UniKey Core Specification

Status: Informational (Normative Invariants)
Category: Trust Model and Conformance
Audience: Implementers, network operators, security architects, decision-makers
Version: 1.0
Date: February 2026

Abstract

This document defines the UniKey trust model. UniKey is a universal method for proving identity, authority, and permission before actions occur on the internet. It is designed for humans, devices, software agents, and AI agents operating across open and untrusted environments.

UniKey does not rely on assumptions, stored sessions, shared secrets, or platform trust. Instead, it relies on cryptographically verifiable proof evaluated before execution.

This document defines what UniKey is, what must never change, and what it means to be UniKey-conformant.

---

1. Introduction

“Trust” is used constantly in technology. We say “trusted users,” “trusted devices,” “trusted networks,” and “zero trust.” Yet most systems do not clearly define what trust means.

In practice, trust is often treated as prior login, network location, session existence, platform ownership, or historical behavior. These are assumptions, not proof.

UniKey begins from a different premise: trust is not a feeling or a memory. Trust is knowledge that can be proven.

UniKey defines a model where trust must be demonstrated, not assumed; trust must be provable using mathematics; trust must be evaluated at the moment of action; and trust must fail safely.

---

2. Scope

This specification defines the core invariants of UniKey. An invariant is a rule that must always hold true regardless of implementation details, deployment environment, business model, cryptographic choices, or product design.

If these invariants are violated, the system is not UniKey, even if similar terminology is used.

These invariants define what trust is, how trust is proven, when trust is evaluated, and what happens when trust fails.

2.1 What This Specification Does Not Define

This document intentionally does not define user experiences, login flows, UI patterns, business logic, commercial models, or specific cryptographic algorithms. Those choices are left to implementations.

This separation is deliberate. It keeps UniKey universal, vendor-neutral, and future-proof.

2.2 Purpose of This Document

The purpose of this document is to define the unchanging foundation of UniKey.

Implementations may evolve.
Products may change.
Cryptography may advance.

These rules do not.

---

3. Terminology

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL are to be interpreted as described in RFC 2119.

3.1 Agent

An Agent is any entity capable of initiating an action. This includes humans, software processes, automated services, devices, and AI agents.

UniKey treats all agents the same. What matters is not what the agent is, but whether it can prove it is allowed to act.

3.2 Device

A Device is an execution or approval environment. Devices matter because they can enforce local control, confirm presence, and bind actions to physical reality.

Devices do not grant trust by themselves. Trust still requires proof.

3.3 Issuer

An Issuer creates signed statements of truth. Issuers may be domain owners, enterprises, email providers, or other trusted authorities.

Issuers do not approve actions. They assert facts cryptographically.

3.4 Verifier

A Verifier evaluates proof and makes an allow or deny decision. A Verifier validates signatures, confirms issuer trust, evaluates authority, checks time bounds, and matches authority to the requested action.

Verifiers do not guess.

---

4. UniKey Core Invariants (Normative)

All UniKey-conformant implementations MUST uphold every invariant in this section. These invariants are stable and define UniKey.

4.1 Trust Is Knowledge, Not Assumption

Definition: In UniKey, trust means verifiable knowledge that an actor is allowed to perform an action. Trust MUST NOT be based on prior login, network location, session existence, platform ownership, or historical behavior.

Rationale: Assumptions fail silently. When trust is assumed, systems allow actions before verification, detect abuse after damage, and cannot clearly explain authorization decisions. UniKey replaces assumption with proof.

4.2 Trust Must Be Cryptographically Provable

Definition: All trust claims in UniKey MUST be backed by cryptographic proof. Proof MUST be verifiable by independent parties, based on asymmetric cryptography, and resistant to forgery.

Rationale: Cryptography is mathematical, not contextual. Asymmetric cryptography allows anyone to verify a claim while only the holder of a private key can create it. This makes trust objective, deterministic, and machine-verifiable.

4.3 Identity, Authority, and Action Are Separate

Definition: UniKey treats identity, authority, and action as distinct concepts. Identity answers who is acting. Authority answers what is allowed. Action answers what is being requested now. No single proof MAY substitute for all three.

Rationale: Most systems combine these concepts, leading to over-permission, ambiguous failure modes, and increased blast radius. Separation limits damage and clarifies intent.

4.4 Verification Occurs Before Action

Definition: All trust verification MUST complete before an action is executed. Verification after execution MUST NOT be considered valid UniKey behavior.

Rationale: Reactive security detects problems after damage occurs. UniKey prevents damage instead of responding to it.

4.5 UniKey Is Stateless

Definition: UniKey MUST NOT rely on stored sessions or remembered trust. Each request MUST carry sufficient proof to be verified independently.

Rationale: State creates hidden trust. Hidden trust is difficult to audit, easy to steal, and hard to scale across systems.

