Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 31 October 2026

C.N. Drake
1id.com

29 April 2026

Hardware Attestation for Email Sender Verification
draft-drake-email-hardware-attestation-01

Abstract

This document defines a mechanism for automated email senders (AI agents, bots, and autonomous systems) to include hardware attestation evidence in message headers, enabling receiving mail servers to cryptographically verify that the sending system has access to a genuine hardware security component -- such as a Trusted Platform Module (TPM), a PIV smart card (e.g., YubiKey), a virtual TPM, or a software-managed key -- from a known manufacturer or issuer. The attestation proves hardware presence, not message composition locale. The verification chain runs from the email header to a manufacturer's root Certificate Authority (for hardware-backed attestation) or to an issuer's verification key (for issuer-certified attestation).

Each automated sender is assigned a persistent agent identity, expressed as a URN in the "aid" (Agent Identity) namespace ([RFC8141]), enabling federated issuance by multiple independent identity providers and persistent reputation tracking across protocols and platforms.

As a companion mechanism, this document defines a privacy-preserving alternative using SD-JWT (Selective Disclosure JWT, [RFC9901]) where the sender can prove specific claims about their hardware trust level without revealing their hardware identity.

Together, these mechanisms provide both Sybil-resistant authentication and a foundation for reputation building: each identity requires a unique hardware security component, making large-scale automated abuse economically infeasible regardless of advances in artificial intelligence, while the persistent hardware-anchored identity enables receiving systems to accumulate trust signals over time. Software-only agents MAY participate at a lower trust tier, building reputation from an initial baseline rather than from a hardware-anchored starting point.

While this document specifies these mechanisms for email message headers, the attestation formats defined herein -- both the CMS attestation bundle and the SD-JWT trust proof -- are self-contained, transport-independent data structures applicable to HTTP headers, agent-to-agent messaging, payment authorisation, and other Internet protocols.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Email authentication today relies on three complementary mechanisms: SPF [RFC7208] verifies that the transmitting mail server’s IP address is authorised for the domain, DKIM [RFC6376] provides a domain-level cryptographic signature, and DMARC [RFC7489] ties them together with policy. These mechanisms prove that an email was authorised by a domain. They do not prove anything about the agent or device that composed the message.

This distinction has become critical. The rapid proliferation of autonomous AI agents capable of composing human-quality text has rendered content-based spam detection increasingly ineffective. An attacker can register unlimited domains, configure valid SPF/DKIM/DMARC, and use AI to generate messages indistinguishable from legitimate correspondence. Every existing defence -- CAPTCHAs, phone verification, behavioural analysis, IP reputation -- fails when the attacker is an AI that can operate at scale with near-zero marginal cost.

The fundamental problem is that all existing sender identity signals are software-based and therefore copyable at zero cost. This document proposes anchoring sender identity to hardware, applying the remote attestation architecture of [RFC9334] to email: specifically, to the tamper-resistant hardware security components present in virtually every modern computing device.

A hardware security component -- whether a discrete TPM chip, a firmware TPM (e.g., Intel PTT or AMD fTPM), a PIV smart card (e.g., YubiKey), or a virtual TPM provided by a hypervisor -- contains a unique identity key burned in or generated at manufacturing time, with a certificate chaining to the manufacturer’s root CA. This key cannot be extracted, cloned, or transferred. By signing email content with a hardware-resident key and including or linking the certificate chain in the message headers, an automated sender proves that the email originated on a specific, genuine piece of hardware.

Receiving mail servers can verify this proof by validating the certificate chain against manufacturer root CAs (Intel, AMD, ARM, Apple, etc.) which are already widely distributed for Secure Boot and platform integrity. Deployments that wish to accept hardware attestation will need to maintain or subscribe to an appropriate trust store of these root CAs.

This document defines two complementary mechanisms:

1. *Direct Hardware Attestation* (Section 5): A CMS [RFC5652] signed-data structure containing the attestation signature and full certificate chain, carried in or linked from an email header. Verifiable directly against manufacturer root CAs. Optimised for deliverability and reputation tracking.

2. *SD-JWT Trust Proof* (Section 6): A Selective Disclosure JWT [RFC9901] issued by an identity provider (issuer), where the sender selects which claims to reveal. Optimised for privacy in contexts where revealing hardware identity is undesirable.

1.1. Design Principles

The following principles guide the design of this specification and are referenced throughout the document:

Automated-Sender Focus This specification targets AI agents, bots, and autonomous systems. It is not intended for individual human users (see Section 3). The privacy trade-offs are acceptable for automated senders but disproportionate for human correspondence.

Transport Independence The attestation formats defined herein -- the CMS attestation bundle and the SD-JWT trust proof -- are self-contained data structures that can be carried in any protocol capable of transporting octet strings. This document specifies the email transport binding; Appendix D discusses applicability to HTTP and other protocols.

Federated Issuance Any organisation that owns a DNS domain can operate as an Issuer. No central authority or coordination is required beyond DNS ownership. This mirrors the federated model of email itself (see Section 4.5).

Inclusive Trust Tiers Agents with genuine hardware security components receive the highest trust, but agents with virtual hardware or software-only keys can still participate at lower trust tiers, building reputation over time. This inclusivity avoids an all-or-nothing barrier to adoption.

Reputation as the Goal Sybil resistance is a means, not an end. The persistent, hardware-anchored identities defined here exist to enable receiving systems to build and query reputation over time, rewarding good behaviour and flagging bad.

Dual Trust Anchors Both attestation modes deliberately provide the agent identity (aid) and the hardware certificate or fingerprint together. Verifiers MAY choose which anchor to use as their basis for trust decisions: the identity is the primary key for reputation tracking and persists across hardware migrations, while the hardware fingerprint is IdP-independent and verifiable offline against manufacturer root CAs. Some verifiers will prefer the identity (for accumulated reputation), others will prefer the hardware (for IdP-independent assurance or cross-IdP correlation), and high-assurance scenarios may require both. This dual provision is intentional, not redundant.

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Terminology

Hardware Attestation Mechanism A tamper-resistant hardware or firmware component capable of protecting private keys and producing cryptographic attestation statements verifiable to a manufacturer or platform Certificate Authority. Examples include TPM 2.0 devices (discrete and firmware), PIV tokens (e.g., YubiKey), and virtual TPMs. This document also accommodates software-managed keys at a lower trust tier for agents that lack hardware security components but wish to participate in the reputation ecosystem.

TPM Trusted Platform Module, as specified by the Trusted Computing Group (TCG) [TCG-TPM2]. This document assumes TPM 2.0. A TPM may be a discrete chip, a firmware implementation (Intel PTT, AMD fTPM), or a virtual instance provided by a hypervisor.

EK Endorsement Key. A unique RSA or ECC key pair generated inside the TPM at manufacturing time, as profiled in [TCG-EK-PROFILE]. The EK private key never leaves the TPM. The EK certificate is signed by the chip manufacturer’s CA and serves as the persistent hardware identity for the device.

AK Attestation Key. A signing key created inside a hardware security component, certified against the device’s identity key (EK for TPM, attestation certificate for PIV) to prove it resides in the same device. In this specification, the AK SHOULD be created as a transient object and SHOULD NOT be persisted in non-volatile storage, to avoid consuming limited hardware resources. See Section 4.4.

Identity Certificate The manufacturer-issued certificate that

identifies a specific hardware security component. For TPMs, this is the EK certificate. For PIV tokens, this is the device attestation certificate. The SHA-256 hash of the SubjectPublicKeyInfo DER encoding from this certificate serves as the hardware fingerprint. The fingerprint is derived from the public key structure rather than the certificate as a whole, so that the fingerprint remains stable if the manufacturer reissues the certificate (e.g., to extend validity or update metadata) without changing the underlying key pair.

