mathcal{A}$ and experiment Exp, the advantage of $\mathcal{A}$ is written

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Exp}}(\lambda)$$

which denotes the probability that $\mathcal{A}$ wins the experiment, minus the probability that a trivial random guess would succeed where applicable.
We use $\mathsf{poly}(\lambda)$ to denote any function bounded by a polynomial in λ. All algorithms and adversaries are assumed to run in time $\mathsf{poly}(\lambda)$.

2.2 Cryptographic Primitives

This subsection defines the abstract interfaces and security notions for the primitives used in UDIF. The proofs refer only to these interfaces and not to specific implementations.

Key Encapsulation Mechanism (KEM). A KEM consists of algorithms

$$(\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$$

where KeyGen outputs a public key and a secret key, Encaps outputs a ciphertext and a shared secret under the public key, and Decaps recovers the shared secret from the ciphertext under the secret key. The KEM is IND–CCA secure if no probabilistic polynomial time adversary with decapsulation oracle access can distinguish the real shared secret from a random value.

Digital Signature Scheme. A signature scheme consists of

$$(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$$

where Verify outputs accept or reject. The scheme is EUF–CMA secure if no adversary with adaptive signing oracle access can forge a valid signature on a new message.

Authenticated Encryption with Associated Data (AEAD). An AEAD interface provides

$$\mathsf{Enc}(k, n, a, m) \quad \text{and} \quad \mathsf{Dec}(k, n, a, c)$$

where k is a key, n is a nonce, a is associated data, m is plaintext, and c is ciphertext. The AEAD must be IND–CPA secure for confidentiality and INT–CTXT secure for integrity. Associated data must be authenticated and bound into the ciphertext.

Hash and XOF Functions. UDIF uses the SHA3 family, including SHA3, SHAKE, cSHAKE, and KMAC. These are modeled either in the standard model or as ideal permutations where appropriate. We assume collision resistance, preimage resistance, and indifferentiability properties where required. For a hash function H, the output is written

$$H(x) \in \{0,1\}^n$$

for a fixed output length n determined by the construction. For XOFs, outputs are variable length.

2.3 Canonical TLV Encoding

UDIF uses canonical Tag–Length–Value encoding with uvarint tags and lengths. We formalize this as a deterministic encoding function

$$\mathsf{C14N} : \mathcal{R} \to \{0,1\}^*$$

mapping abstract records to their canonical byte sequences. The following constraints define the canonical form.

- Tags must appear strictly in ascending numerical order.
- Each tag may appear at most once unless designated as a repeated field.
- Length values must be encoded in minimal uvarint form.
- Fixed length fields must appear as raw bytes.
- UTF–8 strings must be normalized to NFKC prior to encoding.
- Nested TLVs must themselves be canonical under this definition.

We assume that $\mathsf{C14N}$ is injective on all valid UDIF records. Any deviation from canonical form results in an invalid encoding. For Merkle commitments, the digest of a record R is defined as

$$\mathsf{Digest}(R) = H(\mathsf{label} \| \mathsf{C14N}(R))$$

where the label ensures domain separation. The collision resistance of the Merkle trees used for registries, logs, and anchors follows directly from the collision resistance of the hash function combined with injectivity of $\mathsf{C14N}$.

2.4 Adversarial Model and Oracles

We analyze UDIF in a multi party setting with adaptive adversaries capable of interacting with UDIF nodes at all layers. The adversary controls the network and schedules all message delivery. The adversary may corrupt nodes, request signatures, decapsulations, and AEAD encryptions or decryptions under controlled conditions, and submit arbitrary queries to the UDIF system according to the capabilities it has obtained.
The environment interacts with UDIF through oracles representing the operations available to entities in the system. These include:

- Certificate issuance oracles for parent nodes.
- Capability issuance and revocation oracles.
- Object creation, update, and transfer oracles.
- Registry and log completion oracles that return state commitments.

- Transport session oracles that simulate the authenticated handshake and record exchange.
- Query oracles for existence, ownership binding, attribute bucket membership, and registry membership.
- Treaty forwarding oracles for cross domain predicate evaluation.

The adversary may adaptively choose its queries and may attempt to cause inconsistencies, equivocations, replay attacks, or unauthorized escalations. The security games defined in later sections capture the limits of what the adversary can learn or influence without breaking the underlying cryptographic assumptions.

3 UDIF System Model

This section provides a complete functional model of UDIF. All security definitions and proofs in later sections are with respect to this model. The model captures the entities, certificates, capabilities, registries, logs, anchor propagation, transport channels, and query mechanics described in the UDIF specification.

3.1 Entities and Roles

Let $\mathcal{R}$ denote the set of Root Authorities. A Root is a unique element per domain and defines the cryptographic suite. Let $\mathcal{B}$ denote the set of Branch Controllers. Each Branch $B \in \mathcal{B}$ operates in exactly one of two modes. In branch administration mode, B manages subordinate branches. In group administration mode, B serves as a Group Controller. Let $\mathcal{G} \subseteq \mathcal{B}$ denote the set of all Branches in group administration mode, which we refer to as GCs. Let $\mathcal{U}$ denote the set of User Agents. UAs are leaf entities and are the only holders of objects. Let $\mathcal{O}$ denote the set of object records.
We model Roots, BCs, and GCs as long lived servers that maintain state across sessions. UAs are modeled as clients that initiate authenticated transport sessions with their respective GC. BC to BC connections are long term trunks maintained continuously in the hierarchy. All entities have persistent key material and certificate chains.

3.2 Certificates and Identity Chains

Each entity $E \in \mathcal{R} \cup \mathcal{B} \cup \mathcal{U} \cup \mathcal{O}$ holds a certificate

$$\mathsf{Cert}_E = (s, r, \sigma_E, \sigma_{\mathsf{iss}}, t_{\mathsf{from}}, t_{\mathsf{to}}, pk, e, b, \sigma)$$

encoded in canonical TLV form. Here s is the suite identifier. The field r specifies the entity role. The field σ_E is a unique serial. The field σ_{iss} is the issuer serial, omitted in Root certificates. The fields $(t_{\mathsf{from}}, t_{\mathsf{to}})$ define the validity window. The field pk is the public verification key. The field e is the policy epoch. The field b is the capability bitmap embedded directly into the certificate. The final field σ is the parent signature over the TLV encoding of the certificate fields except the signature itself.
The parent child relation is defined by the issuer serial. A certificate chain for an entity E is the ordered sequence

$$\mathsf{Chain}(E) = (\mathsf{Cert}_E, \mathsf{Cert}_{P_1}, \ldots, \mathsf{Cert}_{P_k})$$

where P_1 is the issuer of E and P_k is a Root. Verification of a chain requires signature verification on each link and compliance of roles and capabilities with the UDIF hierarchy rules.

