Resonance Protocol:
The Trust Contract Framework

Abstract

This paper presents the design, implementation model, and formal definition of the Trust Contract Framework within the Resonance Protocol (RP). Building upon the concepts introduced in the prior whitepapers, *The Subsumption Hive Model for Ransomware Defence* and *RP Architectural Whitepaper*, this work defines the mechanism by which Subsumption Hives establish, verify, and maintain cryptographically authenticated communication channels through dynamically generated Trust Contract Certificates (TCCs).

The Trust Contract Framework provides a deterministic method for maintaining verifiable, state-bound trust between autonomous digital systems. Each hive cryptographically attests to its internal integrity using Merkle ancestry. When two hives initiate contact, a Handshake Exchange Sequence (HES) verifies both integrity states before generating a state-dependent contract. This contract governs subsequent encrypted communications, which automatically invalidate upon tampering or configuration drift, resulting in autonomous isolation.

Introduction

The Resonance Protocol (RP) extends zero-trust architecture by coupling identity verification with device integrity attestation. In traditional zero-trust environments, communication between systems is authenticated based on credentials and policies, yet assumes static system integrity. The RP removes this assumption, requiring each hive to demonstrate not only valid credentials but also verifiable internal consistency before participating in the network.

The Trust Contract Framework operationalises this concept. It defines a dynamic, measurable trust state between Adjacent Hives, replacing implicit assumptions of security with mathematically verifiable conditions.

Through the TCC mechanism, RP establishes peer-to-peer channels in which the cryptographic keys are derived from, and dependent upon, the verified state of the communicating hives. The moment a hive's state changes, its communication context becomes invalid, causing automatic contract revocation.

Related Work and Context

The Subsumption Hive Model introduced in earlier RP publications describes systems as layered entities, hardware, firmware, kernel, process, and application, where each layer verifies the one beneath it. Integrity proofs are consolidated via Merkle ancestry, producing a single Merkle root that represents the hive's complete internal integrity.

This whitepaper focuses on the next stage: how these verified hives interact securely. While conventional attestation models (e.g., TPM-based remote attestation, IETF RATS) focus on one-way verification, the Trust Contract Framework introduces bidirectional, state-linked trust relationships that form, maintain, and revoke autonomously.

System Overview

Each Subsumption Hive maintains a self-contained record of its components, stored as Merkle trees under /hives/<hive_name>/. The Merkle root acts as a condensed fingerprint of the hive's integrity.

When two adjacent hives attempt communication, the following sequence occurs:

1. Both hives must be in *ready-active* state, signalling verified integrity.
2. The initiating hive transmits a Handshake Initiation Frame (HIF), containing its identifier, nonce, and signed Merkle root.
3. The recipient responds with its own Handshake Response Frame (HRF) containing equivalent data.
4. Each hive independently verifies the other's attestation.
5. Upon mutual validation, a Trust Contract Certificate (TCC) is generated, cryptographically binding both Merkle roots.

Trust Contract Formation

A Trust Contract Certificate (TCC) is a data structure that defines the cryptographic and logical relationship between two validated hives. It contains:

Field	**Description**
issuer_id	Unique UUID of originating hive
recipient_id	UUID of recipient hive
issuer_merkle_root	Latest Merkle root of issuer hive
recipient_merkle_root	Latest Merkle root of recipient hive
valid_from	Epoch timestamp of contract creation
valid_until	Expiry or revocation timestamp
scope	Permitted communication domains
signature_issuer	Digital signature of issuer
signature_recipient	Digital signature of recipient

Cryptographic Derivation

The contract's cryptographic strength is derived from the inclusion of both hives' Merkle roots as contextual data within the key exchange process. The shared secret is established as:

$$shared\_secret \ = \ HKDF(ECDH(a\_private, b\_public)$$

$$info = (root\_A \,||\, root\_B \,||\, nonce\_A \,||\, nonce\_B))$$

This ensures that encryption keys are valid only when both hives maintain their verified internal states. Any change to a hive's Merkle root immediately invalidates the session's derived key material.