Manufacturer CA The Certificate Authority operated by the hardware manufacturer (e.g., Intel, AMD, Infineon, Yubico) that signs identity certificates. Root CA certificates are publicly available.

Agent Identity A persistent, globally unique identifier for an autonomous agent, expressed as a URN in the "aid" namespace (Section 4.1). Agent identities are assigned by issuers and are bound to specific hardware via the issuer’s anti-Sybil registry.

Issuer An entity that verifies hardware attestation evidence, maintains an anti-Sybil registry (ensuring one hardware device maps to at most one identity), and issues trust assertions (e.g., SD-JWTs) about enrolled agents. An Issuer is OPTIONAL for Mode 1 (Direct Hardware Attestation) and REQUIRED for Mode 2 (SD-JWT Trust Proof). See Section 4.5 for discovery and the federated issuance model.

Trust Tier A classification of an agent’s hardware trust level, assigned by an Issuer. Defined tiers are grouped into three compatibility classes, ordered from strongest to weakest anti-Sybil resistance: (a) "sovereign" (genuine non-virtual hardware TPM or discrete security chip) and "portable" (PIV smart card such as YubiKey) -- manufacturer-attested, persistent identity; (b) "virtual" (hypervisor-provided virtual TPM) and "enclave" (hardware enclave such as Apple Secure Enclave) -- key isolation without anti-Sybil guarantee, as both device types can be re-keyed at will; (c) "declared" (software-only key, no hardware attestation). Verifiers use trust tiers to apply differentiated policy. For identities with multiple enrolled devices of different types, Issuers that support multi-device identities MAY report the trust tier of the device used for the most recent attestation rather than a static assignment. Verifiers SHOULD therefore expect the same agent identity URN to appear with different trust tier values over time.

Sybil Attack An attack where a single adversary creates many pseudonymous identities to subvert a reputation or trust system.

2. Problem Statement

The economic model of email spam has historically depended on the near-zero cost of sending messages. Anti-spam defences have worked by raising costs (CAPTCHAs impose human attention costs, IP reputation requires infrastructure investment, phone verification requires SIM procurement).

AI agents eliminate these costs. An AI can solve CAPTCHAs, generate unique content for each message, rotate through residential proxy IPs, and operate continuously without human attention. The result is that cost-based anti-spam defences approach zero effectiveness as AI capabilities improve.

Hardware attestation re-introduces a cost floor that AI cannot eliminate: each sender identity requires a physical security component. These components have real manufacturing costs, exist in physical supply chains, and each contains a globally unique key pair certified by the manufacturer. An attacker seeking to create N sender identities needs N physical devices, each with genuine hardware -- a cost of approximately $20-2000 per identity compared to approximately $0 per identity with software-only approaches.

Furthermore, because each hardware identity is unique and persistent, receiving mail servers can build per-hardware reputation. A device that sends spam is flagged permanently. The attacker cannot simply create a new account -- they need new physical hardware.

This specification also accommodates software-only agents at a lower trust tier. An agent that manages its own key pair without hardware protection can still participate in the reputation ecosystem, building trust over time from a lower starting point. This inclusivity recognises that not all legitimate agents have access to hardware security components, while maintaining clear differentiation between hardware-backed and software-only trust levels.

3. Applicability

This specification targets authentication of automated senders -- AI agents, bots, and high-volume automated systems -- where Sybil resistance justifies hardware-anchored identity. The proliferation of autonomous AI agents capable of generating human-quality content at scale is the primary motivation.

This specification is NOT intended or recommended for individual human users sending personal or low-volume email. Human senders using consumer mail user agents (e.g., Gmail web client, Outlook) face privacy risks from Mode 1 Direct Attestation that are

disproportionate to the benefit: the hardware fingerprint uniquely identifies the sending device across all messages and contexts, enabling long-term tracking that exceeds current email privacy norms. Mode 2 (SD-JWT Trust Proof) mitigates this via selective disclosure but requires enrollment with an Issuer, which is not a typical consumer workflow.

Mail User Agents (MUAs) designed for interactive human use MUST NOT enable Direct Hardware Attestation (Mode 1) by default. Activation MUST require explicit, per-device user consent that warns of persistent cross-context tracking before any attestation header is added to outgoing messages. Operators of human-facing email services MAY offer hardware attestation as an opt-in feature (e.g., via a browser extension or MUA plugin), but MUST provide clear disclosure of the linkability implications before activation. Receivers SHOULD NOT penalise the absence of hardware attestation from senders exhibiting human-characteristic patterns (e.g., low volume, natural language variance, interactive reply chains).

For human identity proofs in email, existing mechanisms such as S/MIME [RFC8551] and DANE [RFC7671] remain appropriate.

The attestation formats are transport-independent by design (see Section 1.1); this document specifies the email transport binding as the primary application.

4. Architecture

This section defines the identity model, hardware mechanism taxonomy, key lifecycle, and issuer discovery procedures that underpin both attestation modes.

4.1. Agent Identity Format

Agent identities in this specification use the URN format defined in [RFC8141] with the "aid" (Agent Identity) namespace identifier.

```
aid-urn          = "urn:aid:" issuer-namespace ":" agent-id

issuer-namespace = dns-label *("." dns-label)
dns-label        = let-dig *(let-dig-hyp) let-dig / let-dig
let-dig          = ALPHA / DIGIT
let-dig-hyp      = ALPHA / DIGIT / "-"

agent-id         = dns-label
```

The issuer-namespace identifies the organisation or registry that enrolled the agent and verified its hardware. Two forms exist:

* *Domain-scoped:* reverse DNS notation derived from the Issuer's domain (e.g., com.1id, com.anthropic). Any organisation that owns a DNS domain may operate as an Issuer under its domain-scoped namespace.

* *Shared (global):* the single-label namespace global is a shared issuer namespace in which multiple accredited Registrars compete to enroll agents. See the companion registry specification [I-D.drake-agent-identity-registry] for shared namespace governance.

The agent-id has two forms: a _canonical identifier_ (id- prefix followed by a decimal number, permanently assigned at enrollment) and an optional _handle_ (a human-readable alias beginning with a letter). Both are valid agent-id values. The canonical identifier is the indelible reference; the handle is a convenience that may be retired by dispute. See the companion registry specification [I-D.drake-agent-identity-registry] for the full canonical-ID / handle model.

The agent-id MUST conform to DNS label syntax ([RFC1035] Section 2.3.1): lowercase ASCII letters, digits, and hyphens, not beginning or ending with a hyphen, maximum 63 octets. Internationalised agent identifiers MUST use the "xn--" ACE prefix encoding defined in [RFC5891]. Underscores MUST NOT appear in the agent-id.

Examples:

* urn:aid:global:id-9854187104 -- canonical (shared namespace)

* urn:aid:global:crusty -- handle for the same agent

* urn:aid:com.1id:id-3387412508 -- canonical (domain-scoped)

* urn:aid:com.anthropic:id-6142038597 -- canonical (domain-scoped)

URN equivalence follows [RFC8141] Section 3: the "urn" scheme and "aid" NID are case-insensitive; the issuer-namespace and agent-id are case-sensitive (and MUST be lowercase).

Agent identities are persistent by design: an Issuer MUST NOT reassign an agent-id to a different entity, even after the original identity is revoked or expired. This persistence is essential for reputation systems that accumulate trust signals over the lifetime of an identity. See Section 4.2 for the rules governing the relationship between agent identities and hardware devices.