3.3 Capabilities and Access Masks

Capabilities are delegated permission tokens issued by parent authorities. A capability token has the form

$$\mathsf{Cap} = (v, q, \sigma_{\mathsf{to}}, \sigma_{\mathsf{iss}}, t, d, \tau)$$

where v is the verbs bitmap, q is the scope bitmap, σ_{to} and σ_{iss} identify the subject and issuer, t is the expiration time, and d is the domain separated digest

$$d = H(\mathsf{label} \| \mathsf{C14N}(v, q, \sigma_{\mathsf{to}}, \sigma_{\mathsf{iss}}, t))$$

The field τ is a KMAC tag computed under the issuer key. We assume that capabilities cannot be forged without breaking collision resistance or the KMAC construction.
Access masks are embedded in certificates and constrain attribute level visibility. An entity may act only if both its certificate mask and its valid capability set permit the requested predicate. The effective permissions of entity E at time t are the intersection of the certificate bitmap and all unexpired capabilities issued to E.

3.4 Objects and Registries

Objects are immutable identity containers with mutable state fields. An object record is a TLV encoded structure

$$\mathsf{Obj} = (\sigma_O, \tau, \sigma_C, h_A, \sigma_U, t_c, t_u, \sigma)$$

where σ_O is a 32 byte identifying serial. The field τ is a type code. The field σ_C is the creator certificate serial. The field h_A is the Merkle root of committed attributes. The field σ_U is the current owner certificate serial. The fields t_c and t_u are creation and update times. The field σ is the signature of the current owner over the TLV content.
Each UA maintains a registry which is a Merkle tree over object digests. A registry leaf encodes

$$(h_O, h_U, f, t)$$

where h_O is the object digest, h_U is the owner certificate digest, f is a flags field, and t is a timestamp. Leaves are lexicographically sorted by h_O. The registry root is

$$h_{\mathsf{reg}} = H(\mathsf{label} \| \mathsf{C14N}(\mathsf{sorted\ leaves}))$$

Registries are committed by GCs during membership log updates and included in Anchor Records.

3.5 Logs and Anchor Records

Each GC or BC maintains two append only logs. The membership log records UA and branch lifecycle events such as enrollment, suspension, revocation, capability grants, and registry commits. The transaction log records object creations, transfers, attribute updates, and status changes.
Each log entry is a TLV encoded event with fields for an event code, subject serial, time, optional data, and a signature. Log entries are hashed into Merkle trees. Let h_{mem} and h_{tx} denote the Merkle roots for the membership and transaction logs over a given interval.
An Anchor Record has the form

$$\mathsf{Anch} = (\sigma_B, q, t, h_{\mathsf{reg}}, h_{\mathsf{tx}}, h_{\mathsf{mem}}, c, \sigma)$$

where σ_B is the child serial, q is a sequence number, t is time, h_{reg}, h_{tx}, and h_{mem} are Merkle commitments, and c is an optional counter structure. The field σ is a signature by the child.
A parent maintains the last accepted anchor sequence for each child. Upon receiving an anchor, the parent verifies the signature, checks the sequence number, and appends the anchor to its log. By induction up the tree, all logs become committed at the Root.

3.6 Transport and Session Layer

Transport sessions run over TCP and use a mutual authentication handshake with KEM and signature primitives. Let (pk_A, sk_A) and (pk_B, sk_B) be long term keys of two peers. The handshake proceeds by authenticated exchange of nonces, certificates, transcript signatures, and KEM ciphertexts. The resulting shared secrets are passed through a cSHAKE based key schedule to derive a transmit key, a receive key, and nonces.
Each record transmitted over a session includes an authenticated header containing a flag field, a sequence number, a timestamp, an epoch counter, and a suite identifier. AEAD sealing authenticates the header as associated data. The receiver enforces a strict time window and sequence equality check. A branch trunk employs an asymmetric ratchet where each side periodically encapsulates fresh KEM material to the peer and derives new keys, advancing the epoch counter and resetting sequence numbers.
The transport state machine has the states idle, hello sent, hello received, established, ratcheting, and closed. Failures in certificate validation, replay detection, sequence checks, or AEAD verification result in immediate session closure and key erasure.

3.7 Query and Treaty Model

UDIF supports four predicate families. For each UA or object identifier x and predicate p, a query is a TLV encoded structure

$$Q = (\sigma_Q, \alpha, x, p, t, d)$$

where σ_Q is a query identifier, α is a type code, x identifies the subject, p expresses the predicate parameters, t is an optional time anchor, and d is the capability digest used to authorize the query.
The predicate families are:

- existence queries, which determine whether an entity exists under a controller,
- ownership binding queries, which test whether a UA is the current owner of an object,
- attribute bucket queries, which test whether an attribute lies within a predefined bucket,
- registry membership proofs, which produce yes or no results with optional Merkle paths.

A GC evaluates a query only if the subject holds a valid capability that authorizes the predicate type. All queries and responses are appended to the membership log.
Peering treaties define cross domain predicate visibility. A treaty is a bilateral agreement between two domains and carries a treaty identifier, the two branch serials, a scope bitmap specifying allowed predicate families, an expiration time, a policy epoch, and signatures from both domains. Queries permitted by the treaty may be forwarded across a dedicated transport session between the branches. Both sides log the forwarded query and its response. No predicates outside the treaty scope may be evaluated across domains.

4 Security Definitions

This section defines the formal security properties required of UDIF. Each definition is expressed as an experiment between a challenger and a probabilistic polynomial time adversary $\mathcal{A}$. The adversary is given oracle access matching the UDIF system model and may adaptively issue queries. All advantages are defined with respect to the security parameter λ.

