A Formal Analysis of the Anonymous Encrypted Relay Network Protocol

John G. Underhill

Quantum Resistant Cryptographic Solutions Corporation

Abstract. The Authenticated Encrypted Relay Network (AERN) is a post quantum anonymity system that combines centralized certificate anchored trust with multi hop authenticated encryption to provide confidentiality, integrity, and unlinkability in adversarial environments. This work presents an extended and rigorous formal cryptanalysis of the AERN protocol and develops game based proofs that capture all major security objectives, including mutual authentication, key indistinguishability, weak forward secrecy, post compromise security, channel confidentiality, channel integrity, replay resistance, and traffic analysis resistance. The analysis is carried out under adversaries with global network visibility, adaptive active interference capabilities, partial node compromise, and quantum computational power.

We formalize AERN using a structured protocol specification and define a complete set of security games for each property. The results show that handshake secrecy and mutual authentication reduce to the IND CCA2 security of the underlying post quantum KEM and the EUF CMA security of the signature scheme. We establish weak forward secrecy and post compromise security by showing that the key schedule regenerates fresh entropy through new KEM exchanges and that session keys remain independent of long term secrets after erasure of ephemeral material. For channel security, we prove that each hop achieves authenticated encryption security under the pseudo-randomness of the RCS cipher and the integrity guarantees of KMAC, and we provide explicit bounds on the adversary's advantage.

For anonymity, we define a formal indistinguishability game in which an adversary attempts to link a client to one of two candidate destinations or attempts to identify which of two clients originated a flow. We prove that AERN achieves negligible advantage against this adversary whenever at least one endpoint of each route remains honest. We further introduce a k anonymity bound that quantifies anonymity degradation under partial compromise and show that the route selection mechanism produces a uniform distribution over eligible paths, which is essential for resisting path bias attacks. Additional analysis shows that fixed packet sizes, per packet route diversity, and authenticated metadata prevent traffic correlation across multiple hops.

Finally, we extend the model to cover systemic risks stemming from the central trust components of AERN. We define formal models for offline and malicious ARS or ADC behavior and prove that forward secrecy limits the effect of trust anchor compromise by preventing retroactive recovery of session keys or payloads. Together, these results provide a comprehensive and formally verified foundation for the security of AERN and clarify the assumptions required for secure real world deployment.

1 Introduction

1.1 Background and Motivation

Anonymity networks are designed to conceal the relationship between a client and the destinations it contacts across an untrusted communication environment. Classical systems, such as Tor and related onion routing designs, provide anonymity through layered encryption and multi hop routing, but these systems face fundamental challenges. They rely on classical cryptographic assumptions, employ long lived keys, and operate within decentralized trust models that expose the system to global correlation attacks and adversarial node infiltration. These limitations become more pronounced in the presence of widespread traffic monitoring, large scale active interference, and quantum capable adversaries.

The emergence of post quantum cryptography provides an opportunity to redesign anonymity networks with stronger cryptographic foundations and more reliable authentication infrastructures. The Authenticated Encrypted Relay Network (AERN) is intended to meet these requirements. AERN combines a centralized certificate anchored trust model with a multi hop authenticated encryption architecture that uses post quantum key establishment, deterministic signatures, and SHAKE based key derivation. The system provides authenticated routing, per packet randomized paths, fixed size encrypted payloads, and controlled trust anchors that can be analyzed with formal models. This structure allows AERN to achieve stronger guarantees for confidentiality, integrity, and unlinkability than earlier systems while maintaining resistance to quantum capable adversaries.

1.2 Overview of AERN

AERN consists of several cooperating components that maintain global trust and support secure multi hop communication. The AERN Root Security server, or ARS, serves as the trust anchor and signs all node certificates. The AERN Domain Controller, or ADC, distributes the certified topology and ensures that nodes maintain a consistent network state. Proxy nodes (APS) form the relay mesh through which encrypted packets are forwarded. Clients establish symmetric tunnels with selected proxies using a post quantum Key Encapsulation Mechanism and derive authenticated encryption keys from the shared secret. Each packet is routed through a sequence of proxies selected at random from the certified topology, and each hop processes encrypted and authenticated metadata that determines the forwarding decision.

Traffic is encapsulated in fixed size ciphertexts, which hide information about packet length and reduce correlation channels. The combination of certificate anchored authentication, randomized routing, and symmetric encrypted tunnels provides a foundation for provable anonymity and multi hop confidentiality. AERN is designed to resist global passive observation, targeted active interference, and partial compromise of proxy nodes.

1.3 Contributions of this Work

This paper provides a complete and strengthened formal cryptanalysis of the AERN protocol. It extends prior analysis by introducing rigorous game based definitions for all major security properties and by aligning the proofs with modern post quantum cryptographic standards. The contributions of this work include the following.

First, we define formal security games for handshake correctness, mutual authentication, key indistinguishability, weak forward secrecy, post compromise security, channel confidentiality, channel integrity, and replay resistance. These games capture the interactions of a quantum capable adversary with oracle access to protocol messages, node compromise events, and authenticated metadata. We provide explicit reductions that tie each property to the

security of ML-KEM, ML-DSA, the SHAKE based KDF, and the KMAC authentication layer.
Second, we develop a strengthened anonymity model for AERN. This model defines a formal indistinguishability game for client destination unlinkability and provides a k anonymity bound that quantifies privacy loss under partial compromise of relay nodes. We introduce a route selection uniformity lemma that proves the distribution of multi hop paths is uniform over the certified topology when seeded with fresh KDF output.
Third, we provide a formal model for systemic trust and analyze the implications of ARS or ADC compromise. We show that forward secrecy prevents retroactive decryption even under trust anchor compromise and outline the conditions under which a malicious or offline ARS, APS, or ADC cannot violate the cryptographic properties of active or completed sessions.
These contributions produce a complete and verifiable foundation for AERN security and establish the assumptions necessary for secure deployment.

1.4 Structure of the Paper

The remainder of the paper is structured as follows. Section 2 introduces notation, summarizes protocol components, and defines the message formats using structured protocol notation. Section 3 presents the adversary model and formal security games for each security property. Section 4 provides a complete formal specification of AERN, including its certificate lifecycle, handshake protocol, tunnel operation, and route selection algorithm. Section 5 describes the cryptographic assumptions that underpin the security reductions. Section 6 gives proofs for handshake secrecy, mutual authentication, weak forward secrecy, and post compromise security. Section 7 presents the authenticated encryption analysis and the replay resistance proofs. Section 8 develops the anonymity and traffic correlation models and proves the k anonymity and path uniformity results. Section 9 formalizes systemic risks and models trust anchor compromise. Section 10 concludes with a synthesis of the results and discusses implications for real world deployment.

2 Engineering Description of AERN

This section gives a specification faithful engineering description of AERN at the level required by a formal analysis. The aim is to describe the protocol as it operates in deployed systems.
AERN is built around a certificate anchored trust architecture. The ARS acts as the sole signing authority and issues root signed device certificates. Each device generates its own asymmetric signature key pair and submits the public verification key to the ADC, which forwards it to the ARS for signing. Certificates contain only identity and signing material. All cipher keys in AERN are ephemeral and created only during initialization or tunnel establishment.
The ADC maintains the active topology, including the list of proxy servers (APS) and their certificates. It distributes this information as a signed topology snapshot. Devices accept new topology only after verifying the ARS signature and ensuring that the version number increases. This signed snapshot gives every node a consistent global view of the proxy network.
When a proxy enters service, it completes registration, loads its root signed certificate, and obtains the current topology from the ADC. Once online, each proxy establishes encrypted control and relay channels with its peers. These channels are secured using a master encryption key that is derived through an asymmetric key exchange between the two proxies. The derived master key is expanded using a SHAKE based KDF to produce the symmetric cipher keys and nonces that initialize the RCS instances used for inter proxy

traffic. Proxies repeat this process with every peer where a tunnel is required. Because certificate keys are only signing keys, the asymmetric cipher key material used for the key exchange is freshly generated and never reused across tunnels.