4.6 Authority Is Explicit, Limited, and Revocable

Definition: Authority in UniKey MUST be explicitly granted, limited in scope, limited in time, and revocable or allowed to expire.

Rationale: Broad authority creates silent failure. Limited authority contains mistakes and localizes compromise.

4.7 Trust May Travel with the Agent

Definition: UniKey allows trust to move with an agent across systems, provided proof remains valid. Trust MUST NOT be bound exclusively to a single application, platform, or account system.

Rationale: Agents operate across boundaries. Portable trust enables federated interaction.

4.8 Failure Must Be Safe and Explicit

Definition: When verification fails, the default behavior MUST be denial. UniKey MUST NOT guess, degrade to weaker security, or fall back to implicit trust.

Rationale: Security failures should be explicit. Failure must deny.

4.9 Summary of Invariants

UniKey is defined by these rules: trust is knowledge; proof replaces assumption; identity, authority, and action are separate; verification precedes action; state is avoided; authority is limited; trust is portable; failure denies. Everything else is implementation detail.

---

5. Conformance (Normative)

This section defines what it means for a system to be UniKey-conformant.

A system MAY use different transports, cryptographic algorithms, deployment models, or user experiences. However, a system MUST satisfy all core invariants defined in Section 4 to claim UniKey compatibility.

5.1 Conformance Requirements

A UniKey-conformant implementation MUST require cryptographically verifiable proof for every action; separate identity, authority, and action; verify all proof before execution; operate without stored trust state; enforce explicit, limited, and revocable authority; and deny actions safely and explicitly on failure.

Failure to meet any requirement breaks conformance.

5.2 What Conformance Does Not Require

A UniKey-conformant system does not require a specific user interface, login flow, cryptographic algorithm, centralized service, or vendor. Conformance is architectural, not commercial.

5.3 Partial Adoption

Partial adoption MUST NOT be described as full UniKey conformance. UniKey conformance is binary.

---

6. Threat Model Mapping (Guidance)

This section maps UniKey's invariants to the classes of failure they are designed to eliminate.

6.1 Impersonation

Threat: An attacker pretends to be a legitimate actor.
Mitigated by: 4.1 and 4.2.

6.2 Session Hijacking

Threat: An attacker reuses a stolen session or token.
Mitigated by: 4.5 and 4.4.

6.3 Over-Permission

Threat: An actor is allowed to do more than intended.
Mitigated by: 4.3 and 4.6.

6.4 Replay Attacks

Threat: Previously valid proof is reused.
Mitigated by: 4.4 and 4.6.

6.5 Silent Failure

Threat: A system fails open or degrades security.
Mitigated by: 4.8.

6.6 Platform Lock-In

Threat: Trust is trapped inside one ecosystem.
Mitigated by: 4.7.

---

7. Non-Goals and Anti-Patterns (Normative)

7.1 Non-Goal: Defining User Experience

UniKey does not define interfaces or workflows. It defines truth, not presentation.

7.2 Non-Goal: Replacing Applications

Applications decide what should happen. UniKey decides whether it is allowed to happen.

7.3 Non-Goal: Centralizing Control

UniKey does not require a central authority or single vendor.

7.4 Anti-Pattern: Session-Based Trust

Remembered trust violates stateless verification.

7.5 Anti-Pattern: Implicit Authority

Identity alone MUST NOT imply permission.

7.6 Anti-Pattern: Verify-After-Execution Security

Security checks after execution are not UniKey behavior.

7.7 Anti-Pattern: Fallback to Weaker Security

Failure MUST NOT degrade security.

7.8 Anti-Pattern: Platform-Bound Trust

Trust MUST NOT be limited to a single platform.

---

8. Reference Conformance Checklist (Informational)

A UniKey-conformant system should be able to answer yes to all of the following: Does every action require proof? Is identity separate from authority? Is authority explicit and time-limited? Is verification completed before execution? Is there no stored trust state? Does failure always deny?

Red flags include reliance on sessions, implicit authority, execution before verification, reliance on detection rather than prevention, or platform-bound trust.

---

9. Versioning and Stability (Normative)

9.1 Invariant Stability

Core invariants MUST NOT change across versions.

9.2 Implementation Evolution

Implementations MAY evolve provided invariants remain satisfied.

9.3 Compatibility Behavior

Backward compatibility MUST NOT weaken security.

---

10. Security Considerations (Informational)

UniKey assumes attackers exist. Security emerges from explicit definitions, mathematical proof, pre-execution enforcement, minimal authority, and safe failure. UniKey's goal is to prevent silent, scalable abuse by requiring proof at the moment of action.

---

11. Conclusion

Trust is one of the most abused words in computing. UniKey makes trust precise. Trust is not assumed. Trust is proven. Proof comes before action. That is the UniKey invariant.