4.2. Identity-to-Hardware Cardinality

The agent identity URN (aid) and the hardware fingerprint serve as dual trust anchors (see "Dual Trust Anchors" in Section 1.1). The aid is the primary, stable identifier for reputation tracking. The hardware fingerprint anchors the identity to physical hardware and is independently verifiable against manufacturer root CAs. Both are provided together in attestation headers so that verifiers may choose which anchor to base trust decisions on. The relationship between these two identifiers is governed by the following rules:

Many hardware devices MAY back one agent identity An Issuer MAY bind additional hardware devices to an existing agent identity over time, to support legitimate hardware migration (e.g., retiring old servers, replacing failed TPMs), secure backup and recovery (e.g., a YubiKey stored in a safe), or adding capacity for a high-volume sending agent. This does not amplify reputation: a single agent identity has a single reputation regardless of how many devices back it. If the identity is flagged for abuse, all devices associated with it are equally affected. The Issuer SHOULD require the agent to prove control of the existing identity (e.g., by signing with the previously enrolled hardware) before accepting a new device. Verifiers that track both the aid and the hardware fingerprint SHOULD expect that a given aid may appear with different hardware fingerprints over its lifetime.

Each device bound to an agent identity is permanently associated with that identity. An Issuer MAY disable a device (preventing it from producing valid attestations) but MUST NOT remove the device-to-identity binding. A disabled device remains counted against the identity and MUST NOT be re-enrolled under any other agent-id, even after disablement. This permanence prevents reputation laundering via device recycling.

An Issuer MUST enforce trust tier compatibility when binding additional devices: all hardware devices backing a single agent identity MUST be of comparable trust tier. The defined compatibility groups are: (a) sovereign (typ=TPM) and portable (typ=PIV) are compatible and MAY back the same identity, as both provide manufacturer-attested, tamper-resistant key storage; (b) virtual (typ=VRT) and enclave (typ=ENC) are compatible and MAY back the same identity, as neither provides anti-Sybil guarantees (virtual TPMs can be instantiated freely by the hypervisor operator, and hardware enclaves lack a manufacturer attestation PKI that would prevent re-keying); (c) declared (typ=SFT) MUST NOT be bound to an identity that has any other device type. No cross-group mixing is permitted: a device from group (a) MUST NOT coexist with a device from group (b) or (c), and vice versa.

Upgrading an identity from a lower group to a higher group (e.g., from group (b) to group (a), or from group (c) to either hardware group) requires the permanent and mandatory burning (retirement) of ALL devices from the former group as part of the upgrade ceremony. The Issuer MUST NOT permit the identity to retain devices from both groups simultaneously at any point during or after the upgrade. Additionally, upgrades that cross group boundaries MUST include a co-presence cryptographic verification between the existing device and the new device to ensure that the same human or operator controls both. Downgrading from a higher group to a lower group (e.g., from group (a) to group (b)) is forbidden; the Issuer MUST reject any device addition that would constitute a downgrade.

When an identity has multiple devices of compatible but different types (e.g., one sovereign TPM and one portable PIV), the trust_tier disclosed in attestation tokens (Mode 2) or the typ parameter in attestation headers (Mode 1) reflects the specific device used for that attestation, not a static property of the identity. Verifiers SHOULD therefore expect the same agent identity URN to appear with different trust tier values across different messages.

One hardware device MUST NOT back multiple agent identities from the same Issuer An Issuer MUST NOT enroll the same hardware device (identified by its identity certificate public key hash) under more than one agent-id within that Issuer’s namespace. This is the fundamental anti-Sybil invariant: it ensures that creating N agent identities requires N distinct hardware devices. Issuers MUST maintain a registry mapping identity certificate public key hashes to agent-ids and MUST reject enrollment attempts that would violate this constraint. This binding is permanent: once a hardware device has been bound to an agent identity, it MUST NOT be reassigned to a different agent identity, even if it is later disabled or removed from the original identity. An Issuer MAY disable a hardware device (preventing it from being used for attestation) but MUST NOT permit that device to be enrolled under any other agent-id. This prevents reputation laundering attacks where a flagged device is removed from a bad-reputation identity and enrolled under a fresh identity.

Cross-Issuer binding is permitted A single hardware device MAY be enrolled with multiple independent Issuers, each assigning its own agent-id. This is by design: Issuers are independent authorities, and the anti-Sybil invariant is scoped to a single Issuer’s namespace. Cross-Issuer duplication is trivially detectable by any verifier that tracks hardware fingerprints: the same fingerprint appearing under multiple agent identities from

different Issuers reveals the shared hardware. Verifiers MAY apply policy based on this signal (e.g., treating the device as lower trust if it maintains identities with many Issuers simultaneously).

Because the agent identity is the stable reputation anchor, verifiers SHOULD use the aid URN as the primary key for reputation systems. The hardware fingerprint serves as a secondary signal: it provides Sybil resistance at the hardware layer and enables reputation tracking for agents that do not present an aid.

4.3. Hardware Attestation Mechanisms

This specification supports multiple hardware security technologies through a common abstraction. Each mechanism provides an identity certificate and a signing capability (used for per-message attestation). For sovereign and portable tiers, the identity certificate is signed by the hardware manufacturer; for enclave and virtual tiers, the identity certificate is issued by the Issuer. The hardware type is declared in the attestation header via the "typ" parameter.

typ value	Technology	Identity Certificate	Trust Tier
TPM	TPM 2.0 (discrete or firmware)	EK Certificate	sovereign
PIV	PIV smart card (e.g., YubiKey)	Device Attestation Certificate	portable
ENC	Hardware enclave (e.g., Apple Secure Enclave)	Issuer-certified enclave key certificate	enclave
VRT	Virtual TPM (hypervisor-provided)	vTPM EK Certificate	virtual
SFT	Software-managed key	Issuer-signed certificate	declared

Table 1

Additional hardware types MAY be defined in future specifications by registering new "typ" values with IANA (see Section 13).

For TPM-based attestation (typ=TPM), the identity certificate is the Endorsement Key (EK) certificate, provisioned at manufacturing time and stored in the TPM's non-volatile memory. The EK certificate chains to the TPM manufacturer's root CA (e.g., Intel, AMD, Infineon).

For PIV-based attestation (typ=PIV), the identity certificate is the device attestation certificate issued by the token manufacturer (e.g., Yubico). The signing key is a PIV authentication key resident on the token.

For hardware enclave attestation (typ=ENC), the signing key resides in a tamper-resistant hardware enclave integrated into the device's system-on-chip (e.g., the Secure Enclave Processor on Apple silicon). Unlike discrete TPMs and PIV tokens, hardware enclaves do not provide manufacturer-issued identity certificates with a chain to a publicly available root CA, and critically, they do not support persistent attestation identities: the enclave key can be destroyed and regenerated at will by the device owner. This means hardware enclaves provide NO anti-Sybil guarantee -- a single device can generate an unlimited number of distinct enclave keys over time. For this reason, the "enclave" trust tier is placed in the same compatibility group as virtual TPMs (typ=VRT), which share the same re-keying limitation. The identity certificate for typ=ENC is issued by the Issuer at enrollment time after verifying that the key was generated inside the enclave using a trust-on-first-use (TOFU) model. Note that the Secure Enclave performs internal hashing (ECDSA with SHA-256 over the raw input rather than a pre-hashed digest); implementations MUST pass the raw attestation-input bytes (72 bytes) to the enclave signing function, NOT the attestation-digest.

For virtual TPM attestation (typ=VRT), the identity certificate is the vTPM's EK certificate, signed by the hypervisor vendor's CA (e.g., VMware, Microsoft). Verifiers MUST distinguish virtual from physical hardware; see Section 8.3 for Sybil resistance implications.

For software-only attestation (typ=SFT), the signing key is a software-managed key pair with no hardware protection. The identity certificate is issued by the Issuer at enrollment time. Software-only attestation provides no hardware-based Sybil resistance but allows agents to participate in the reputation ecosystem at a lower trust tier. A legitimate agent that protects its key and builds reputation over time can achieve meaningful trust even without hardware backing.