Clients integrate into the network by registering with the ADC, receiving their signed device certificate, and obtaining a proxy list and topology hash. When the topology hash changes, the client refreshes the list. A client selects an entry proxy by drawing a random index over this list, establishes a single encrypted tunnel to that entry proxy, and routes all traffic through it. The tunnel is created using the same asymmetric key exchange mechanism used between proxies. The client sends its certificate, a freshly generated public cipher key, and a timestamp hash signed with its private signing key. The entry proxy verifies the certificate chain, verifies the signature on the timestamp hash, computes the shared secret using the client's public cipher key, signs the response metadata, and derives the symmetric keys needed for the bidirectional tunnel. The client performs the corresponding verification and key derivation steps. Once both sides initialize their RCS instances, the tunnel is active and ready for data.

After the client establishes its entry tunnel, each outgoing packet is handled independently. The entry proxy selects a path through the network using the topology snapshot distributed by the ADC. The specification defines a simple mechanism: a hop count is chosen uniformly from a configured range, and each hop is selected by sampling uniformly from the proxy list. Every hop in the path is represented by a sixteen bit route hint that indexes into the device topology list, which all proxies maintain identically. The result is a compact and deterministic representation of a route that is locally resolvable at each hop without negotiation or interaction with the ADC.

Packet structure is uniform across the entire proxy mesh. Every relay packet is exactly 1500 bytes: a fixed header followed by a fixed length encrypted payload region. The header contains the route hints, the hop index, a per hop sequence number, a timestamp, and packet flags. The payload area carries exactly 1446 bytes of encrypted data. The entire packet is authenticated with KMAC using the per hop symmetric keys. Because size never varies, and because each packet may follow a different route, external observers cannot infer traffic characteristics from packet length or routing consistency.

Replay protection is enforced through authenticated sequence numbers and timestamp validation at every hop. Each tunnel direction tracks the highest accepted sequence number and enforces a bounded acceptance window. Timestamps must lie inside a configured freshness period. A packet that falls outside these bounds, or whose authentication tag fails, is discarded before any state is updated. The use of authenticated metadata ensures that replay attempts require either violating the freshness bounds or forging a KMAC tag.

When certificates expire or when a tunnel is torn down, all secrets derived from that certificate or tunnel are erased. This includes master encryption keys, symmetric tunnel keys, nonce state, and all cipher contexts. Forward secrecy is achieved through the fact that all shared secrets are produced from ephemeral asymmetric exchange steps, never stored in certificates, and deleted immediately after tunnel setup or tear-down. Because the master keys are derived purely from these ephemeral secrets, long term compromise of device certificates or signing keys does not retroactively reveal past traffic.

This engineering description reflects the protocol as defined by the AERN specification: certificate anchored identity, centrally distributed topology, ephemeral asymmetric key exchange for all cipher keys, symmetric per hop tunnel encryption, fixed size packets, and randomized per packet routes. These elements form the concrete substrate on which the formal model in the following sections is built.

3 Protocol Summary and Notation

3.1 Notation Table

Table 1 summarizes the notation used throughout the analysis. All symbols remain consistent with the specification and are chosen to support the game based definitions in later sections.

Table 1: Summary of notation used throughout the paper.

Symbol	Meaning
λ	Global security parameter
$\mathcal{N}$	Set of all AERN nodes
C_D^σ	Certificate of device D signed by the ARS root key
H_{CS}^σ	Signed hash of certificate and timestamp metadata
$\mathrm{pk}_i, \mathrm{sk}_i$	Public and private keys of node i
$\mathrm{kem.pk}_i, \mathrm{kem.sk}_i$	KEM key pair for node i
(c, ss)	KEM ciphertext and shared secret
$\mathrm{Sign}_{\mathrm{sk}}(m)$	Signature on message m
$\mathrm{Verify}_{\mathrm{pk}}(m, \sigma)$	Verification of signature σ
$\mathrm{KDF}(ss, \mathrm{ctx})$	SHAKE based key derivation on shared secret
K_{enc}, K_{mac}	Per tunnel encryption and MAC keys
σ_{state}	Cipher state for RCS stream cipher
Hdr	Authenticated packet header
Enc	RCS encrypted payload
Tag	KMAC authentication tag
Route	Ordered sequence of proxy identifiers
$P_1, \ldots, P_h$	Entry through exit nodes in a multi hop route
TState	Local topology state of a node
Adv	Adversary advantage in a security game

3.2 System Participants and Roles

AERN consists of four classes of entities, each with certified identities and specific responsibilities.

AERN Root Security Server (ARS). The ARS acts as the root of trust. It generates the global signing key pair, issues all child certificates C_D^σ, validates topology updates, and maintains authoritative state across the network.

AERN Domain Controller (ADC). The ADC distributes topology information, manages registration events, stores the canonical certificate list, and ensures that all nodes operate with the same authenticated network view. The ADC never generates signatures but distributes ARS signed data.

Proxy Nodes (APS). Proxies are certified relays that forward encrypted packets along a route. Each proxy holds a unique certified key pair, a KEM key pair, and tunnel state for active sessions.

Clients. Clients initiate end to end sessions, validate certificates, perform KEM encapsulation with proxies, derive tunnel keys from the resulting shared secret, and generate per packet routes.

3.3 Cryptographic Primitives and Parameters

AERN uses standardized post quantum and symmetric primitives:
ML-KEM (FIPS 203). Used for handshake key establishment. Provides IND CCA2 security. Alternatively, the implementation can use the Classic McEliece code-based asymmetric cipher. The asymmetric cipher type and parameter sets are defined at compile time.
ML-DSA (FIPS 204). Used for ARS certificate issuance and authentication of device identities. Provides EUF CMA security. AERN can also use the SLH-DSA (SPHINCS+) hash based signature scheme. The asymmetric signature scheme type and parameter sets are defined at compile time.
SHAKE based Key Derivation. Used to derive tunnel keys from the KEM shared secret through a context bound KDF.
RCS Stream Cipher. Used for payload encryption within each hop. Modeled as a pseudo-random keystream generator keyed with K_{enc}. RCS is an AEAD cipher that uses Rijndael-256 for the primary encryption function, cSHAKE as a tweakable key schedule, and KMAC for authentication.
KMAC (SP 800 185). Used to authenticate ciphertext and header metadata for each packet.
These primitives together support authenticated encryption properties for tunnels and enable formal reductions for the security proofs.

3.4 Message Types and Packet Structure

All AERN communication is encoded using authenticated and encrypted message formats.
Certificate Messages. Carry C_D^{σ} and H_{CS}^{σ}, used during topology synchronization and handshake verification.
Handshake Messages. Implement the KEM key exchange. The primary fields are the KEM ciphertext c, a client generated nonce, certificate information for the proxy, and an authenticated timestamp.
Tunnel Packets. Each encrypted packet has the form

$$\text{Pkt} = (\text{Hdr}, \text{Enc}, \text{Tag})$$

with:

- Hdr: hop index, sequence number, timestamp, version, route hash,
- Enc: RCS encrypted payload,
- Tag: KMAC authentication tag.

All packets are padded to a fixed length as mandated by the specification to prevent size based correlation.
Metadata Integrity. All fields that influence routing or replay logic are included in Hdr and authenticated with Tag.

3.5 Security Protocol Notation for AERN Flows

We use a structured Security Protocol Notation (SPN) format to express message flows precisely. Each message is represented as:

$$A \rightarrow B : m$$

indicating that entity A sends message m to entity B.

The handshake is expressed as:

$$C \to P : C_C^{\sigma}, c, \text{nonce}_C, t_C$$
$$P \to C : C_P^{\sigma}, \text{nonce}_P, t_P, \text{auth\_data}$$

Tunnel packet forwarding across h hops is expressed as:

$$P_i \to P_{i+1} : \text{Hdr}_i, \text{Enc}_i, \text{Tag}_i.$$

This notation enables unambiguous definition of oracles and adversarial interaction in Section 3 and supports the formal reductions in later sections.