4.1 Certificate Chain Authenticity

UDIF requires that no adversary can produce a certificate chain that verifies under the UDIF rules without being issued by the Root hierarchy. The adversary interacts with certificate issuance oracles that allow it to request certificates for entities it controls but does not allow impersonation of parents.

Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{cert}}$. The adversary may adaptively:

- request new child certificates from any parent it legitimately controls,
- query any public certificate verification procedure,
- request revocations or suspensions for certificates it controls.

Finally, $\mathcal{A}$ outputs a certificate chain Chain^* for some entity E^*. The adversary wins if:

$$\mathsf{VerifyChain}(\mathsf{Chain}^*) = 1 \quad \text{and} \quad \mathsf{Chain}^* \text{ contains a certificate not issued by a legitimate parent.}$$

Certificate Authenticity Advantage.

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cert}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathcal{A}}_{\mathsf{cert}}]$$

UDIF provides certificate chain authenticity if this advantage is negligible.

4.2 Capability Soundness and Delegation Safety

Capabilities must be unforgeable and must not allow escalation beyond permissions granted by parents.

Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{cap}}$. The adversary may:

- request capability tokens issued by parents it controls,
- request certificate issuance oracles for its own entities,
- query verification oracles for any capability digest or tag.

Finally, $\mathcal{A}$ outputs a capability token Cap^* for some subject E^*. The adversary wins if:

- the token verifies under UDIF rules, and
- the effective permissions derived from Cap^* exceed the permissions implied by any valid chain of parents.

Capability Soundness Advantage.

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cap}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathcal{A}}_{\mathsf{cap}}]$$

UDIF provides capability soundness if this advantage is negligible.

4.3 Object and Registry Integrity

Object integrity ensures that object states cannot be forked or equivocated. Registry integrity ensures that UA registries cannot be manipulated or contradicted without detection.

Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{obj-reg}}$**.** The adversary may:

- create objects under UAs it controls,
- request object updates and registry updates,
- request Merkle proofs and registry commitments.

Finally, $\mathcal{A}$ outputs two distinct encodings of either:

- an object record for the same serial that both verify under the UDIF rules, or
- a registry state and a Merkle proof that contradict an anchored registry root.

The adversary wins if the challenger verifies both views.

Object and Registry Advantage.

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{obj-reg}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathcal{A}}_{\mathsf{obj-reg}}]$$

UDIF provides object and registry integrity if this advantage is negligible.

4.4 Anchor Chain Soundness and Auditability

Anchor chain soundness guarantees that no adversary can generate two valid but inconsistent anchor histories for the same child or violate the monotonic sequence property.

Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{anch}}$**.** The adversary may:

- observe all anchors produced by honest nodes,
- request anchor verification oracles,
- schedule arbitrary network delivery patterns.

Finally, $\mathcal{A}$ outputs either:

- two valid anchor sequences for the same child with inconsistent Merkle roots, or
- a valid anchor with a non monotonic sequence number.

Anchor Soundness Advantage.

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{anch}}(\lambda) = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}^{\mathcal{A}}_{\mathsf{anch}}]$$

The anchor system is sound if this advantage is negligible.

4.5 Transport Confidentiality and Integrity

The transport channel must provide confidentiality, integrity, replay resistance, and authenticated header protection.

Confidentiality Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{conf}}$. The adversary may:

- initiate arbitrary sessions with honest nodes,
- observe all ciphertexts and headers,
- request encryptions and decryptions subject to standard IND–CPA restrictions,
- interfere with message ordering and timing.

The adversary chooses m_0 and m_1 of equal length. The challenger flips a bit b, encrypts m_b under an active UDIF session, and returns the ciphertext. The adversary outputs a guess b'. The advantage is

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{conf}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

Integrity Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{int}}$. The adversary may adaptively submit ciphertexts with arbitrary headers. It wins if any forged record is accepted by an honest node.
UDIF transport is secure if both advantages are negligible.

4.6 Forward Secrecy and Post Compromise Security

Long lived BC to BC trunks use an asymmetric ratchet that provides forward secrecy across epochs.

Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{fs}}$. The adversary may:

- compromise session state at some epoch i,
- observe all past and future ciphertexts,
- force ratchet initiations.

The challenger derives new keys at epoch $i + 1$. The adversary wins if it distinguishes any ciphertext encrypted under the new epoch keys or decrypts any past epoch ciphertexts.
Forward secrecy holds if this advantage is negligible.

4.7 Query Minimal Disclosure

A query reveals only Boolean predicate outcomes and optional proofs permitted by capabilities. Hidden state should remain indistinguishable.

Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{md}}$. The adversary chooses two UDIF worlds W_0 and W_1 that differ only in hidden attributes or registry entries. For all predicate types and all capabilities granted to $\mathcal{A}$, the predicate evaluations in both worlds must be identical. The challenger picks b, instantiates world W_b, and answers all authorized queries. The adversary outputs b'. The advantage is

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{md}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

Minimal disclosure holds if this advantage is negligible.

4.8 Cross Domain Containment

Cross domain queries must not reveal any information outside the treaty defined predicate scope.

Game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{cdc}}$. The adversary may:

- establish treaties between domains it controls,
- forward queries across domains,
- observe reply logs and anchor records,
- interact with honest GCs through forward and backward tunnels.

The adversary chooses two worlds that differ only in predicate responses not allowed by the treaty scope. The challenger flips a bit b, chooses one world, and answers all cross domain queries permitted by the treaty. The adversary outputs b'. The advantage is

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cdc}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

UDIF satisfies cross domain containment if this advantage is negligible.

5 Cryptographic Assumptions

The security of UDIF relies on the hardness of several well studied cryptographic problems and on the standard security definitions for the primitives used throughout the system. This section states these assumptions and the cryptographic models used in the proofs.

5.1 Primitive Security

The following assumptions are required for the reductions used in later sections.