Handshake and Cryptographic Design

Handshake Exchange Sequence (HES)

The handshake process (HES) consists of the following stages:

1. **Integrity Announcement:**
   Each hive broadcasts readiness via multicast or controller registry, including a signed attestation hash.
2. **Nonce Exchange:**
   Both hives exchange nonces to guarantee freshness and prevent replay.
3. **Mutual Attestation Verification:**
   Each hive validates the received Merkle root against its registry of trusted states.
4. **Shared Secret Derivation:**
   An ECDH exchange is performed, incorporating Merkle context into HKDF.
5. **TCC Generation:**
   Both hives sign and exchange the final Trust Contract Certificate.
6. **Session Establishment:**
   Encrypted channels (TLS/IPSec) are negotiated using derived session keys.

Time Constraints

Each TCC is designed to be short-lived. Default lifetime recommendations are ≤15 minutes for volatile systems and ≤1 hour for stable devices. Renewal requires re-attestation.

State-Bound Communication and Verification Logic

All data transmission between hives occurs under an IPSec or TLS tunnel using AEAD ciphersuites (e.g., AES-GCM, ChaCha20-Poly1305).

Every outbound packet includes a state validation token, containing:

- session_id
- contract_id
- merkle_digest_current

The receiving hive validates that the Merkle digest matches the last attested root associated with the active TCC. If it differs, the packet is discarded, and the session enters degraded verification mode.

Contract Lifecycle and Revocation Mechanisms

Normal Lifecycle

1. **Creation:** Generated upon successful handshake.
2. **Active:** Used for session encryption and validation.
3. **Refresh:** Revalidated automatically before expiry if no integrity deltas exist.
4. **Expiry:** Contract expires and is renewed via a new handshake.

Revocation Conditions

A TCC is revoked under the following conditions:

- Integrity delta detected (Merkle mismatch).
- Heartbeat timeout beyond tolerance threshold.
- Explicit revocation broadcast from Controller Hive.
- Signed notification from peer hive marking itself unhealthy.

Upon revocation:

1. All sessions tied to the TCC are dropped.
2. Session keys are securely deleted.
3. Adjacent hives mark each other as quarantined until successful re-attestation.

Failure Handling and Recovery

When a hive fails integrity verification:

1. It transitions to Unhealthy State.
2. All adjacent contracts are invalidated.
3. Logs are written to local immutable audit storage.
4. The hive attempts self-repair (rollback or restoration).
5. Post-recovery, a Re-attestation Sequence (RAS) occurs to obtain new contracts.

Recovery policies can be centrally managed by a Controller Hive or locally autonomous depending on deployment topology.

Security Analysis

The Trust Contract Framework provides several inherent security guarantees:

Property	Description
State-linked encryption	Keys are only valid for verified system states.
Automatic isolation	Tampered hives are instantly disconnected.
Replay protection	Nonce and timestamp enforcement prevent old contract reuse.
Forward secrecy	Ephemeral keys prevent retroactive decryption.
Auditability	Every contract issuance and revocation is logged cryptographically.

Attack Surface Considerations

- **DoS Resistance:** Integrity validation includes a grace window to avoid disconnect storms.
- **Rollback Protection:** Historical Merkle ancestry is tracked to detect reversion attempts.
- **Privacy:** Only blinded Merkle roots are shared; component hashes remain internal.

Implementation Considerations

A reference implementation uses:

- **ECDH (Curve25519)** for key exchange.
- **HKDF-SHA512** for key derivation.
- **AES-GCM-256** for symmetric encryption.
- **Ed25519** for contract signing.

TCCs are serialised as JSON or CBOR objects, signed using detached signatures. Implementations must use monotonic clocks for expiry enforcement and store revocation data in an append-only ledger for audit consistency.

Conclusion

The Trust Contract Framework operationalises the principle of *verifiable integrity as a prerequisite for communication*. Through its use of state-linked cryptography, short-lived attestation, and autonomous revocation, it establishes a foundational shift in how distributed systems maintain trust.

By embedding integrity verification into the communication layer itself, the Resonance Protocol eliminates the separation between system health and security posture, ensuring that trust is continuously earned, never assumed.

References

1. Collinson, L. *Resonance Protocol: Technical Subsumption Hive Ransomware Defence*. Cyber Security Stack, 2024.
2. Collinson, L. *Resonance Protocol: Architectural Whitepaper*. Cyber Security Stack, 2024.
3. IETF RATS Working Group. *Remote Attestation Procedures Architecture (RFC 9334)*. 2023.
4. NIST. *Zero Trust Architecture (SP 800-207)*. 2020.
5. IETF. *The Internet Key Exchange Protocol Version 2 (RFC 7296)*. 2014.