4 Adversary and Security Models

4.1 Adversary Capabilities

The adversary is modeled as a probabilistic polynomial time algorithm with the ability to observe, delay, reorder, drop, inject, and modify packets on any network link. The adversary has complete visibility of the network topology, including all certified public information distributed by the ADC. The adversary may compromise selected nodes and obtain their long term private keys, active tunnel keys, ephemeral state, and any unencrypted memory accessible at the moment of compromise.
The adversary is assumed to have quantum computational capabilities for operations on public parameters and may use these capabilities when attempting to break the post quantum KEM or signature scheme. The adversary cannot break authenticated signature verification or KEM decryption except by violating the corresponding cryptographic assumptions. All nodes under adversarial control follow an adversary specified algorithm rather than the protocol.

4.2 Compromise and Exposure Model

AERN security depends on the treatment of three classes of secrets: long term keys, session keys, and ephemeral secrets. The adversary may interact with the protocol through the following compromise oracles.
Long term key compromise. The adversary may query $\mathsf{CorruptLT}(i)$ to obtain the long term signing key and long term KEM secret key of node i.

Ephemeral state compromise. The adversary may query
$\mathsf{CorruptEphemeral}(i, s)$ during session s to obtain ephemeral KEM secrets, KDF intermediate values, and cipher states. Ephemeral secrets are assumed to be erased after use.

Session key reveal. The adversary may query $\mathsf{Reveal}(i, s)$ to obtain the session keys held by node i for session s unless s is the challenge session in a key indistinguishability game.

Control over malicious nodes. Nodes that the adversary corrupts behave according to adversarial instructions, including forging headers, altering routing metadata, or relaying malformed packets.
All compromise oracles are included in the formal games defined below.

4.3 Game Based Security Definitions

AERN security is expressed using a collection of games. Each game formalizes a security property and defines the adversary's interaction through explicit oracle access. The games

are aligned with the ML-KEM and ML-DSA security definitions and use the notation introduced in Section 2.

4.3.1 Game for Mutual Authentication

The mutual authentication game $\mathsf{Game}_{\mathrm{AUTH}}$ models an adversary who attempts to make an honest node accept a peer identity without a matching honest handshake instance.

The adversary interacts with:

- $\mathsf{Send}(i, m)$ which delivers message m to node i,
- $\mathsf{CorruptLT}(j)$ which reveals sk_j,
- $\mathsf{Reveal}(j, s)$ for non challenge sessions.

The adversary succeeds if an honest node i accepts a session with purported peer j but no matching session exists in which node j produced the corresponding handshake messages. The advantage is

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{AUTH}} = \Pr[\mathcal{A} \text{ wins } \mathsf{Game}_{\mathrm{AUTH}}].$$

AERN achieves mutual authentication if this advantage is negligible.

4.3.2 Game for Key Indistinguishability

Key indistinguishability is defined using a challenge session $s^\star$.

The adversary interacts with:

- $\mathsf{Send}(i, m)$,
- $\mathsf{CorruptLT}(j)$,
- $\mathsf{Reveal}(i, s)$ for all $s \neq s^\star$.

When the adversary queries $\mathsf{Test}(i, s^\star)$, the challenger samples a bit b and returns either:

$$K_i^\star \quad \text{if } b = 1$$

or a uniform random key if $b = 0$.
The adversary outputs a guess b'. Its advantage is

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

AERN achieves key indistinguishability if this advantage is negligible.

4.3.3 Game for Weak Forward Secrecy

Weak forward secrecy models compromise of long term keys after a session completes. The adversary may call:

- Send and Reveal for all sessions,
- $\mathsf{CorruptLT}$ only after the challenge session completes.

When the adversary queries Test for the challenge session, the challenger again returns a real or random key. The adversary wins if it distinguishes the key after observing the long term key of one or both parties.
AERN achieves wPFS if the advantage in this game is negligible.

4.3.4 Game for Post Compromise Security

Post compromise security models the scenario where the adversary corrupts a node, learns all secrets, and then observes a recovery event followed by a new session. The adversary may:

- Query CorruptLT and CorruptEphemeral at arbitrary times,
- Force recovery events which regenerate fresh KEM keys,
- Query Test only on the first session after recovery.

The adversary wins if it distinguishes the new session key from random. AERN satisfies PCS if this advantage is negligible.

4.3.5 Game for AEAD Tunnel Security

Tunnel security is analyzed using an IND CPA plus INT CTXT style authenticated encryption game. The adversary interacts with:

- Enc(M) returning authenticated ciphertexts,
- Dec(C) returning either plaintext or rejection,
- CorruptLT and CorruptEphemeral for non challenge sessions.

The adversary attempts to win by either distinguishing challenge encryptions or forging a ciphertext with a valid authentication tag. The advantage is the sum of confidentiality and integrity advantages.
AERN achieves AEAD tunnel security if this advantage is negligible.

4.3.6 Game for Replay Resistance

The replay game models an adversary who attempts to cause an honest node to accept previously transmitted packets. The adversary has access to:

- Enc,
- Send,
- CorruptLT or CorruptEphemeral.

The adversary wins if an honest node accepts a stale packet. Replay resistance holds if the adversary's advantage in this game is negligible.

4.3.7 Game for Anonymity and Unlinkability

The anonymity game $\mathsf{Game}_{\mathrm{ANON}}$ considers two possible communication scenarios chosen by the adversary:

$$(C_0, D_0) \quad \text{or} \quad (C_1, D_1).$$

The challenger selects $b \in \{0, 1\}$ and executes the session corresponding to (C_b, D_b).
The adversary may:

- Observe all encrypted traffic,
- Compromise any subset of interior proxies,

- Query CorruptLT and CorruptEphemeral on non endpoint nodes.

The adversary outputs a bit b'. The advantage is

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

AERN achieves unlinkability if this advantage is negligible when at least one endpoint remains honest and achieves k anonymity if the adversary's advantage under k candidates is bounded by $1/k$ plus negligible factors.

5 Formal Specification of AERN

5.1 Global State and Topology

AERN maintains a global authenticated topology that describes all nodes, their certified identities, and the routing information needed for multi hop forwarding. The topology is anchored in the ARS root key and is distributed by the ADC. Every node stores a local view of this topology in the form

$$\mathsf{TState} = (\mathsf{NodeList}, \mathsf{CertList}, \mathsf{Adjacency}),$$

where:

- NodeList is the set of identifiers for all known nodes,
- CertList contains certificates C_D^{σ} for each node D,
- Adjacency contains the routing relevant neighbor relationships or the proxy set from which routes may be drawn.

Each node maintains a local state consisting of:

$$\mathsf{State}_i = (\mathsf{TState}_i, \mathsf{Keys}_i, \mathsf{Sessions}_i, \mathsf{CipherState}_i),$$

where Keys_i contains long term signing keys and KEM keys, $\mathsf{Sessions}_i$ contains established tunnels, and $\mathsf{CipherState}_i$ contains the active RCS cipher state for each tunnel direction. Topology synchronization occurs when the ADC distributes updated topology files. The ARS signs all certificates and ensures that each entry in the topology is validated before distribution. All honest nodes verify ARS signatures before accepting topology updates.

5.2 Certificate Lifecycle and Trust Anchors

AERN uses a hierarchical certificate structure rooted at the ARS. The ARS generates the root signing key pair $(\mathrm{pk}_{\mathrm{ARS}}, \mathrm{sk}_{\mathrm{ARS}})$. Each node D submits its public parameters to the ADC, which forwards them to the ARS for signing.
A certificate for device D has the form

$$C_D^{\sigma} = (D, \mathrm{pk}_D, \mathrm{kem.pk}_D, \mathrm{role}_D, H_{\mathrm{CS}}, \sigma),$$

where:

- pk_D is the ML-DSA verification key,
- $\mathrm{kem.pk}_D$ is the ML-KEM public key,
- role_D identifies D as ARS, ADC, Proxy, or Client,
- H_{CS} is a timestamp bound hash of certificate metadata,

- $\sigma = \mathrm{Sign}_{\mathrm{sk}_{\mathrm{ARS}}}(D, \mathrm{pk}_D, \mathrm{kem.pk}_D, \mathrm{role}_D, H_{\mathrm{CS}})$.