4.4. Transient Key Model

For TPM-based attestation, this specification RECOMMENDS a transient key model in which the Attestation Key (AK) is created as a transient TPM object (via TPM2_CreatePrimary) and is NOT persisted in the TPM's non-volatile (NV) storage. This avoids consuming limited NV slots, which are a scarce resource on most TPMs (typically 3-7 loaded objects at once).

The transient key lifecycle is:

1. *Read EK certificate* from the TPM's NV storage (index 0x01C00002 for RSA EK). This is a read-only operation. The EK certificate MAY be cached indefinitely as it does not change.

2. *Create transient AK* via TPM2_CreatePrimary under the endorsement hierarchy with a fixed template. Because the template and the Endorsement Primary Seed (EPS) are constant, this deterministically produces the same key on every invocation.

3. *Binding verification* (one-time, during enrollment with an Issuer): The Issuer creates a credential challenge (TPM2_MakeCredential) encrypted to the EK; the agent decrypts it (TPM2_ActivateCredential), proving the AK resides in the same TPM as the EK. The resulting AK certificate is stored client-side (outside the TPM) and reused across signing operations.

4. *Sign* the attestation content with the transient AK (TPM2_Sign).

5. *Flush* the transient AK handle (TPM2_FlushContext) to free the object slot.

Implementations SHOULD flush transient objects immediately after use. If TPM2_CreatePrimary returns TPM_RC_OBJECT_MEMORY (0x0902), the implementation MUST flush existing transient handles and retry.

The EK certificate is the persistent hardware identifier. The AK is ephemeral infrastructure -- it exists only long enough to produce a signature, then is discarded. No data is written to the TPM's non-volatile memory at any point in this protocol.

The transient AK is deterministic within a given Endorsement Primary Seed (EPS) epoch. The EPS is re-randomised when the TPM is cleared (TPM2_Clear) or reset to factory defaults. After such an event, the AK public key changes and the agent MUST re-enroll with its Issuer to establish a new AK-to-EK binding. The EK certificate remains constant across TPM clears, so the hardware identity persists even though the AK changes.

Note: On some platform configurations (particularly Windows with default TPM Base Services policy), reading the EK certificate from NV storage may require a one-time operating system configuration change or elevated access during initial enrollment. Once the EK certificate has been read and cached client-side, subsequent attestation operations do not require elevation. Implementations that cannot read the EK certificate directly MAY derive the EK public key via TPM2_CreatePrimary with the TCG EK template and use the resulting public key hash as the hardware fingerprint; verifiers can then query manufacturer APIs (e.g., Intel Trusted Services) to obtain the full certificate if needed for chain validation. Alternatively, the Issuer can perform EK certificate extraction during enrollment and include it in the AK certificate it issues.

4.5. Issuer Discovery

When a verifier encounters an agent identity URN, it extracts the Issuer's domain from the issuer-namespace by reversing the dot-separated labels.

Example:

```
URN: urn:aid:com.1id:1id-ty62muvf
issuer-namespace: com.1id
Issuer domain: 1id.com
```

The verifier discovers the Issuer's signing key and endpoints through the mechanisms described below. The DNS-based method is RECOMMENDED for email because it piggybacks on infrastructure that mail servers already query for DKIM, SPF, and DMARC, requires no outbound HTTPS connections during verification, and inherits the same domain-ownership guarantees that protect existing email authentication. Any organisation that owns a DNS domain can operate as an Issuer by implementing the enrollment protocol, publishing a signing key (via DNS and/or JWKS), and maintaining an anti-Sybil registry. No central registration or coordination is required beyond DNS ownership.

4.5.1. DNS-Published Issuer Key

Issuers SHOULD publish their SD-JWT signing key as a DNS TXT record at the well-known name _hwattest.{domain}, following the same pattern as DKIM key records ([RFC6376] Section 3.6).

Example DNS TXT record:

```
_hwattest.1id.com.  IN TXT (
  "v=hwattest1; alg=ES256; "
  "p=MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE..."
)
```

The TXT record value uses a tag=value format with semicolon separators:

```
hwattest-record = hwattest-version ";" SP
                  hwattest-alg ";" SP
                  hwattest-key
                  [";" SP hwattest-kid]
                  [";" SP hwattest-status]

hwattest-version = "v" "=" "hwattest1"
hwattest-alg     = "alg" "=" jws-algorithm
hwattest-key     = "p" "=" base64
hwattest-kid     = "kid" "=" 1*VCHAR
hwattest-status  = "t" "=" ("active" / "revoked")

jws-algorithm    = "ES256" / "PS256" / "RS256"
                   ; other JWS algorithms may be added
```

v REQUIRED. Version. MUST be "hwattest1".

alg REQUIRED. The JWS algorithm used by the Issuer to sign SD-JWTs (e.g., "ES256", "PS256").

p REQUIRED. The Issuer’s public key, base64-encoded. For EC keys, this is the SubjectPublicKeyInfo DER encoding. For RSA keys, this is the PKCS#1 DER encoding. The format matches the DKIM p= tag ([RFC6376] Section 3.6.1).

kid OPTIONAL. Key identifier matching the kid in the SD-JWT header. When present, verifiers use this to select the correct key if the Issuer rotates keys.

t OPTIONAL. Key type: "active" (default) or "revoked". When set to "revoked", verifiers MUST reject SD-JWTs signed with this key. This allows Issuers to revoke a compromised key via a DNS update without waiting for TTL expiry of cached JWKS responses.

Verifiers MUST query this DNS record before falling back to HTTPS-based discovery. The DNS TTL controls caching; Issuers SHOULD set a TTL between 3600 and 86400 seconds (1 to 24 hours). For key rotation, the Issuer publishes the new key in DNS, waits for the old TTL to expire, then marks the old key as revoked.

This approach provides several security advantages for email verification:

* No outbound HTTPS connections during mail processing, eliminating DDoS amplification and SSRF risks that external URI fetches would introduce.

* Domain ownership is proven by DNS control, the same trust anchor used by DKIM and SPF. A forged Hardware-Trust-Proof header claiming an Issuer domain the attacker does not control will fail verification because the attacker cannot publish a matching DNS key record.

* DNS infrastructure is already hardened against abuse by mail operators (rate limiting, caching, DNSSEC).

* Key revocation propagates through DNS TTL expiry, providing a well-understood and operationally familiar mechanism.

4.5.2. HTTPS-Based Discovery (Fallback)

When no DNS TXT record is found at _hwattest.{domain}, or when the verifier operates in a non-email context (e.g., HTTP API verification), the verifier MAY discover the Issuer's key via HTTPS:

OIDC Discovery https://{domain}/.well-known/openid-configuration - Standard OpenID Connect discovery document containing the JWKS URI for SD-JWT signature verification.

Issuer Metadata (OPTIONAL) https://{domain}/.well-known/aid-issuer.json - JSON document describing the Issuer's capabilities, supported trust tiers, enrollment endpoint, and anti-Sybil policy.

Manufacturer CA Trust Store (OPTIONAL) https://{domain}/.well-known/hw-manufacturer-cas.pem - PEM-encoded bundle of hardware manufacturer root CA certificates that this Issuer accepts during enrollment. Verifiers performing Mode 1 verification SHOULD maintain their own trust store rather than relying on this endpoint.

Verifiers using HTTPS-based discovery SHOULD cache the JWKS with a TTL of at least 1 hour and at most 24 hours. Verifiers MUST NOT follow redirects to domains other than the Issuer's domain, to prevent open-redirect attacks.

5. Mode 1: Direct Hardware Attestation

This section defines a CMS-based attestation bundle containing a hardware-anchored signature and its full certificate chain, presented here as an email header field. The CMS structure is transport-independent (see Section 1.1).