IND CCA Security of the KEM. Let $\mathsf{KEM} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ be either ML KEM or Classic McEliece. The KEM provides IND CCA security if no probabilistic polynomial time adversary with access to a decapsulation oracle can distinguish the shared secret generated by Encaps from a uniform string of the same length except with negligible probability. Formally, for any adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) = \mathsf{negl}(\lambda).$$

EUF CMA Security of the Signature Scheme. Let $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be either ML DSA or SLH DSA. The scheme is existentially unforgeable under adaptive chosen message attack if no adversary with oracle access to Sign can produce a valid signature on any message that was not previously submitted. Formally,

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\mathcal{A}) = \mathsf{negl}(\lambda).$$

Security of the AEAD Construction. Let $\mathsf{AEAD} = (\mathsf{Enc}, \mathsf{Dec})$ be the authenticated encryption scheme used in UDIF, based on RCS and KMAC. The AEAD must satisfy the following properties:

- IND CPA confidentiality, meaning that ciphertexts reveal no information about plaintexts beyond their length,
- INT CTXT integrity, meaning that no forgery can be accepted by a decrypting party.

For any adversary $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AEAD}}(\mathcal{A}) = \mathsf{negl}(\lambda) \quad \text{and} \quad \mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{AEAD}}(\mathcal{A}) = \mathsf{negl}(\lambda).$$

Collision Resistance of SHA3 and Merkle Trees. For the hash function H used in UDIF (SHA3 256 or cSHAKE 256), no adversary can find distinct inputs x and y such that

$$H(x) = H(y)$$

except with negligible probability. Since registries, logs, and anchor structures use Merkle constructions with domain separated hashes, collision resistance of H implies collision resistance of all Merkle commitments.

Indifferentiability of Sponge Constructions. SHAKE, cSHAKE, and KMAC are modeled as indifferentiable from random oracles or ideal permutations when used in XOF or MAC mode. This supports the reductions used for the ratchet KDF, capability KMAC tags, and authenticated headers.

5.2 Modeling Assumptions

The analysis uses the following additional modeling assumptions.

Ideal Permutation Model for Keccak. When required, SHA3 based primitives are modeled using the ideal permutation model where the underlying Keccak permutation is treated as a uniformly random bijection. This assumption is standard for indifferentiability proofs of sponge based constructions.

Random Oracle Model for cSHAKE and KMAC. When used as key derivation functions or MAC generators, cSHAKE and KMAC are treated as random oracles or random oracle like objects. This is consistent with their security analyses and the domain separation rules in UDIF.

Deterministic and Injective Canonical Encoding. The canonical TLV encoding function `C14N` is assumed to be deterministic and injective over the domain of valid UDIF records. This prevents malleability attacks based on alternate encodings and is required for collision resistance of committed structures.

Merkle Tree Soundness. Merkle trees are assumed to be sound. A valid Merkle proof authenticates exactly one leaf and no adversary can construct inconsistent proofs for different leaves without finding a hash collision.

Deterministic Signature and Certificate Verification Model. Verification procedures for signatures, certificates, logs, and anchors are assumed to be deterministic functions of their inputs. No side channels or timing leakage are modeled.

6 Security of Certificates and Capabilities

This section proves that UDIF certificate chains cannot be forged and that capabilities cannot be constructed or escalated beyond what parent authorities authorize. The proofs rely on the EUF CMA security of the signature scheme and the unforgeability of KMAC tags.

6.1 Certificate Chain Authenticity Theorem

Theorem 1. *Let $\mathcal{A}$ be any probabilistic polynomial time adversary in the certificate authenticity game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{cert}}$. Suppose the signature scheme used for UDIF certificates is EUF CMA secure. Then any adversary that forges a valid UDIF certificate chain that was not issued by the Root hierarchy can be used to construct an adversary that breaks the EUF CMA security of the signature scheme. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cert}}(\lambda) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\lambda) + \mathsf{negl}(\lambda).$$

6.2 Proof of Chain Authenticity

Proof. The proof follows a hybrid argument. Let $\mathcal{A}$ be an adversary that wins the certificate authenticity game. We construct an adversary $\mathcal{B}$ that breaks the signature scheme.

Hybrid H0. This is the real certificate authenticity game. The challenger issues certificates by signing the TLV encodings using the real signing key. The adversary outputs a forged chain.

Hybrid H1. The challenger simulates all certificate issuance using the signature oracle of the EUF CMA challenger. Since signatures are always produced by the oracle on honestly formed messages, the view of $\mathcal{A}$ is identical to the real world.

Hybrid H2. When $\mathcal{A}$ outputs a forged certificate chain, at least one certificate in the chain must contain a signature that the challenger did not request from the signing oracle. Otherwise the certificate must originate from a legitimate parent and cannot violate the chain relation. Hence the bottom certificate contains a new signature on a message not requested previously.

Conclusion. The adversary $\mathcal{B}$ forwards this signature and the signed message to the EUF CMA challenger as a forgery. Since $\mathcal{B}$ perfectly simulates the environment, the success probability of $\mathcal{B}$ is identical to that of $\mathcal{A}$ up to negligible error. This concludes the reduction. □

6.3 Capability Unforgeability and Non Escalation

Theorem 2. *Let $\mathcal{A}$ be an adversary in the capability game $\mathsf{Game}^{\mathcal{A}}_{\mathsf{cap}}$. Suppose KMAC is a secure MAC under chosen message attack and the signature scheme is EUF CMA secure. Then any adversary that forges an effective capability not implied by parent permissions yields an adversary that either forges a KMAC tag or forges a signature. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cap}}(\lambda) \leq \mathsf{Adv}^{\mathsf{MAC\text{-}UF}}_{\mathsf{KMAC}}(\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. A forged capability must either contain:

- a valid KMAC tag under the issuer key that was never issued by the parent, or
- a reference to a certificate with an inflated capability bitmap not signed by the parent.

The first case gives a direct MAC forgery. We construct an adversary that uses the capability forger to produce a new digest and tag pair not requested during oracle queries. The second case implies that the adversary has produced a certificate whose embedded permission bits exceed those of its parent. This requires forging a signature on a certificate TLV that the legitimate parent never signed. Hence this gives a signature forgery.

In each case the reduction is straightforward because the canonical encoding of the capability or certificate is deterministic and injective. This allows extraction of the forged message on which the adversary succeeded. Therefore the advantage in forging or escalating capabilities is bounded by the sum of the advantages in forging KMAC tags or signatures, which are negligible. □

7 Security of Objects, Registries, and Anchors

This section analyzes the integrity guarantees provided by UDIF for object records, UA registries, and the anchored audit chain. Each result follows from the collision resistance of the hash function, the deterministic canonical encoding, the EUF CMA security of signatures, and the append only property of logs. Together these components ensure that no adversary can produce conflicting object states, equivocate registry roots, or create inconsistent anchor sequences without breaking one of the underlying assumptions.