Nodes verify certificates using $\mathrm{pk}_{\mathrm{ARS}}$ before accepting them. Revocation is modeled by removal of C_D^σ from CertList. The specification requires that topology files include a monotonically increasing version to prevent rollback attacks.

5.3 Formal Handshake Specification (SPN format)

We formalize the handshake using the Security Protocol Notation (SPN). Let C be a client and P be a proxy. The handshake establishes a symmetric tunnel with keys derived from a KEM shared secret.

Step 1: Client to Proxy. The client selects ephemeral randomness r_C and computes

$$(c, ss) \leftarrow \mathrm{KEM.Encaps}(\mathrm{kem.pk}_P, r_C).$$

It sends:

$$C \to P : (C_C^\sigma, c, \mathrm{nonce}_C, t_C, H_{\mathrm{CS}}^\sigma)\,.$$

Step 2: Proxy validation. Upon receipt, P:

- verifies C_C^σ using $\mathrm{pk}_{\mathrm{ARS}}$,
- computes $ss \leftarrow \mathrm{KEM.Decaps}(c, \mathrm{kem.sk}_P)$,
- derives tunnel keys using

$$K_{\mathrm{enc}}, K_{\mathrm{mac}} = \mathrm{KDF}(ss, \mathrm{context}),$$

  where context contains both certificates and authenticated metadata.

Step 3: Proxy to Client. The proxy responds with:

$$P \to C : (C_P^\sigma, \mathrm{nonce}_P, t_P, H_{\mathrm{CS}}^\sigma)\,.$$

The client verifies C_P^σ, accepts ss, and derives the same $(K_{\mathrm{enc}}, K_{\mathrm{mac}})$.

Correctness. KEM correctness ensures both parties obtain the same ss. The KDF deterministically expands ss into identical tunnel keys.

5.4 Tunnel Operation and Key Schedule

After the handshake, all communication between C and P is encrypted and authenticated using the derived keys and per direction cipher states.
Each packet transmitted along a tunnel has the form

$$\mathrm{Pkt} = (\mathrm{Hdr}, \mathrm{Enc}, \mathrm{Tag})$$

with:

$$\mathrm{Enc} = \mathrm{RCS.Enc}(K_{\mathrm{enc}}, \sigma_{\mathrm{state}}, M),$$
$$\mathrm{Tag} = \mathrm{KMAC}(K_{\mathrm{mac}}, \mathrm{Hdr} \parallel \mathrm{Enc}).$$

Encryption process. For each packet:

1. The node increments its sequence number and embeds it in Hdr.
2. The node attaches a timestamp t and hop index i in Hdr.
3. The payload M is encrypted under the current cipher state.
4. The tag is computed over the header and encrypted payload.
5. Packet size is padded to a fixed length before transmission.

Decryption and forwarding. Upon receiving a packet, a node:

1. verifies sequence and timestamp freshness,
2. verifies the KMAC tag using K_{mac},
3. decrypts M using RCS with σ_{state},
4. updates the cipher state,
5. increments the hop index and forwards if appropriate.

Ephemeral cipher states are erased when tunnels are reset or re-keyed.

5.5 Route Selection and Uniformity Model

AERN employs probabilistic multi-hop routing in which packet paths are constructed by network proxies. Route selection is performed independently for each packet, subject to fixed session endpoints, in order to balance unlinkability, load distribution, and resistance to traffic analysis.

Session endpoints. For each session, a single entry proxy P_1 is selected by the client from the set of available ingress proxies provided by the ADC. An exit proxy P_h is selected when the first outbound route is constructed and remains fixed for the lifetime of the session. These two proxies define the session endpoints and do not vary across packets.

Hop count selection. For each packet, the routing proxy draws a hop count

$$h \in \{h_{\min}, \ldots, h_{\max}\},$$

according to the distribution defined in the specification. The hop count includes the fixed entry and exit proxies, so that the number of interior hops is $h - 2$.

Interior route generation. Given the selected hop count, the routing proxy samples an ordered sequence of interior proxies

$$(P_2, \ldots, P_{h-1}) \leftarrow \mathsf{RouteGen}(h-2),$$

where $\mathsf{RouteGen}$ selects proxies uniformly at random from the eligible proxy set, subject to the constraint that no proxy appears in consecutive positions. The resulting packet route is therefore:

$$(P_1, P_2, \ldots, P_{h-1}, P_h),$$

with fixed endpoints and freshly sampled interior hops.

Per-packet independence. For successive packets within the same session, the entry and exit proxies P_1 and P_h remain constant, while the interior hop sequence $(P_2, \ldots, P_{h-1})$ is resampled independently for each packet. This ensures that interior path segments are statistically independent across packets, even when they share common session endpoints.

Reverse-path routing. For return traffic, the exit proxy performs an analogous route-generation process. The entry and exit proxies associated with the session remain fixed, while the interior hops for each return packet are sampled independently according to the same uniformity and non-repetition constraints.

Uniformity properties. Conditioned on fixed session endpoints, the distribution over interior routes of length $h - 2$ is uniform over the admissible proxy sequences defined by $\mathsf{RouteGen}$. As a result, no interior proxy can distinguish whether it occupies a particular position in the route based on path-selection bias alone.

Security implications. This routing model ensures that no single proxy, except in the degenerate case of collusion between the session entry and exit proxies, can reconstruct complete packet paths. Fixed endpoints provide session stability, while per-packet interior randomization limits correlation and supports the anonymity and unlinkability claims analyzed in subsequent sections.

6 Cryptographic Assumptions

6.1 IND CCA2 Security of ML-KEM

The ML-KEM used in AERN is assumed to satisfy indistinguishability under chosen ciphertext attack. The assumption is expressed through the standard IND CCA2 game played between a challenger and an adversary $\mathcal{A}$.

Definition 1 (IND CCA2 Game for ML-KEM)**.**

1. The challenger generates $(\text{kem.pk}, \text{kem.sk})$ and gives kem.pk to $\mathcal{A}$.
2. $\mathcal{A}$ may query a decryption oracle $\mathsf{Dec}(c)$ which returns $\text{KEM.Decaps}(c, \text{kem.sk})$, except for the challenge ciphertext.
3. $\mathcal{A}$ submits two candidate shared secrets (ss_0, ss_1).
4. The challenger encapsulates ss_b into a challenge ciphertext $c^\star$ and gives $(c^\star, ss_b)$ to $\mathcal{A}$.
5. $\mathcal{A}$ may continue to issue decryption queries, again excluding $c^\star$.
6. $\mathcal{A}$ outputs a bit b'.

The adversary wins if $b' = b$. Its advantage is

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA2}} = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

AERN assumes this advantage is negligible in the security parameter λ. This is the primary assumption supporting handshake secrecy, weak forward secrecy, and post compromise security.

6.2 EUF CMA Security of ML-DSA

The ML-DSA signature scheme is assumed to be existentially unforgeable under adaptive chosen message attack. This assumption underlies all authentication guarantees in AERN, including certificate validation and mutual authentication.

Definition 2 (EUF CMA Game for ML-DSA)**.**

1. The challenger generates $(\mathrm{pk}, \mathrm{sk})$ and gives pk to $\mathcal{A}$.
2. $\mathcal{A}$ may query a signing oracle $\mathsf{Sign}(m)$ which returns $\sigma = \mathrm{Sign}_{\mathrm{sk}}(m)$.
3. Eventually, $\mathcal{A}$ outputs a pair $(m^\star, \sigma^\star)$.

The adversary wins if $\mathrm{Verify}_{\mathrm{pk}}(m^\star, \sigma^\star) = 1$ and $m^\star$ was never submitted to the signing oracle. The advantage is

$$\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}} = \Pr[\mathcal{A} \text{ forges a valid signature}].$$

AERN assumes this advantage is negligible. All impersonation attacks on AERN reduce to breaking this assumption.

6.3 PRF Security of SHAKE Based KDF

The SHAKE based key derivation function used in AERN is modeled as a pseudo-random function keyed by the KEM shared secret ss. Its security is expressed using a standard PRF indistinguishability game.