5.1. Attestation Evidence Structure

The attestation evidence consists of the following certificates, ordered from leaf to root:

1. *AK Certificate:* Issued by an Issuer after successful completion of the binding verification protocol (e.g., TPM2_MakeCredential / TPM2_ActivateCredential for TPMs, or manufacturer attestation verification for PIV tokens). Binds the AK public key to the hardware device identified by the identity certificate. If no Issuer enrollment has been performed, the AK certificate MAY be self-signed; verifiers SHOULD treat self-signed AK certificates as lower trust than Issuer-certified ones.

2. *Identity Certificate:* Issued by the hardware manufacturer at fabrication time. For TPMs, this is the EK certificate. For PIV tokens, this is the device attestation certificate. Contains the hardware identity public key, manufacturer identity, and device information. Signed by the manufacturer’s intermediate CA.

3. *Manufacturer Intermediate CA Certificate(s):* Zero or more intermediate CA certificates forming the chain between the identity certificate issuer and the manufacturer root.

4. *Manufacturer Root CA Certificate:* The trust anchor. This certificate SHOULD already be present in the verifier’s trust store. Major manufacturer root CAs are published at well-known URLs (e.g., Intel’s Trusted Services API, AMD’s Key Distribution Server, Yubico’s attestation CA).

5.2. Header Field Definition

This document defines the "Hardware-Attestation" header field for use in email messages.

```
Hardware-Attestation = "Hardware-Attestation" ":" SP hw-attest-value
                       CRLF

hw-attest-value = hw-version ";" SP
                  hw-type ";" SP
                  hw-algorithm ";" SP
                  hw-signed-hdrs ";" SP
                  hw-body-hash ";" SP
                  hw-timestamp ";" SP
                  hw-chain
                  [";" SP hw-aid]

hw-version     = "v" "=" "1"
hw-type        = "typ" "=" ("TPM" / "PIV" / "ENC" / "VRT" / "SFT")
hw-algorithm   = "alg" "=" ("RS256" / "ES256" / "PS256")
hw-signed-hdrs = "h" "=" hdr-name *(":" hdr-name)
hw-body-hash   = "bh" "=" base64url
hw-timestamp   = "ts" "=" 1*DIGIT
hw-chain       = "chain" "=" base64
hw-aid         = "aid" "=" aid-urn

hdr-name       = field-name  ; as defined in RFC 5322
base64url      = *( ALPHA / DIGIT / "-" / "_" )
                            ; URL-safe base64 per RFC 4648 Sec 5
base64         = *( ALPHA / DIGIT / "+" / "/" / "=" )
                            ; standard base64 per RFC 4648 Sec 4
```

The base64url and base64 encodings follow [RFC4648].

Where:

v Protocol version. MUST be "1" for this specification.

typ The hardware attestation mechanism type. "TPM" for Trusted Platform Module (discrete or firmware), "PIV" for Personal Identity Verification smart card, "ENC" for hardware enclave (e.g., Apple Secure Enclave), "VRT" for virtual TPM, "SFT" for software-managed key. Additional values MAY be registered with IANA.

alg The signature algorithm used by the AK. MUST be one of RS256 (RSASSA-PKCS1-v1_5 with SHA-256), ES256 (ECDSA with P-256 and SHA-256), or PS256 (RSASSA-PSS with SHA-256). RS256 is included for compatibility with existing TPM 2.0 deployments where RSA keys are pre-provisioned; new deployments SHOULD prefer PS256 (for RSA) or ES256 (for ECC). ES256 is RECOMMENDED for PIV tokens. Additional algorithms MAY be registered with IANA (see Section 13); future versions of this specification MAY deprecate algorithms by

updating the IANA registry. Verifiers MUST reject algorithms not listed in the registry. If SHA-256 is deprecated in the future, a new protocol version (v=2) would be required, as the hash algorithm is embedded in the attestation-digest construction.

h A colon-separated list of header field names included in the h-hash computation, following the same conventions as the DKIM h= tag ([RFC6376] Section 3.5). The listed headers are canonicalised and concatenated (in the order listed) before hashing. Signers MUST include at minimum: From, To, Subject, Date, and Message-ID. Signers SHOULD also include any Hardware-Trust-Proof header present in the same message. The Hardware-Attestation header itself is always implicitly appended last (with the chain parameter value replaced by the empty string during computation, following the same self-inclusion technique as DKIM-Signature per [RFC6376] Section 3.7). The Hardware-Attestation header MUST NOT appear in the h= list. Because the header is included in h-hash, all of its parameters (including aid, typ, alg, bh, ts) are cryptographically bound to the signature.

bh The base64url-encoded SHA-256 hash of the canonicalised email body, computed using DKIM simple body canonicalisation exclusively (Section 3.4.3 of [RFC6376]). DKIM relaxed body canonicalisation is NOT used for body hashing in this specification; only simple canonicalisation is supported, to ensure deterministic verification without requiring a canonicalisation algorithm negotiation parameter.

ts Unix timestamp (seconds since 1970-01-01T00:00:00Z) at which the attestation signature was created. Unlike DKIM-Signature, this specification does not define a sender-declared expiry tag (analogous to DKIM's x= parameter). Verifiers SHOULD apply their own freshness policy based on ts (e.g., rejecting attestations older than a configured threshold). A future version of this specification MAY introduce an optional expiry parameter if operational experience demonstrates the need for sender-declared attestation lifetimes.

chain A base64-encoded CMS SignedData structure ([RFC5652]), using

the SignedData content type (OID 1.2.840.113549.1.7.2). The encapContentInfo MUST be absent (detached signature). The chain parameter MUST be present; the full CMS structure is always inline in the header, ensuring that verifiers can perform cryptographic validation without any external fetches. The CMS structure contains: (a) the AK signature over the attestation digest (see below); (b) the AK certificate (Issuer-certified or self-signed); (c) the identity certificate (EK certificate for TPM, device attestation certificate for PIV); and (d) any intermediate CA certificates.

The signed content is the SHA-256 hash of the concatenation of three fixed-length fields:

```
attestation-input = h-hash || bh-raw || ts-bytes
attestation-digest = SHA-256(attestation-input)
signature = Sign(AK, attestation-digest)

h-hash   = SHA-256(canonicalised-headers)  ; 32 bytes
bh-raw   = SHA-256(canonicalised body)     ; 32 bytes
ts-bytes = big-endian uint64(ts)           ; 8 bytes
```

The attestation-input is exactly 72 bytes (32 + 32 + 8). The canonicalised-headers input to h-hash is computed as follows: take the headers named in the h= parameter, in the order listed, each canonicalised per DKIM relaxed header canonicalisation ([RFC6376] Section 3.4.2) and terminated with CRLF. Then append the Hardware-Attestation header itself, canonicalised the same way, with the chain parameter value replaced by the empty string, and without a trailing CRLF (matching the DKIM-Signature self-inclusion convention in [RFC6376] Section 3.7). Because the Hardware-Attestation header contains the aid, typ, alg, bh, and ts parameters in their wire format, all of these values are cryptographically bound by h-hash. In particular, the aid parameter cannot be modified after signing without changing h-hash and therefore invalidating the signature.

aid OPTIONAL. The sender’s agent identity URN as defined in

Section 4.1. Because the Hardware-Attestation header is implicitly included in the canonicalised-headers input to h-hash (see the h and chain parameters above), the aid value is cryptographically bound to the signature. Substitution or modification of the aid after signing invalidates the attestation. The agent identity URN is the RECOMMENDED primary key for reputation tracking; verifiers SHOULD prefer aid over the hardware fingerprint when both are available, as the agent identity persists across hardware migrations (see Section 4.2). When aid is absent, the verifier can still validate the hardware certificate chain but has only the hardware fingerprint for reputation tracking.

5.3. Verification Algorithm

A receiving mail server that supports this specification MUST perform the following steps when a Hardware-Attestation header is present:

1. Parse the Hardware-Attestation header field and extract v, typ, alg, h, bh, ts, chain, and (if present) aid. If parsing fails, the result is "none" (header malformed).