7.1 Object Record Integrity

Theorem 3. *Let $\mathcal{A}$ be an adversary in the object integrity game. Suppose the hash function used for object digests and attribute commitments is collision resistant and the signature scheme is EUF CMA secure. Then any adversary that produces two distinct valid object records for the same serial, or produces a valid object record with an invalid creator or owner binding, can be used to either find a hash collision or forge a signature. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{obj}}(\lambda) \leq \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. An object record includes a serial, type code, creator digest, attribute Merkle root, current owner serial, timestamps, and a signature by the owner. The digest of the object is

$$h_O = H(\mathsf{label} \| \mathsf{C14N}(\mathsf{Obj})).$$

A successful forgery must satisfy one of the following:

1. Two different TLV encodings Obj and Obj' yield valid digests for the same serial. Since $\mathsf{C14N}$ is injective, this gives a collision in H.
2. A forged object record binds a creator or owner that never signed the record. This implies that the adversary has produced a valid signature on a message not signed by the legitimate owner, violating EUF CMA security.
3. A forged attribute Merkle root corresponds to a tree that does not match any attribute set produced by the legitimate creator. This again implies a hash collision in either the leaf hashes or internal nodes.

Since these are the only ways to construct a conflicting or invalid object record, the adversary does not win the game except with negligible probability. □

7.2 Registry Consistency and Non Repudiation

Theorem 4. *Let $\mathcal{A}$ be an adversary in the registry consistency game. Assume that the hash function is collision resistant and the membership logs are append only. Then any adversary that produces two distinct registry states for the same UA and same time, or contradicts an anchored registry root using a forged Merkle proof, can be used to find a hash collision or to violate log consistency. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{reg}}(\lambda) \leq \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. A UA registry is a Merkle set of object digests. The root is

$$h_{\mathsf{reg}} = H(\mathsf{label} \| \mathsf{C14N}(\mathsf{sorted\ leaves})).$$

The adversary may attempt the following attacks:

- Produce two Merkle trees with different leaves but identical root values. Since the canonical ordering of leaves is deterministic, this implies a collision in H.
- Produce a Merkle proof for an object that is not in the registry. A proof includes sibling hashes along a path. A false proof must either collide with an existing leaf or recreate an internal Merkle node without knowing the correct child. Either condition yields a collision in H.
- Contradict an anchored root by producing a registry state that does not match the root already sent to the GC and embedded in the latest Anchor Record. Since anchored states are signed and logs are append only, contradicting an anchor requires either forging a signature or generating inconsistent Merkle roots.

Thus any successful attack against registry integrity implies either a hash collision or a break of append only logs, both of which are ruled out by assumption. □

7.3 Anchor Chain Soundness

Theorem 5. *Let $\mathcal{A}$ be an adversary in the anchor soundness game. Assume the hash function is collision resistant and the signature scheme is EUF CMA secure. Then any adversary who constructs two valid but inconsistent anchor sequences for the same child, or any anchor that violates monotonic sequence properties, can be reduced to a collision attack or a signature forgery. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{anch}}(\lambda) \leq \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. An Anchor Record binds together the registry root, transaction root, membership root, and counters under a child signature. Let two anchors for the same child be

$$\mathsf{Anch}_1 = (\sigma_B, q_1, t_1, h^1_{\mathsf{reg}}, h^1_{\mathsf{tx}}, h^1_{\mathsf{mem}}, c_1, \sigma_1),$$

$$\mathsf{Anch}_2 = (\sigma_B, q_2, t_2, h^2_{\mathsf{reg}}, h^2_{\mathsf{tx}}, h^2_{\mathsf{mem}}, c_2, \sigma_2).$$

The adversary wins if:

- $q_1 = q_2$ but the Merkle roots differ,
- $q_1 < q_2$ yet the roots contradict the append only log semantics,
- the signature on either anchor verifies but does not correspond to a legitimate child,
- the adversary produces a root that matches the parent's expectations while reflecting inconsistent underlying logs.

Case analysis:

1. If two anchors with equal sequence numbers have different roots, the attacker has constructed two different Merkle commitments with identical signed metadata. This is impossible without forging signatures or finding hash collisions.
2. If the attacker presents a later anchor whose roots contradict earlier Merkle roots, they must contradict the append only nature of logs. The only way to achieve this is to construct a false Merkle root or forge a signature.

3. If the signature verifies but was not produced by the legitimate child, this is an EUF CMA forgery.

Thus, the adversary cannot produce inconsistent anchors without breaking the collision resistance of the hash function or the unforgeability of signatures. □

8 Transport Layer Security

This section establishes confidentiality, integrity, replay protection, ordering guarantees, and forward secrecy for UDIF transport sessions. The analysis applies to both UA to GC sessions and BC to BC trunk sessions, with the latter supporting periodic asymmetric ratchets.

8.1 Handshake Security

Theorem 6. *Let $\mathcal{A}$ be any probabilistic polynomial time adversary in the UDIF handshake game. Suppose the KEM is IND CCA secure and the signature scheme is EUF CMA secure. Then any adversary that distinguishes the UDIF session key from a random string, or impersonates either party in the handshake, can be used to break either the IND CCA security of the KEM or the EUF CMA security of the signature scheme. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{hs}}(\lambda) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. The UDIF handshake exchanges nonces, certificates, signatures, and two KEM ciphertexts. Key derivation is

$$\mathsf{IKM} = ss_A \| ss_B \| \mathsf{nonce}_A \| \mathsf{nonce}_B \| \mathsf{transcript\_hash}.$$

If an adversary distinguishes the derived key from random, then either:

- one of the KEM shared secrets has been distinguished from random, contradicting IND CCA security, or
- the adversary has impersonated a peer by forging the signed transcript, contradicting EUF CMA security.