Definition 3 (PRF Game for SHAKE Based KDF)**.**

1. The challenger samples a uniform random key k.
2. The adversary $\mathcal{A}$ gains oracle access to $\mathsf{KDF}(\cdot)$, defined as $\mathrm{KDF}(ss, \mathrm{ctx})$ for a fixed secret ss.
3. In the challenge phase, the challenger flips a random bit b. If $b = 1$, the oracle remains equal to the true KDF. If $b = 0$, the oracle returns uniform random strings of appropriate length.
4. $\mathcal{A}$ outputs a bit b'.

The adversary's advantage is

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

AERN assumes PRF security for the KDF. This assumption supports key separation, tunnel key pseudo-randomness, route seed generation, and per packet route uniformity.

6.4 INT CTXT Security of KMAC

KMAC is modeled as a message authentication code that satisfies integrity of ciphertext. An adversary cannot forge a valid authentication tag except with negligible probability.

Definition 4 (INT CTXT Game for KMAC)**.**

1. The challenger samples a secret MAC key K.
2. The adversary may query an oracle $\mathsf{Tag}(m)$ which returns $\mathrm{KMAC}(K, m)$.
3. The adversary outputs a pair $(m^\star, \tau^\star)$.

The adversary wins if

$$\mathrm{KMAC}(K, m^\star) = \tau^\star \quad \text{and} \quad m^\star \text{ was not queried to the oracle.}$$

Its advantage is

$$\mathsf{Adv}_{\mathcal{A}}^{\text{INT-CTXT}} = \Pr[\mathcal{A} \text{ forges a valid tag}].$$

This assumption supports tunnel integrity, replay protection, and route metadata authentication.

6.5 RCS Stream Cipher Pseudo-randomness

The RCS cipher is modeled as a pseudo-random keystream generator keyed with K_{enc}. AERN requires that no adversary can distinguish its keystream from uniform randomness.

Definition 5 (PRG Game for RCS)**.**

1. The challenger samples a symmetric key K_{enc}.
2. The adversary gains oracle access to:

$$\mathsf{KS}(n) = \text{first } n \text{ bits of } \mathrm{RCS}(K_{\text{enc}}).$$

3. In the challenge phase, the challenger flips a bit b. If $b = 1$, the oracle continues to return RCS keystream bits. If $b = 0$, it returns uniform random bit strings of equal length.
4. The adversary outputs a bit b'.

The advantage is

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

AERN assumes pseudo-randomness of RCS keystreams. This supports the confidentiality portion of tunnel security and underpins several hybrid transitions in the AEAD reduction.

7 Handshake Level Security

7.1 Correctness

Handshake correctness requires that whenever two honest parties complete a handshake session, they derive identical symmetric keys. Let C be a client and P be a proxy with certified KEM public key kem.pk_P. The client generates

$$(c, ss) \leftarrow \mathrm{KEM.Encaps}(\text{kem.pk}_P)$$

and sends c to P. By KEM correctness,

$$\mathrm{KEM.Decaps}(c, \text{kem.sk}_P) = ss.$$

Both parties compute

$$(K_{\text{enc}}, K_{\text{mac}}) = \mathrm{KDF}(ss, \text{context})$$

using identical context values defined by the certificate pair and authenticated handshake metadata. The KDF is deterministic. Therefore the keys match.
This completes the correctness argument.

7.2 Key Indistinguishability Reductions

Key indistinguishability requires that an adversary cannot distinguish the session keys derived in a completed handshake from uniform random strings. This security property is defined by the game $\mathsf{Game}_{\mathrm{KIND}}$ introduced in Section 3.

Theorem 1 (Key Indistinguishability)**.** *Let $\mathcal{A}$ be an adversary in $\mathsf{Game}_{\mathrm{KIND}}$. The advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{KIND}}$ is bounded by the IND CCA2 advantage of the ML-KEM and the PRF advantage of the SHAKE based KDF. Formally,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{KIND}} \leq \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA2}} + \mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRF}}.$$

Proof. The proof proceeds through a sequence of hybrid games.

Hybrid 0. The real key indistinguishability game. The adversary receives the real session keys.

Hybrid 1. Replace the KDF with a random oracle. If an adversary distinguishes Hybrid 0 from Hybrid 1, then it distinguishes the KDF from a pseudo-random function. By definition, this advantage is bounded by $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRF}}$.

Hybrid 2. Replace the KEM shared secret ss with a uniform random string. The only way the adversary can detect this change is by breaking the IND CCA2 security of ML-KEM. This difference in distinguishing probability is bounded by $\mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA2}}$.

Hybrid 3. Replace the final derived keys with uniform random strings. Since both the KDF and the shared secret are now random, the output is uniformly random from the adversary's perspective.
Hybrid 3 is identical to the challenger output in the indistinguishability experiment. The adversary's advantage is the sum of negligible differences across the transitions. This yields the stated bound. □

7.3 Mutual Authentication Reductions

Mutual authentication ensures that if an honest node accepts a peer identity during a handshake, then that peer must have generated a matching session unless the adversary forges a signature.

Theorem 2 (Mutual Authentication)**.** *Let $\mathcal{A}$ be an adversary in $\mathsf{Game}_{\mathrm{AUTH}}$. If an honest party accepts a peer identity without a matching session, then $\mathcal{A}$ has produced a forgery against ML-DSA. Hence,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{AUTH}} \leq \mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}.$$

Proof. Suppose the adversary succeeds in causing an honest node C to accept peer P without a matching session initiated by P. Acceptance requires that C verify a certificate C_P^{σ} and authenticated metadata produced by P.
For this to occur without an honest matching session, one of the following must hold.

1. The adversary created a forged certificate for P. This contradicts the EUF CMA security of ML-DSA because C_P^{σ} contains an ARS signature.
2. The adversary modified P's messages without detection. This also contradicts the EUF CMA security of ML-DSA because the handshake binds identity using signed certificates, and modifying certificate fields requires forging the ARS signature.
3. The adversary replaced P's ephemeral key material. This requires producing a new certified key binding in C_P^{σ}, again requiring a forged ARS signature.

All cases reduce to forging signatures. Therefore mutual authentication holds under the EUF CMA security of ML-DSA. □

7.4 Weak Forward Secrecy Reductions

Weak forward secrecy (wPFS) asserts that session keys remain secure even if long term keys are compromised after the session completes.

Theorem 3 (Weak Forward Secrecy)**.** *Let $\mathcal{A}$ be an adversary in* $\mathsf{Game}_{\mathrm{wPFS}}$.
If $\mathcal{A}$ compromises long term keys only after the challenge session completes, then

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{wPFS}} \leq \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA2}} + \mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRF}}.$$

Proof. In the wPFS game, the adversary observes all handshake transcripts and may later obtain long term keys. The adversary attempts to distinguish the challenge session key from random.
Once the handshake completes, the ephemeral shared secret ss is erased. Compromise of long term keys does not recover ss because the challenge ciphertext c cannot be decapsulated without violating IND CCA2 security.
Thus the same reduction sequence used in key indistinguishability applies, except that the adversary is additionally given the long term keys after the session. These keys provide no advantage in recovering ss due to erasure. Hence the privacy of the challenge session key relies only on KEM IND CCA2 and KDF PRF security. □

7.5 Post Compromise Security Reductions

Post compromise security (PCS) models compromise of a node followed by recovery and creation of a new session.

Theorem 4 (Post Compromise Security)**.** *Let $\mathcal{A}$ be an adversary in* $\mathsf{Game}_{\mathrm{PCS}}$. *After a compromise event and a subsequent recovery event that generates new KEM keys, the adversary's advantage in distinguishing the next session key from random is bounded by*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{PCS}} \leq \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CCA2}} + \mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRF}}.$$

Proof. After compromise, the adversary learns all long term keys and all
ephemeral material for active sessions. During recovery, the node regenerates a fresh ML-KEM key pair and wipes all compromised state.
The next session uses:

- a fresh public key kem.pk unknown to the adversary,
- new ephemeral randomness,
- a fresh KEM shared secret ss produced using kem.pk.