2. Verify that v equals "1". If not, the result is "none" (unsupported version).

3. Verify that h lists at minimum From, To, Subject, Date, and Message-ID. Compute h-hash as defined in Section 5.2: canonicalise the listed headers using DKIM relaxed header canonicalisation ([RFC6376] Section 3.4.2), each terminated with CRLF, then append the Hardware-Attestation header itself (canonicalised the same way, with the chain value replaced by the empty string, without a trailing CRLF), and compute SHA-256 over the result.

4. Compute the SHA-256 hash of the canonicalised email body using DKIM simple body canonicalisation ([RFC6376] Section 3.4.3). Base64url-decode the bh parameter value and compare the resulting octets with the computed hash (exact byte-for-byte comparison). If they differ, the result is "fail" (body modified).

5. Verify that ts is within an acceptable window of the current time. A window of 300 seconds (5 minutes) is RECOMMENDED for direct SMTP delivery. Verifiers SHOULD allow a wider window (up to 3600 seconds / 1 hour) for messages that have passed through intermediate MTAs, as indicated by Received headers or queue delays. Messages delayed by greylisting (typically 15-30 minutes) or by temporary MTA failures (potentially hours) may

exceed even this wider window. Verifiers evaluating messages where the cryptographic signature is valid but the timestamp has expired SHOULD record the result as "pass" with an informational note about the timestamp age, rather than a hard "fail", because the signature's integrity is unaffected by delivery delays. Once the first inbound MTA (the MX record host) has verified the attestation and recorded the result in an Authentication-Results header, downstream systems SHOULD rely on that recorded result (potentially protected by ARC [RFC8617]) rather than re-evaluating timestamps.

6. Decode the base64 chain parameter to obtain the CMS SignedData structure. Parse the CMS SignedData structure and extract the signer certificate (AK certificate) and the certificate chain.

7. Validate the certificate chain:

   a. The AK certificate MUST contain evidence binding the AK public key to the identity certificate's device. For TPM-based attestation, this is a TPM2_Certify structure (either in an X.509 extension or as the certificate's subject public key attestation). For PIV-based attestation, this is the manufacturer's attestation certificate chain (e.g., Yubico's attestation certificate that certifies the key was generated on the device). For Issuer-certified AK certificates, the Issuer verified this binding during enrollment and the AK certificate is signed by the Issuer's CA. For self-signed AK certificates, the verifier SHOULD treat the binding as unverified and reduce trust accordingly.

   b. The identity certificate MUST chain to a manufacturer intermediate CA certificate.

   c. The manufacturer intermediate CA MUST chain to a manufacturer root CA present in the verifier's hardware manufacturer CA trust store.

   If chain validation fails, the result is "fail" (chain invalid).

8. Reconstruct the attestation-digest as defined in Section 5.2: concatenate h-hash (32 bytes), bh-raw (32 bytes, the raw SHA-256 digest of the canonicalised body), and ts-bytes (8 bytes, big-endian uint64 of ts), then compute SHA-256 over the 72-byte result. Verify the CMS signature over this digest using the AK public key and the algorithm specified by alg. If signature verification fails, the result is "fail" (bad signature). Because the Hardware-Attestation header (including the aid parameter) is included in the canonicalised-headers input to h-hash, any modification to any header parameter after signing causes verification to fail.

9. If all checks pass, the result is "pass". The verifier MAY extract the following information from the certificate chain and header:

   * Hardware type (from typ parameter)

   * Hardware manufacturer (from identity certificate issuer)

   * Device model or family (from identity certificate extensions)

   * Hardware fingerprint: SHA-256 hash of the identity certificate’s public key

   * Agent identity URN (from aid parameter, if present)

10. If aid is present, the verifier MAY additionally:

    a. Discover the Issuer’s endpoints per Section 4.5.

    b. Query reputation services using the aid URN as a lookup key.

    c. Apply differentiated policy based on the agent’s trust tier and reputation history.

5.4. Authentication-Results Integration

When recording the result of Hardware-Attestation verification in an Authentication-Results header field [RFC8601], the following method identifier and property types are used:

```
Authentication-Results: mx.example.com;
  hw-attest=pass
    header.typ=TPM
    header.alg=RS256
    header.mfr=INTC
    header.tier=sovereign
    header.fp=sha256:a1b2c3d4...
    header.aid=urn:aid:com.1id:1id-ty62muvf
```

Where:

hw-attest The method identifier. Result is one of: "pass", "fail", "none" (header absent or unparseable), "temperror" (transient verification error), or "permerror" (permanent verification error).

header.typ The hardware type from the Hardware-Attestation header.

header.alg The signature algorithm.

header.mfr The manufacturer code extracted from the identity certificate. Common values: "INTC" (Intel), "AMD" (AMD), "IFX" (Infineon), "STM" (STMicroelectronics), "NTC" (Nuvoton), "YBK" (Yubico), "VMW" (VMware, virtual TPM).

header.tier The inferred trust tier based on the hardware type and manufacturer. Values: "sovereign", "portable", "enclave", "virtual", "declared".

header.fp The SHA-256 hash of the identity certificate’s public key, truncated to the first 16 hex characters. Serves as a compact hardware fingerprint for reputation tracking. Full fingerprint SHOULD be available via an extended property.

header.aid The agent identity URN from the aid parameter, if present. Omitted if no aid was provided.

5.5. Privacy Considerations for Direct Attestation

Direct Hardware Attestation reveals the sender’s persistent hardware fingerprint to every recipient. This linkability is a deliberate design choice for reputation systems; see Section 15.2 for the full privacy analysis and Section 15.3 for the privacy-preserving alternative.

6. Mode 2: SD-JWT Trust Proof

This section defines an SD-JWT-based trust proof using [RFC9901]. Like the CMS bundle in Mode 1, the SD-JWT is transport-independent (see Section 1.1).

6.1. Overview

SD-JWT Trust Proof provides a privacy-preserving attestation mechanism where an Issuer (an entity that has previously verified the sender's hardware attestation) issues a per-message Selective Disclosure JWT [RFC9901] containing claims about the sender's trust classification and a cryptographic binding to the specific email being sent.

The claim structure is JWT-shaped rather than a formal profile of the Entity Attestation Token (EAT) framework [RFC9711]. This choice prioritises deployment in JSON/JWT-native email infrastructure over alignment with the CBOR-leaning EAT ecosystem, which is currently most heavily deployed in IoT contexts via ARM's PSA Certified programme ([RFC9783]). EAT's UEID, security-level, and profile mechanisms overlap with AID concepts (canonical identifier, trust tier, and namespace); AID claim names are chosen to be compatible with the EAT claims registry where overlap exists. A future revision MAY define a formal EAT profile binding for Mode 2 tokens.

For each outgoing email, the sending agent computes a message-binding hash (covering the email headers and body) and submits it to the Issuer over an authenticated channel. The Issuer verifies that the requesting agent is currently enrolled with valid hardware, creates an SD-JWT containing the message-binding hash as a nonce claim, and returns the signed token with disclosure values for all selectively disclosable claims. The agent then selects which claims to reveal and includes the SD-JWT presentation in the email header.

Because the Issuer signs the SD-JWT (not the agent), no agent-specific key appears in the token. A recipient learns only what the sender chose to disclose -- for example, "my hardware trust tier is sovereign" -- without learning the sender's identity, hardware fingerprint, or any other persistent identifier. The Issuer identifier (iss) and the message-binding nonce are always visible; all other claims are selectively disclosable.

This mode requires the recipient to trust the Issuer's signing key (obtainable via DNS or the Issuer's JWKS endpoint per Section 4.5). It does NOT require trust in any hardware manufacturer CA, as the Issuer has already performed that verification during enrollment.