Since both KEM contributions are included, the adversary must break at least one of the primitives to win. The random oracle and transcript binding eliminate reflection and unknown key share attacks. All other paths to advantage lead to negligible probability. □

8.2 Record Confidentiality and Integrity

Theorem 7. *Let $\mathcal{A}$ be an adversary in the UDIF record security game. Suppose the AEAD construction is IND CPA and INT CTXT secure and the header is authenticated as associated data. Then UDIF records are confidential and unforgeable under chosen ciphertext attacks, even in the presence of adversarial control of network scheduling, including reordering and chosen ciphertext submission. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{rec}}(\lambda) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AEAD}}(\lambda) + \mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{AEAD}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. Each UDIF record is encrypted as

$$c = \mathsf{Enc}(k, n, \mathsf{AAD}, m)$$

where the authenticated header AAD includes the sequence number, timestamp, epoch, and suite identifier. If an adversary distinguishes the ciphertext, it breaks IND CPA confidentiality of the AEAD. If it causes an honest node to accept a forged record, it breaks INT CTXT integrity.
Authenticated headers bind ordering, time, and suite information into the ciphertext. Because AAD is covered by the MAC, the adversary cannot modify sequence, epoch, or time fields without detection. Chosen ciphertext attacks cannot succeed because decryption rejects any record that fails either AEAD verification or header constraint checks.
Therefore the advantage of a successful attack is negligible. □

8.3 Replay, Reordering, and Time Window Enforcement

Theorem 8. *Let $\mathcal{A}$ be an adversary that controls the network and attempts replay or reordering attacks. Suppose timestamps are checked within a bounded window and sequence numbers must increase monotonically. Then any adversary that causes an honest node to accept a replayed or reordered record must either forge a valid AEAD ciphertext or break the header authentication. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{replay}}(\lambda) \leq \mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{AEAD}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. A UDIF receiver accepts a record only if:

$$\mathsf{seq} = \mathsf{last\_seq} + 1, \qquad |\mathsf{now} - \mathsf{utctime}| \leq 60, \qquad \mathsf{epoch} = \mathsf{current\_epoch},$$

and the AEAD tag verifies.
For a replayed record, the sequence number will not match. For a reordered record, the sequence number will not equal the expected value. For an out of window timestamp, the time check fails. Any tampering with these header fields requires forging a valid AEAD tag, since AAD is included in the MAC. Thus the only winning path for $\mathcal{A}$ is to break INT CTXT integrity, which is negligible by assumption.
Therefore replay and reordering attacks are prevented except with negligible probability. □

8.4 Forward Secrecy and Post Compromise Security of Trunks

Theorem 9. *Let $\mathcal{A}$ be an adversary in the forward secrecy game. Suppose the asymmetric ratchet uses fresh KEM encapsulations and updates keys according to*

$$\mathsf{IKM}_{i+1} = ss^A_{i+1} \| ss^B_{i+1} \| rn_A \| rn_B \| \mathsf{state}_i,$$

followed by cSHAKE based expansion. If the KEM is IND CCA secure and the hash function is collision resistant, then compromising all secret state at epoch i does not enable the adversary to derive keys from epoch $i+1$ or decrypt ciphertexts from earlier epochs. Therefore,

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{fs}}(\lambda) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. The ratchet introduces new entropy from two independent KEM contributions at each epoch. If an adversary compromises state_i, the derivation of state_{i+1} still includes ss^A_{i+1} and ss^B_{i+1}, which require breaking the KEM to predict. Because the ratchet discards all prior keys and state, recovering state_i does not reveal past secrets without inverting the hash function or forging earlier KEM ciphertexts.
Backward secrecy: Past keys depend on state_{i-1} which is erased, and reversing the hash chain requires breaking preimage resistance.
Forward secrecy: Future keys depend on fresh KEM encapsulations that are indistinguishable from random without breaking IND CCA security.
Thus UDIF trunk sessions achieve forward secrecy and post compromise security. □

9 Security of Queries and Peering Treaties

This section analyzes the privacy and containment guarantees of the UDIF query system. Queries operate under a predicate based minimal disclosure model, and cross domain forwarding is constrained by bilateral treaties. The results below show that no adversary can learn information beyond the allowed predicates unless it breaks the collision resistance of hash commitments, the unforgeability of capabilities, or the soundness of anchor propagation.

9.1 Minimal Disclosure for Local Queries

Theorem 10. *Let $\mathcal{A}$ be an adversary in the minimal disclosure game. Suppose the hash function used for attribute commitments and registries is collision resistant, and Merkle proofs are sound. Consider two worlds W_0 and W_1 that differ only in hidden attributes or registry entries, and agree on all predicate evaluations permitted by the adversary's capabilities. Then any adversary that distinguishes which world it interacts with can be used to either find a hash collision or produce inconsistent Merkle proofs. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{md}}(\lambda) \leq \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. A query response contains only:

$$\mathsf{verdict} \in \{0, 1, 2\}, \qquad \mathsf{proof}, \qquad \mathsf{signature}.$$

If the adversary distinguishes W_0 from W_1, then either:

- the predicate outcome differs, which contradicts the game definition, or
- the proof structure reveals information about hidden state.

A proof is a Merkle authentication path. If the adversary detects a difference in paths without a predicate change, the two worlds must use different leaves or internal nodes that hash to the same root. Since the Merkle root is fixed by anchored registry commitments, this implies either:

1. a collision in H, or
2. an invalid proof that contradicts the Merkle construction.

Thus no distinguishing strategy exists except by breaking the hash function. Hence the advantage is negligible. □

9.2 Cross Domain Query Containment

Theorem 11. *Let $\mathcal{A}$ be a cross domain adversary with treaty mediated access to a foreign domain. Suppose capabilities are unforgeable and the anchor system is sound. Then no adversary can learn information outside the predicate families permitted by the treaty scope without forging either a capability token, a certificate, or an anchor. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cdc}}(\lambda) \leq \mathsf{Adv}^{\mathsf{forge}}_{\mathsf{cap}}(\lambda) + \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{anch}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. A treaty specifies a scope bitmap indicating the allowed predicate types. The evaluation procedure is:

1. the origin GC validates the caller capability,
2. the GC checks that the predicate type is allowed by the treaty,

3. the GC forwards the query to the treaty peer,
4. the peer evaluates the predicate locally,
5. both sides log the query and response.

If an adversary extracts additional information, then one of the following must occur:

- the adversary submitted a predicate not covered by its capability bitmap,
- the adversary submitted a predicate not covered by the treaty bitmap,
- the foreign GC evaluated a query outside the treaty scope,
- the adversary suppressed or altered a logged predicate without detection.