Even though the adversary retains old secrets, it cannot decapsulate the new ciphertext without violating IND CCA2 security. The derived keys are pseudo-random under the PRF security of the KDF.
Thus PCS holds with the stated bound. □

8 Channel Security and Replay Resistance

8.1 AEAD Tunnel Security Game

AERN provides per hop confidentiality and integrity through the composition of the RCS stream cipher and KMAC authentication. We model tunnel security using an authenticated encryption with associated data (AEAD) security game. The game captures both confidentiality (IND CPA like security) and integrity (INT CTXT like security) with respect to the adversary's ability to interact with tunnel oracles.

Definition 6 (AEAD Tunnel Security Game)**.** The AEAD game $\mathsf{Game}_{\mathrm{AEAD}}$ between a challenger and adversary $\mathcal{A}$ proceeds as follows.

1. The challenger samples a tunnel key pair $(K_{\mathrm{enc}}, K_{\mathrm{mac}})$ from the output of the KDF.
2. The adversary receives oracle access to:
   - $\mathsf{Enc}(M, \mathrm{Hdr})$ which increments the local sequence number, encrypts M under RCS, computes $\mathrm{Tag} = \mathrm{KMAC}(K_{\mathrm{mac}}, \mathrm{Hdr} \parallel \mathrm{Enc})$, and returns $(\mathrm{Hdr}, \mathrm{Enc}, \mathrm{Tag})$,
   - $\mathsf{Dec}(C)$ which verifies the tag, checks sequence and timestamp freshness, and returns the plaintext or the symbol $\perp$.
3. For confidentiality, the adversary submits (M_0, M_1) and receives a challenge ciphertext $C^\star$ computed from M_b under an unknown bit b.
4. For integrity, the adversary outputs a candidate packet $(\mathrm{Hdr}^\star, \mathrm{Enc}^\star, \mathrm{Tag}^\star)$ that did not originate from Enc.
5. The adversary wins the confidentiality game by guessing b. It wins the integrity game if $(\mathrm{Hdr}^\star, \mathrm{Enc}^\star, \mathrm{Tag}^\star)$ is accepted by Dec.

The AEAD advantage is defined as the sum of confidentiality and integrity advantages.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{AEAD}} = \mathsf{Adv}_{\mathcal{A}}^{\text{IND-CPA}} + \mathsf{Adv}_{\mathcal{A}}^{\text{INT-CTXT}}.$$

8.2 Confidentiality and Integrity Bounds

The following theorem states that AERN achieves AEAD security against all polynomial time adversaries under the assumptions stated in Section 5.

Theorem 5 (Per Hop AEAD Security)**.** *Let $\mathcal{A}$ make at most q_e encryption queries and q_d decryption queries. The AEAD advantage is bounded by*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{AEAD}} \leq \mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRG}} + \mathsf{Adv}_{\mathcal{A}}^{\text{INT-CTXT}} + \frac{q_e + q_d}{2^\ell},$$

where ℓ is the output length of the KMAC tag.

Proof. We use standard hybrid arguments.

Hybrid 1: Replace RCS with a random keystream generator. If an adversary distinguishes real RCS encryption from idealized random keystream encryption, it breaks the PRG assumption for RCS. This contributes at most $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRG}}$.

Hybrid 2: Replace KMAC with a random function keyed with K_{mac}.

If the adversary distinguishes Hybrid 1 from Hybrid 2 or forges a valid tag in Hybrid 2, it breaks the INT CTXT security of KMAC. This contributes at most $\mathsf{Adv}_{\mathcal{A}}^{\text{INT-CTXT}}$.

Hybrid 3: Treat authenticated ciphertexts as ideal AEAD outputs. In this hybrid, any ciphertext forgery attempt succeeds only if the adversary guesses a valid ℓ bit tag by chance. The probability of success per query is $2^{-\ell}$. Across $q_e + q_d$ adversarial interactions, the advantage is bounded by $(q_e + q_d)2^{-\ell}$.

Summing the negligible differences yields the stated bound. □

8.3 Replay Resistance Game and Proof

Replay resistance ensures that an adversary cannot cause an honest node to accept a packet that was previously transmitted, even if the adversary can reorder, delay, or duplicate traffic.

Definition 7 (Replay Resistance Game)**.** In $\mathsf{Game}_{\mathrm{Replay}}$, the adversary observes all tunnel packets and interacts with:

- Send which delivers packets to honest nodes,
- Enc which produces valid ciphertexts,
- CorruptLT and CorruptEphemeral for non challenge tunnels.

The adversary wins if an honest recipient accepts a packet that is stale with respect to its sequence number or timestamp window.

Theorem 6 (Replay Resistance)**.** *AERN is replay resistant. Any successful replay attack requires a tag forgery or a violation of header metadata integrity. Therefore,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{Replay}} \leq \mathsf{Adv}_{\mathcal{A}}^{\text{INT-CTXT}}.$$

Proof. Consider a replayed packet (Hdr, Enc, Tag).

Case 1: The adversary replays the exact packet. The sequence number embedded in Hdr is stale. Honest recipients reject stale sequence numbers. The adversary cannot modify Hdr without recomputing Tag, which contradicts KMAC integrity.

Case 2: The adversary modifies the header. If the adversary changes timestamps, sequence numbers, or hop indices, the tag no longer matches, unless the adversary produces a valid forgery. This contradicts INT CTXT security.

Case 3: The adversary forges a new ciphertext. The only way this succeeds is by forging a valid tag. The probability is exactly the INT CTXT advantage.

Thus replay resistance reduces to the integrity of KMAC. □

8.4 Packet Size Uniformity Lemma

AERN enforces a fixed packet size to eliminate traffic analysis vectors based on packet lengths.

Lemma 1 (Packet Size Uniformity)**.** *Let all AERN packets be padded to a fixed length L before transmission. Then for any adversary $\mathcal{A}$ observing ciphertexts across the network, the packet size leaks no information about plaintext length, routing choices, or session state. Formally,*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{SizeLeak}} = 0.$$

Proof. All packets transmitted across all links have identical length L by construction. The packet format

$$(\mathrm{Hdr}, \mathrm{Enc}, \mathrm{Tag})$$

contains no length dependent fields outside the fixed size payload region, and padding is indistinguishable from ciphertext under the PRG security of RCS. Therefore the adversary receives no distributional difference between packets carrying short plaintexts, long plaintexts, or padding.

Consequently, the plaintext length distribution and any structural metadata encoded in packet size are perfectly hidden. The adversary gains no advantage in distinguishing any property correlated with packet length. □

9 Anonymity and Traffic Analysis Resistance

9.1 Anonymity Game for AERN

Anonymity in AERN is defined as the inability of an adversary with global network visibility and partial node compromise capabilities to distinguish which client destination pair generated a given traffic flow. This property is captured by the indistinguishability game $\mathsf{Game}_{\mathrm{ANON}}$.

Definition 8 (Anonymity Game)**.** The adversary $\mathcal{A}$ selects two possible communication scenarios:

$$(C_0, D_0) \quad \text{and} \quad (C_1, D_1).$$

The challenger samples a random bit b and initiates a session between C_b and D_b. The adversary may:

- observe all encrypted traffic across all links,
- compromise any subset of interior proxies,
- query $\mathsf{CorruptLT}$ or $\mathsf{CorruptEphemeral}$ on non endpoint nodes,
- issue adaptive queries to Send to influence scheduling or routing patterns.

The adversary outputs b'. The anonymity advantage is

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

AERN achieves anonymity if this advantage is negligible whenever at least one endpoint of each route remains honest.

The analysis below proves anonymity under endpoint honesty assumptions and describes how anonymity degrades under partial compromise.

9.2 k Anonymity Bounds under Partial Compromise

A more refined anonymity measure considers not only indistinguishability between two choices, but also adversarial uncertainty across a set of k possible source destination pairs. Let $\mathcal{S} = \{(C_1, D_1), \ldots, (C_k, D_k)\}$ be a set of candidate pairs. The adversary selects a challenge subset and attempts to identify which pair generated the traffic.