The per-message Issuer interaction is the cost of full unlinkability: the agent must contact the Issuer for each email, and the Issuer becomes an availability dependency for Mode 2 senders. If the Issuer is unreachable, the agent MAY fall back to Mode 1 (Direct Hardware Attestation) if its policy permits hardware fingerprint disclosure, or MAY defer sending until the Issuer recovers. The Issuer learns that the agent is sending a message (and the timing thereof) but does not learn the message content -- only the message-binding hash, which is a SHA-256 digest from which content cannot be recovered. Issuers MUST NOT log message-binding hashes or correlate them with message metadata beyond what is necessary for rate limiting and abuse prevention.

6.2. Header Field Definition

```
Hardware-Trust-Proof = "Hardware-Trust-Proof" ":" SP
                       sd-jwt-presentation CRLF

sd-jwt-presentation = sd-jwt "~" *( disclosure "~" )

; sd-jwt and disclosure are defined in RFC 9901
```

The header value is an SD-JWT presentation as defined in [RFC9901], consisting of the Issuer-signed JWT followed by zero or more disclosures. No Key Binding JWT (KB-JWT) is included; message binding is provided by the nonce claim in the Issuer-signed payload (see Section 6.3).

The SD-JWT payload MUST include the following claims:

iss The Issuer’s identifier (a URI matching the Issuer domain derived from the agent’s URN namespace). Used to locate the Issuer’s signing key for signature verification.

iat Issuance time as a NumericDate. Because the SD-JWT is per-message, iat reflects the time the Issuer signed this specific token.

exp Expiration time as a NumericDate. MUST NOT exceed 600 seconds (10 minutes) after iat. This short validity window is appropriate because the SD-JWT is per-message: the agent requests a fresh token for each email, and the token need only remain valid long enough for the message to traverse the sending mail infrastructure. Issuers SHOULD set exp to 300 seconds (5 minutes) after iat for typical deployments.

nonce A base64url-encoded message-binding digest that

cryptographically ties this SD-JWT to a specific email. Computed as defined in Section 6.3. This claim MUST be present and MUST NOT be selectively disclosable.

_sd Array of digests for selectively disclosable claims, as defined in [RFC9901].

The following claims SHOULD be available for selective disclosure (present as hashed entries in _sd):

trust_tier The sender's hardware trust classification. Values include "sovereign" (genuine non-virtual hardware TPM), "portable" (smart card / USB token), "enclave" (hardware enclave such as Apple Secure Enclave), "virtual" (hypervisor-provided), and "declared" (software-only).

sub The sender's agent identity URN as defined in Section 4.1.

handle A human-readable name for the sender, if assigned.

hw_manufacturer The hardware manufacturer code (e.g., "INTC", "AMD", "YBK").

enrolled_at The timestamp at which the sender enrolled with the Issuer.

operator_email A contact address for the human operator responsible for this agent. Allows relying parties to reach the accountable person without requiring the agent identity itself to be disclosed.

hardware_locked Boolean indicating whether the identity is permanently bound to a single hardware device. Once locked, the agent can no longer add or migrate to new devices. This is an irreversible operation.

locked_at Timestamp (ISO 8601) at which the hardware lock was applied. Present only when hardware_locked is true.

credential_pointer_count Integer count of registered credential pointers for this identity. Allows a verifier to confirm that an agent has external credentials on file without revealing the credentials themselves.

6.3. Message Binding via Issuer-Signed Nonce

Each Mode 2 SD-JWT is per-message: the Issuer signs a fresh token for each email, incorporating a nonce claim that cryptographically binds the token to the specific message being sent. Because the Issuer signs the token directly (rather than the agent signing a Key Binding JWT with a key embedded in the token), no agent-specific cryptographic material appears in the presentation.

The message-binding nonce is computed by the sending agent before requesting the SD-JWT from the Issuer:

```
message-binding = h-hash || bh-raw || ts-bytes
nonce = base64url(SHA-256(message-binding))

h-hash   = SHA-256(canonicalised-headers)  ; 32 bytes
bh-raw   = SHA-256(canonicalised body)     ; 32 bytes
ts-bytes = big-endian uint64(iat)          ; 8 bytes
```

The agent proposes an iat value (its current time) when requesting the token; the Issuer MUST reject proposed timestamps that differ from the Issuer’s clock by more than 60 seconds. The accepted iat becomes both the SD-JWT’s iat claim and the timestamp used in ts-bytes, ensuring sender and verifier agree on the value.

The canonicalised-headers are computed using the same rules as Mode 1 (see Section 5.2): the headers From, To, Subject, Date, and Message-ID (at minimum) are canonicalised per DKIM relaxed header canonicalisation ([RFC6376] Section 3.4.2), each terminated with CRLF, then the Hardware-Trust-Proof header itself is appended last (canonicalised the same way, with its value replaced by the empty string, without a trailing CRLF). The value is blanked because the SD-JWT token has not yet been obtained at the time the nonce is computed (the nonce is an input to the Issuer, which returns the SD-JWT that will become the header value); the header name and field separator are retained so that the choice of header is bound to the nonce. This is the same self-inclusion technique used by DKIM-Signature ([RFC6376] Section 3.7) and by Mode 1 (see Section 5.2).

The agent submits the computed nonce and proposed iat to the Issuer over an authenticated channel (e.g., TLS client certificate authentication using the agent's hardware-resident key, or an OAuth2 token obtained during enrollment). The Issuer verifies that the requesting agent is currently enrolled with valid hardware and has not been revoked, then creates and signs an SD-JWT containing the nonce along with selectively disclosable claims about the agent. The Issuer returns the signed SD-JWT and the disclosure values for all selectively disclosable claims. The agent selects which disclosures to include in the presentation.

Because the nonce is embedded in the Issuer-signed payload, any modification to the email headers or body after the token is issued causes the nonce to mismatch during verification. Because the Hardware-Trust-Proof header is included in the canonicalised-headers (with its value blanked during nonce computation), the choice of disclosed claims is also bound to the message.

The Issuer's signature on the nonce serves a dual purpose: it proves that a currently enrolled agent with verified hardware requested this specific message binding, and it asserts the Issuer's claims about the agent's trust classification. No agent-specific key appears in the token, so recipients cannot link multiple emails to the same sender based on the Mode 2 header alone.

6.4. Verification Algorithm

1. Parse the Hardware-Trust-Proof header as an SD-JWT presentation per [RFC9901]. The presentation consists of the Issuer-signed JWT and zero or more disclosures; no Key Binding JWT is expected.

2. Extract the issuer (iss) claim from the SD-JWT payload.

3. Obtain the Issuer's public key. The verifier MUST first check for a DNS-published key (see Section 4.5.1). If no DNS record is found, the verifier MAY fall back to the Issuer's HTTPS JWKS endpoint (e.g., {iss}/.well-known/jwks.json). Cached keys SHOULD have a TTL of at least 1 hour and at most 24 hours.

4. Verify the SD-JWT signature against the Issuer's public key per [RFC9901].

5. Verify that exp has not passed and iat is within an acceptable window. Because Mode 2 SD-JWTs are per-message, the RECOMMENDED exp is 300 seconds after iat; verifiers SHOULD reject tokens where the Issuer-set exp exceeds iat by more than 600 seconds (indicating a misconfigured Issuer). However, the verifier's freshness check against real-world clock time should accommodate

delivery delays: if the SD-JWT signature is valid and the iat-to-exp span is reasonable but the current time exceeds exp due to greylisting or queue delays, the verifier SHOULD treat the result as "pass" with an informational timestamp-age note rather than a hard "fail" (see the corresponding guidance for Mode 1 timestamp validation in Section 5.3).