The first two cases imply forging a capability digest or KMAC tag. The third case implies forging an anchor or altering logs, because query evaluations are logged and anchored in both domains. The fourth implies either breaking collision resistance of the membership log tree or producing a forged anchor.
Thus any violation of containment reduces to either capability forgery or anchor equivocation. Both are negligible by assumption. □

9.3 Auditability of Query Flows

Theorem 12. *Let $\mathcal{A}$ be an adversary attempting to repudiate treaty interactions or suppress evidence of cross domain queries. Suppose the membership logs and anchor chains are append only and collision resistant. Then no adversary can alter or remove evidence of a query, or present an inconsistent audit record across domains, without breaking hash collision resistance or anchor soundness. Therefore,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{audit}}(\lambda) \leq \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{Adv}^{\mathsf{sound}}_{\mathsf{anch}}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. Each query produces a log entry in both the origin and treaty peer domains. These entries include:

- a digest of the query,
- a type code,
- the capability reference,
- a verdict,
- a timestamp,
- a signature of the GC or BC.

Later, both logs are aggregated into the Merkle roots of Anchor Records. For the adversary to suppress or alter evidence of treaty interactions, it must either:

1. remove or alter a logged entry while preserving the Merkle root, or
2. produce two anchor sequences with contradictory log roots.

The first case implies a collision in H, because log entries are canonical and the Merkle root must remain unchanged. The second implies an anchor forgery or a break of append only log semantics. Both are ruled out by collision resistance and anchor soundness.
Therefore all treaty interactions are non repudiable and the adversary cannot produce inconsistent audit trails. □

10 Composition and End to End Security

This section provides a system level view of UDIF security and shows how the local guarantees proven in earlier sections compose into global security properties for full UDIF deployments. The intent is to demonstrate that UDIF behaves as a secure identity, authorization, and provenance framework when all components operate together as specified.

10.1 Composition Framework

We analyze UDIF as an interactive multi party protocol executed among Roots, Branch Controllers, Group Controllers, User Agents, and the adversarial environment. The execution model consists of:

- long term state held by Roots, BCs, and GCs,
- short lived sessions between UAs and their GCs,
- long lived trunk sessions between BCs,
- certificate issuance events,
- capability delegation and revocation events,
- object creation, update, and transfer events,
- registry and log updates with periodic anchoring,
- query evaluations with optional treaty forwarding.

Each component has already been shown to satisfy its respective security definition. To compose these guarantees, we use the following framework.

Independence of Subsystems. Certificates, capabilities, registries, anchors, and transport sessions operate with cryptographically independent keys and nonces. No key material is shared across primitives. This ensures that the failure of one subsystem cannot be used to attack another, except through violations already ruled out by reductions.

Canonical Encoding and Domain Separation. All committed structures are encoded with canonical TLV encoding under strict labels. The domain separation labels ensure that outputs of hash and MAC functions cannot be confused or substituted across contexts. This removes cross protocol interactions that commonly complicate composition.

Monotonicity of State. Logs, registries, and anchor chains are append only. The monotonic nature of their Merkle roots ensures that later states cannot contradict earlier states without breaking collision resistance. This provides temporal structure that simplifies global reasoning.

Authenticated Binding of Components. Certificates bind identity, permissions, and suites. Anchors bind logs. Transport binds time, sequence, and epoch metadata into AEAD associated data. Queries bind predicate semantics to capability tokens. These authenticated bindings enable modular composition proofs.

Adversarial Scheduling and Concurrency. The adversary controls message scheduling but does not break the cryptographic assumptions. UDIF components are sequentially composable since adversarial scheduling cannot remove authenticated bindings or violate monotonicity.

10.2 Main End to End Security Theorem

Theorem 13. *Let $\mathcal{A}$ be any probabilistic polynomial time adversary interacting with a full UDIF deployment under the system model described above. Suppose the signature scheme is EUF CMA secure, the KEM is IND CCA secure, the AEAD construction provides IND CPA and INT CTXT security, the hash function is collision resistant, and Merkle trees are sound. Then the global advantage of $\mathcal{A}$ in violating any UDIF security goal is bounded by the sum of the advantages in breaking these primitives. Formally,*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{UDIF}}(\lambda) \leq \mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{AEAD}}(\lambda) + \mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{AEAD}}(\lambda) + \mathsf{Adv}^{\mathsf{CR}}_{H}(\lambda) + \mathsf{negl}(\lambda).$$

Proof. Every global UDIF failure corresponds to a violation of one of the local security properties proven earlier. We consider the categories of global failures and apply reduction arguments.

Identity or permission failures. Certificate chain authenticity and capability soundness have already been reduced to EUF CMA and MAC unforgeability. Thus any attempt to impersonate an entity, escalate privileges, or inject unauthorized identity material reduces to signature or MAC forgery.

Object or registry failures. Contradictory object states or inconsistent registry views reduce to hash collisions or signature forgeries. All registry states are Merkle committed and included in anchored logs which are append only.

Anchor chain failures. Any equivocation of logs or anchors also reduces to either a hash collision or a signature forgery. The monotonic sequence property enforces unique progression.

Transport failures. Confidentiality and integrity failures reduce to breaking IND CPA or INT CTXT security of the AEAD or IND CCA security of the KEM. Replay and reordering attacks reduce to INT CTXT because all header metadata is authenticated.

Query and treaty failures. Minimal disclosure failures reduce to hash collisions in committed structures. Treaty scope violations reduce to capability or certificate forgeries, or anchor equivocation. All query flows are logged, and logs are anchored, so any suppression or modification of evidence reduces to hash or signature forgery.

Composition. Since each subsystem operates under independent keys and is protected by authenticated outputs, no combined attack provides more advantage than the sum of the advantages for each primitive. All reductions are tight up to negligible loss due to simulation overhead.
Thus the adversary cannot violate any global UDIF security property without breaking at least one underlying primitive. The right hand side bounds the adversarial advantage by the sum of negligible values, completing the proof. □

11 Adversarial Work Factors and Parameter Choices

This section provides concrete security estimates for the cryptographic primitives used in UDIF and discusses how the recommended parameter sets contribute to the overall security margin. Although the formal proofs rely on asymptotic assumptions, a deployment requires concrete work factor estimates against the best known classical and quantum adversaries.