Definition 9 (k Anonymity Game)**.** The challenger chooses a random index $b \in \{1, \ldots, k\}$ and initiates communication between (C_b, D_b). The adversary interacts with the network exactly as in $\mathsf{Game}_{\mathrm{ANON}}$ and outputs a guess b'. The adversarial advantage is

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{kAnon}} = \Pr[b' = b] - \frac{1}{k}.$$

The system achieves k anonymity if this advantage is negligible.

Theorem 7 (Anonymity Bound under Partial Compromise)**.** *Let $\mathcal{A}$ compromise a set of interior nodes $\mathcal{C}$ but not the entry or exit nodes for the challenge route. Then*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{kAnon}} \leq \mathrm{negl}(\lambda).$$

If the adversary controls c proxies, the advantage is bounded by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{kAnon}} \leq \frac{c}{|\mathsf{ProxySet}|} + \mathrm{negl}(\lambda),$$

reflecting the probability that a random route intersects adversarial nodes.

Proof Sketch. If either endpoint remains honest, the adversary does not learn the innermost or outermost layer of onion encryption. Every ciphertext segment observed at compromised hops is computationally indistinguishable from random due to RCS pseudo-randomness and KMAC integrity.
The adversary's only link to the session is through the sequence of compromised interior nodes. The probability that a random route intersects any of the c compromised nodes is $c/|\mathsf{ProxySet}|$ because route generation is uniform over all proxies. If the route avoids all compromised nodes, the adversary learns no distinguishing information. Thus, the adversary's distinguishing power is at most $c/|\mathsf{ProxySet}|$ plus negligible terms from hybrid transitions used to argue ciphertext indistinguishability.
This bound implies full k anonymity when c is small relative to the proxy population. □

9.3 Uniform Route Selection Lemma

AERN derives route seeds using a SHAKE based KDF seeded with fresh client randomness. Routes are then generated deterministically from the seed. The anonymity reduction relies on the assumption that all valid routes are equally likely.

Lemma 2 (Uniform Route Selection)**.** *Let s_r be a route seed derived as $s_r = \mathrm{SHAKE}(r_C)$ with r_C chosen uniformly at random. Let $(P_1, \ldots, P_h) =$*
$\mathrm{RouteGen}(s_r, \mathsf{TState})$ *be the route of length h. Then for any two valid routes R and R' of the same length,*

$$\Pr[\mathrm{RouteGen}(s_r) = R] = \Pr[\mathrm{RouteGen}(s_r) = R'].$$

Proof. SHAKE is modeled as a pseudo-random function in the PRF sense. Therefore s_r is computationally indistinguishable from a uniform random string.
Since RouteGen is deterministic and its input is a uniform seed, the output distribution is the image of a uniform input passed through a deterministic mapping. If the mapping treats all candidate proxies symmetrically, each route of a given length appears with equal probability.
More precisely, for each route R there exists a set of seeds S_R mapping uniquely to R. Because SHAKE output is uniform over its domain, and the sets S_R have equal cardinality by construction of the selection algorithm, the resulting distribution over routes is uniform. This uniformity guarantees unbiased path selection required for anonymity reductions. □

9.4 Correlation Resistance Analysis

Traffic correlation attacks attempt to match patterns of packets entering the network with packets leaving it. AERN mitigates such attacks through several structural mechanisms.

Fixed size ciphertexts. All packets are padded to a fixed length, preventing correlation through size or fragmentation patterns.

Per packet randomized routes. Each packet is independently assigned a fresh route. Even if two packets belong to the same logical flow, they traverse statistically independent paths. The correlation probability across independent routes decreases rapidly with increasing hop count.

Hop by hop authenticated encryption. Each hop removes only one layer of encryption. A compromised node sees only its local ciphertext layer and hop index. It cannot link this to any incoming ciphertext except by forward direction deterministic routing, which reveals no information about the origin.

Endpoint honesty assumption. If either the entry or exit node remains honest, the adversary never sees the entire encryption onion. It cannot correlate ingress and egress ciphertexts without solving the anonymity game.

Timing disturbance. Because adversarially controlled nodes do not see the full route, they cannot rely on inter packet timing to correlate flows. Each interior hop introduces independent scheduling delays, and clients can introduce voluntary jitter. This breaks timing correlation except under fully adversarial route compromise.

Combined effect. Together, these mechanisms ensure that the adversary's ability to correlate flows across the network is limited by route compromise probability. When at least one endpoint is honest, the adversary gains negligible information. When routes are long relative to compromised nodes, the attacker can correlate at most a fraction proportional to the ratio $c/|\mathsf{ProxySet}|$.

10 Systemic Risks and Trust Anchor Modeling

10.1 Centralized Trust Model

AERN relies on a centralized certificate anchored trust structure. The ARS serves as the global trust anchor, and the ADC distributes the authoritative topology and certificate list. This design simplifies authentication and prevents the uncontrolled growth of self signed or unverified routing information. However, it also introduces concentration of trust. To analyze this structure formally, we define the trust anchor set

$$\mathcal{T} = \{\mathrm{ARS}, \mathrm{ADC}\}.$$

Each trust anchor holds:

- ARS: a long term ML-DSA signing key that certifies all device identities,
- ADC: the authoritative topology and certificate list, together with a distribution mechanism.

Security of the entire system depends on the correctness and integrity of $\mathcal{T}$. A compromise or malfunction of either trust anchor affects the global view of the network. Because trust is centralized, security proofs must include adversarial models in which the ARS or ADC becomes unavailable, behaves incorrectly, or behaves maliciously.

10.2 ARS or ADC Offline Model

Temporary unavailability of trust anchors is an important operational scenario. The ARS is typically offline except during certificate generation, and the ADC may become temporarily unreachable during network events. We model offline behavior using the following assumptions.

Definition 10 (Offline Trust Anchor Model)**.** A trust anchor is offline if it does not respond to certificate or topology requests, but its previously signed data remains available to nodes. Nodes continue to use:

$$\mathsf{CertList}, \quad \mathsf{NodeList}, \quad \mathsf{Adjacency},$$

from the last authenticated topology distribution.

Security implications are as follows.

Certificate validation. Nodes can still verify certificates because all signatures remain valid under $\mathrm{pk}_{\mathrm{ARS}}$.

Topology stability. Topology information remains stable as long as no node requires new certificates or revocations. The protocol continues to operate securely without additional trust anchor interaction.

Handshake security. Handshake security does not degrade because the ML-KEM handshake and the authenticated tunnel mechanisms do not rely on live trust anchor activity.
Thus, offline behavior does not break cryptographic security. It only delays updates, which may affect liveness but not confidentiality or authentication.

10.3 Malicious ARS or ADC Model

A more challenging scenario arises when a trust anchor becomes adversarial. We model this by allowing the adversary to obtain the signing key of the ARS or to fully control ADC behavior.

Malicious ARS. If the adversary learns $\mathrm{sk}_{\mathrm{ARS}}$, it can create arbitrary certificates. The consequences are:

- Node authentication can no longer be trusted,
- Any entity can be impersonated,
- New certificates cannot be validated.

However, the adversary cannot retroactively decrypt ciphertexts protected by previously established tunnels due to forward secrecy, as shown below.

Malicious ADC. If ADC is malicious, it may distribute incorrect or adversarial topology information. The consequences are:

- Routing may be biased toward compromised proxies,
- Clients may be misled into selecting unsafe routes,
- Topology consistency may be lost.

However, even a malicious ADC cannot forge or modify certificates without breaking ML-DSA, since all certificates must still verify under $\mathrm{pk}_{\mathrm{ARS}}$.