11.1 Work Factor Estimates

UDIF supports two signature families (ML DSA and SLH DSA), two KEM families (ML KEM and Classic McEliece), and a sponge based AEAD (RCS with KMAC) built upon SHA3. The security levels of these primitives are well understood.

ML KEM. ML KEM at level 5 offers an estimated cost of at least 2^{256} classical operations and 2^{128} for quantum search type adversaries. The best known lattice attacks, including primal and dual attacks, remain significantly above the 256 bit classical security target for the recommended parameters.

Classic McEliece. Classic McEliece with binary Goppa codes remains resistant to both classical and quantum structural attacks. The work factor is conventionally estimated at 2^{256} for level 5 parameters. No subexponential quantum attack is known.

ML DSA. ML DSA at level 5 provides approximately 2^{256} classical security. Quantum attacks based on Grover's algorithm reduce the security margin to roughly 2^{128} but remain at or above the desired requirement for long term identity and audit preservation.

SLH DSA. SLH DSA, based on hash based signatures, offers concrete security equal to the collision and preimage security of the underlying hash function. With SHA3 256, this is at least 2^{128} for collision resistance and 2^{256} for preimage resistance. The overall signature scheme inherits the lower bound.

Rijndael based RCS and KMAC. The wide block RCS construction operates over a 256 bit capacity and uses sponge permutations based on Keccak. Keccak 1600 provides at least 2^{256} security for preimage and related forging attacks in its capacity constrained modes. The indifferentiability of KMAC and cSHAKE bounds MAC forging advantages by roughly 2^{-256}.

Merkle Trees and SHA3 Commitments. Registry and anchor commitments use SHA3 256. Collision resistance is 2^{128}, and preimage resistance is 2^{256}. Since the anchor chain extends indefinitely, registry and anchor proofs inherit the long term binding properties of SHA3.
Taken together, the adversarial work factor required to violate UDIF in any subsystem is dominated by the smallest of the primitive security levels, which is at least 2^{128} in the quantum setting and at least 2^{256} in the classical setting. UDIF therefore matches or exceeds the standard 128 bit post quantum security target across all families.

11.2 Discussion of Suite Configurations

A UDIF suite specifies a complete set of cryptographic primitives selected at compile time. The choice of suite impacts performance, bandwidth, audit durability, and long term cryptographic confidence.

Lattice based suite. An ML KEM plus ML DSA suite yields the smallest certificate and anchor sizes and the fastest handshake operations. This configuration is suitable for high throughput deployments. Its long term quantum security rests on the hardness of module lattice problems.

Hash based suite. An ML KEM plus SLH DSA suite provides maximal conservatism for signatures, since SLH DSA reduces to hash based hardness alone. Signature sizes and verification costs are larger but provide high assurance for domains requiring multi decade audit durability.

Code based suite. A McEliece plus ML DSA suite offers very high security margins for KEM operations at the cost of larger public keys. This configuration suits deployments where bandwidth is abundant and the long term failure probability must be minimized.

AEAD and hash families. All suites use RCS plus KMAC for AEAD and SHA3 256 for commitments. These primitives balance performance, security, and implementation simplicity and remain conservative under both classical and quantum models.
Tradeoffs between suites do not affect the validity of earlier proofs. All suites provide at least 128 bit post quantum security and at least 256 bit classical security.

12 Limitations and Discussion

This section discusses the limitations of the UDIF security model, including modeling choices, trust assumptions on domains, and caveats for real world deployment. While the formal analysis covers the core cryptographic mechanisms, some classes of attacks are outside scope and must be addressed by implementation policy or environment constraints.

12.1 Modeling Limitations

The formal model assumes that all cryptographic primitives behave according to their idealized security assumptions. Several classes of attacks remain outside the present analysis.

Side channel leakage. The analysis assumes that all cryptographic implementations operate in constant time and do not leak key dependent information. Timing attacks, power analysis, and fault attacks must be mitigated at the implementation level.

Random number generation. The model assumes access to high quality randomness. Failures in random number generation can compromise key generation, signatures, and KEM security but are not modeled.

Domain trust assumptions. The analysis assumes correct Root behavior. A malicious or compromised Root can issue arbitrary certificates, override capabilities, and alter anchor histories. Root trust is a global assumption inherent to the UDIF hierarchy.

Physical access and endpoint compromise. UDIF does not protect against adversaries with physical access to a UA or GC endpoint. Compromise of endpoint state falls outside the protocol model and must be addressed by operational controls.

Out of band policy failures. Policies enforced by access masks and capabilities assume correct administrative behavior. Errors in policy configuration or human oversight are outside scope.

The following subsection will address deployment caveats, operational concerns, and mitigation strategies for environments where these limitations may arise.

References

1. National Institute of Standards and Technology (NIST). *FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.* U.S. Department of Commerce, 2015. Available at: /mnt/data/udif_specification.pdf.
2. National Institute of Standards and Technology (NIST). *FIPS 203: Module-Lattice-Based Key Encapsulation Mechanism (ML-KEM).* U.S. Department of Commerce, 2024. Available at: https://doi.org/10.6028/NIST.FIPS.203.
3. National Institute of Standards and Technology (NIST). *FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA).* U.S. Department of Commerce, 2024. Available at: https://doi.org/10.6028/NIST.FIPS.204.
4. National Institute of Standards and Technology (NIST). *SP 800-185: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash.* U.S. Department of Commerce, 2016. Available at: https://doi.org/10.6028/NIST.SP.800-185.
5. Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. *Keccak Specifications and Implementations.* NIST, 2012. Available at: https://doi.org/10.6028/NIST.SP.800-185.
6. Underhill, J. G. *Universal Digital Identity Framework (UDIF) Specification.* Quantum Resistant Cryptographic Solutions Corporation, 2025. Available at: /mnt/data/udif_specification.pdf.
7. Underhill, J. G. *RCS: A Wide-Block Sponge-Based AEAD Construction.* Quantum Resistant Cryptographic Solutions Corporation, 2024. Available at: https://www.qrcscorp.ca.
8. Underhill, J. G. *Quantum Secure Tunnelling Protocol (QSTP) Specification.* Quantum Resistant Cryptographic Solutions Corporation, 2024. Available at: https://www.qrcscorp.ca.