Theorem 8 (Trust Anchor Limitation)**.** *Let $\mathcal{A}$ fully control ADC or ARS. If $\mathcal{A}$ does not break ML-DSA and does not interfere during the KEM handshake, then it cannot retroactively compromise past session keys or decrypt past ciphertexts.*

Proof. The handshake keys depend solely on:

$$ss = \text{KEM.Decaps}(c, \text{kem.sk}_P).$$

The ARS and ADC do not know kem.sk_P unless it is compromised directly. Certificates do not contain shared secrets. Past ciphertexts are protected by RCS keystreams derived from ss, and ss is erased after handshake completion.
Even if ARS or ADC keys are compromised, no information about the decapsulation secret ss is revealed. The adversary gains influence over future trust relationships but cannot decrypt existing ciphertexts.
Thus past sessions remain secure. □

10.4 Limits Imposed by Forward Secrecy

Forward secrecy prevents retroactive compromise of previously established tunnel keys even if long term keys are later exposed. AERN provides weak forward secrecy because session keys are derived from ephemeral KEM secrets that are erased after tunnel establishment. Let ss be the shared secret derived in a completed session. Forward secrecy is achieved if:

$$\text{sk}_{\text{ARS}}, \text{sk}_{\text{ADC}}, \text{sk}_P, \text{sk}_C \quad \text{do not reveal ss after erasure.}$$

Recovery after ARS compromise. Even with complete access to ARS secrets, the adversary cannot derive ss because the ARS does not participate in the KEM handshake.

Recovery after ADC compromise. Topology manipulation does not yield any information about ss and does not influence existing cipher states.

Recovery after proxy compromise. If a proxy is compromised after a session completes and after ss has been erased, the adversary cannot reconstruct ss or derive past keys.

Limit of forward secrecy. Forward secrecy does not protect active sessions. If a node is compromised during an active tunnel, the cipher state and derived keys are revealed. This limitation is inherent to all symmetric tunnel based protocols.

Theorem 9 (Forward Secrecy Limit)**.** *Forward secrecy in AERN holds only for completed sessions. If a node is compromised during an active tunnel, the adversary may decrypt all future traffic until re keying or recovery, but cannot decrypt past ciphertexts.*

Proof. During an active session, ephemeral secrets are present in memory. Compromise reveals:

$$ss, K_{\text{enc}}, K_{\text{mac}}, \sigma_{\text{state}}.$$

This allows decryption of future packets. However, once the session completes and all ephemeral secrets are erased, none of these values remain accessible. Past ciphertexts require ss or the corresponding keystream, which are no longer present. Therefore passive retroactive decryption is impossible. □

Together, these results demonstrate that AERN provides strong protection
against trust anchor failure for past sessions, while revealing predictable limitations for active sessions.

11 Conclusion

11.1 Summary of Formal Results

This work presented a complete and rigorous formal cryptanalysis of the Authenticated Encrypted Relay Network. The analysis introduced a unified game based framework and derived explicit reductions for all major security properties of AERN. We established handshake correctness, key indistinguishability, mutual authentication, weak forward secrecy, and post compromise security under the IND CCA2 assumption for the ML-KEM and the EUF CMA assumption for the ML-DSA signature scheme.
We proved that each tunnel hop achieves authenticated encryption security through the combined pseudo-randomness of the RCS stream cipher and the integrity guarantees of KMAC. The reductions show that the adversary's advantage is bounded by the security of these underlying primitives, along with negligible probability events associated with random tag collisions.
We introduced a formal anonymity game and extended it to a k anonymity setting that quantifies anonymity degradation when the adversary compromises some but not all nodes in the network. We proved that AERN achieves negligible adversarial advantage when at least one endpoint of each route remains honest and derived explicit bounds on anonymity under interior node compromise. A route uniformity lemma demonstrated that the route selection mechanism provides unbiased path generation, which is essential for unlinkability and resistance to traffic analysis.
Finally, we presented formal models for trust anchor behavior and analyzed systemic risks associated with ARS or ADC compromise. We showed that offline trust anchors do not compromise cryptographic security and that malicious trust anchors cannot break confidentiality of past sessions due to the weak forward secrecy provided by the KEM based handshake. These results yield a complete formal foundation for trust analysis in the AERN architecture.

11.2 Implications for Deployment

The formal results demonstrate that AERN provides strong protection against both classical and quantum adversaries and supports secure multi hop encrypted communication within a centrally managed topology. The certificate anchored trust structure simplifies authentication and enhances control over routing behavior, which is valuable in managed environments where security guarantees are prioritized over decentralization.
In practical deployments, the guarantees derived from this analysis depend on several operational requirements. Nodes must implement strict erasure procedures for ephemeral secrets to preserve forward secrecy. KEM and signature algorithms must be implemented with constant time behavior and without side channel leakage. ADC updates must be authenticated and timestamped to prevent rollback attacks or topology desynchronization. Fixed size packet padding and per packet route generation must be enforced consistently to maintain anonymity guarantees.
These considerations reflect the boundary between formal security and real world engineering. If implemented carefully, AERN provides strong confidentiality, integrity, and anonymity guarantees even when confronted with adversaries with global visibility or partial compromise capability.

11.3 Future Work

There are several directions for extending this analysis. One important direction is the development of more detailed models for traffic analysis under realistic network conditions, including congestion, scheduling variability, and adversarial delay manipulation. Simulation

based validation of anonymity guarantees across large scale deployments would provide insight into the interaction between topology structure and anonymity bounds.

Additional work may explore threshold cryptographic mechanisms for the ARS to reduce risks arising from trust centralization. Formal leakage models for ML-KEM, RCS, and KMAC implementations would strengthen the analysis by quantifying resilience against side channel attacks. Further study of the interaction between packet timing, route length, and link layer behavior could produce stronger correlation resistance guarantees.

Future revisions of AERN may incorporate post quantum authenticated key exchange mechanisms or integrate alternative KEM and signature schemes with improved performance or reduced key sizes. These developments would expand the protocol's efficiency without compromising analytical strength.

The results presented here establish a formal foundation for the security of AERN and identify the assumptions required for robust real world deployment. Continued refinement of the protocol and its underlying primitives will further strengthen its long term viability as a quantum resistant anonymity solution.

References

1. National Institute of Standards and Technology (NIST). *FIPS 202: SHA 3 Standard.* U.S. Department of Commerce, 2015. Available at: https://doi.org/10.6028/NIST.FIPS.202.
2. National Institute of Standards and Technology (NIST). *SP 800 185: SHA 3 Derived Functions.* U.S. Department of Commerce, 2016. Available at: https://doi.org/10.6028/NIST.SP.800-185.
3. National Institute of Standards and Technology (NIST). *FIPS 203: Module Lattice Based Key Encapsulation Mechanism Standard.* U.S. Department of Commerce, 2024. Available at: https://csrc.nist.gov.
4. National Institute of Standards and Technology (NIST). *FIPS 204: Stateless Hash Based Digital Signature Standard.* U.S. Department of Commerce, 2024. Available at: https://csrc.nist.gov.
5. Bertoni, G., Daemen, J., Peeters, M., and Van Assche, G. *The Keccak Reference.* NIST SHA 3 Competition Submission, 2011. Available at: https://keccak.team/files/Keccak-reference-3.0.pdf.
6. Rogaway, P., and Shrimpton, T. *A Provable Security Treatment of the Authenticated Encryption Problem.* ACM Conference on Computer and Communications Security, 2006. Available at: https://doi.org/10.1145/1180405.1180427.
7. Bellare, M., and Namprempre, C. *Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm.* IACR Cryptology ePrint Archive, 2000. Available at: https://eprint.iacr.org/2000/025.
8. Goldreich, O. *Foundations of Cryptography, Volume 1: Basic Tools.* Cambridge University Press, 2001. Available at: https://www.cambridge.org.
9. Danezis, G., Diaz, C., Troncoso, C., and Serjantov, A. *Impact of Network Topology on Anonymity and Routing Metrics.* Technical Report, 2009. Available at: https://arxiv.org/abs/0901.0325.
10. Syverson, P., Dingledine, R., and Mathewson, N. *Onion Routing Revisited.* IEEE Security and Privacy, 2004. Available at: https://doi.org/10.1109/MSECP.2004.1281240.
11. Underhill, J.G. *Authenticated Encrypted Relay Network (AERN) Specification.* Quantum Resistant Cryptographic Solutions Corporation, 2025. Available at: https://www.qrcscorp.ca/documents/aern_specification.pdf.
12. QRCS Corporation. AERN Reference Implementation (Source Code Repository). https://github.com/QRCS-CORP/AERN