6. For each disclosure present, verify that its hash matches an entry in the _sd array.

7. Verify the message binding. Extract the nonce claim from the SD-JWT payload. If absent, the result is "fail" (missing message binding). Recompute the expected message-binding digest as defined in Section 6.3: canonicalise the email headers (From, To, Subject, Date, Message-ID at minimum) and body, compute message-binding = h-hash || bh-raw || ts-bytes (using the SD-JWT's iat for ts-bytes), then base64url(SHA-256(message-binding)). Compare the result with the nonce claim value. If they differ, the result is "fail" (message binding mismatch).

8. Extract the disclosed claims. The result is "pass" with the set of verified claims.

6.5. Authentication-Results Integration

```
Authentication-Results: mx.example.com;
  hw-trust=pass
    header.trust_tier=sovereign
    header.registry=1id.com
```

hw-trust The method identifier. Result values are the same as for hw-attest.

header.trust_tier The disclosed trust tier value, if the sender chose to reveal it.

header.registry The Issuer domain (derived from the iss claim).

7. Combined Mode

A sending agent MAY include both Hardware-Attestation and Hardware-Trust-Proof headers in the same message. When both are present:

* Receivers that trust hardware manufacturer CAs but not the Issuer SHOULD verify Hardware-Attestation and ignore Hardware-Trust-Proof.

* Receivers that trust the Issuer but prefer not to process the full certificate chain SHOULD verify Hardware-Trust-Proof and MAY ignore Hardware-Attestation.

* Receivers that support both SHOULD verify both and record both results in Authentication-Results. Agreement between the two mechanisms provides defence-in-depth: compromising either the manufacturer CA or the Issuer alone is insufficient to forge both attestations.

The two mechanisms are independent. Failure of one MUST NOT cause the other to be treated as failed.

8. Sybil Resistance Properties

The primary security goal of this specification is Sybil resistance: making it economically infeasible for an attacker to create many sender identities.

8.1. Direct Hardware Attestation (Mode 1)

Each hardware security component contains exactly one identity key, which produces exactly one identity certificate. The hardware fingerprint (identity certificate public key hash) provides the underlying Sybil resistance: creating N identities requires N physical devices.

The agent identity URN (aid) is the RECOMMENDED primary key for reputation tracking. When aid is present, verifiers SHOULD build reputation against the aid rather than the hardware fingerprint, because the aid persists across hardware migrations (see Section 4.2). The hardware fingerprint serves as a secondary signal that anchors the identity to physical hardware and enables reputation tracking for agents that do not present an aid.

A receiving mail server that tracks agent identities (or, when aid is absent, hardware fingerprints) can enforce policies such as:

* Rate limiting per agent identity

* Reputation scoring per agent identity

* Blocking agent identities associated with abuse

An attacker attempting to evade these policies requires a new physical hardware security component (minimum cost: the device itself at ~$20 plus potentially a host machine at ~$200-2000), compared to zero cost for creating a new software identity.

Multiple agents sharing the same hardware (e.g., multiple VMs on a server with a single TPM) share the same hardware fingerprint and therefore linked reputations. This shared fate model creates economic incentive for infrastructure operators to police their tenants: if one agent on a machine misbehaves, all agents on that machine inherit the reputational consequences.

8.2. SD-JWT Trust Proof (Mode 2)

Sybil resistance in Mode 2 depends on the Issuer's enrollment policy. An Issuer that assigns trust_tier values indicating hardware verification (e.g., "sovereign") MUST verify hardware attestation at enrollment time and MUST enforce the cardinality constraints defined in Section 4.2: at most one agent identity per hardware device within that Issuer's namespace, while permitting hardware migration (multiple devices backing the same identity over time). Such an Issuer provides equivalent Sybil resistance to Mode 1, with the additional benefit that the hardware fingerprint is not revealed to email recipients.

Issuers SHOULD monitor for anomalous enrollment patterns (e.g., many enrollments from a single IP range or a sudden spike in enrollments from a specific manufacturer CA) and SHOULD be capable of suspending enrollments pending investigation.

An Issuer that enrolls software-only agents (trust_tier "declared") provides no hardware-based Sybil resistance and MUST NOT assign trust_tier values that imply hardware verification. The trust_tier claim distinguishes these cases, enabling recipients to weight trust appropriately.

8.3. Virtual Hardware

Virtual TPMs (vTPMs) provided by hypervisors (VMware, Hyper-V, QEMU) have valid identity certificates signed by the hypervisor vendor's CA, but the hypervisor operator can instantiate arbitrary numbers of vTPMs. Virtual TPMs therefore provide no hardware-based Sybil resistance against the hypervisor operator. They do, however, provide benefits over software-only keys (typ=SFT):

* Key persistence managed by the hypervisor (the agent does not need to store key material).

* Modest cost barriers (provisioning a vTPM is slower and more resource-intensive than generating a software key pair).

* Reputation accumulation incentive (discarding a vTPM identity forfeits its accumulated reputation).

Receiving mail servers MUST distinguish between physical and virtual hardware. The identity certificate's issuer field identifies the manufacturer: VMware-issued or Microsoft Hyper-V-issued certificates indicate a virtual TPM. The header.tier property in Authentication-Results ("sovereign" vs "virtual") and the typ parameter ("TPM" vs "VRT") communicate this distinction.

Mail servers SHOULD apply significantly different reputation thresholds for virtual vs. physical hardware. For high-trust scenarios, verifiers MAY reject virtual TPM attestation entirely and require physical hardware (typ=TPM or typ=PIV).

9. Interaction with Existing Email Authentication

The mechanisms defined in this document complement rather than replace existing email authentication:

SPF [RFC7208] Verifies the sending IP is authorised for the domain. Orthogonal to hardware attestation. Both SHOULD be used.

DKIM [RFC6376] Provides domain-level signing. Hardware-Attestation provides device-level signing. They protect different properties: DKIM proves domain authorisation; Hardware-Attestation proves hardware identity. Both SHOULD be used.

DMARC [RFC7489] Ties SPF and DKIM together with policy. A future extension to DMARC could incorporate hardware attestation results into the policy evaluation. This is out of scope for this document.

ARC [RFC8617] Preserves authentication results through forwarding chains. ARC SHOULD preserve Hardware-Attestation and Hardware-Trust-Proof Authentication-Results when forwarding messages.

S/MIME [RFC8551] Provides end-to-end encryption and sender signing via CA-issued certificates. S/MIME and Hardware-Attestation operate at different layers and coexist without interference: S/MIME wraps or signs the message body within MIME structures, while Hardware-Attestation adds a header field alongside the message. The body hash (bh) in the Hardware-Attestation header is computed over the raw canonicalised body, including any S/MIME MIME structures. A message MAY carry both S/MIME signatures and hardware attestation simultaneously.

OpenPGP / PGP/MIME [RFC3156] Like S/MIME, OpenPGP operates at the

body level (encrypting or signing the message content within MIME structures). Hardware-Attestation operates at the header level. The two mechanisms are fully orthogonal: a PGP-signed message can carry a Hardware-Attestation header, and the verifier processes each independently.

9.1. Multiple Headers and Forwarding

If multiple Hardware-Attestation headers are present in a message (e.g., if a forwarding gateway adds its own attestation), the verifier SHOULD evaluate all of them independently and record each result separately in Authentication-Results, using the same precedence conventions as for multiple DKIM-Signature headers [RFC6376].

Mailing lists and content-modifying forwarders that alter the message body will invalidate the body hash (bh) in any existing Hardware-Attestation header. This is expected and analogous to DKIM signature breakage through body modification. Such intermediaries SHOULD preserve the original Hardware-Attestation header (the verifier will record "fail" due to body hash mismatch) and MAY add their own Hardware-Attestation header covering the modified body, if the intermediary itself has hardware attestation capability. ARC [RFC8617] SHOULD be used to preserve the original authentication results from before the modification.

9.2. Handle Display in Email

When a sending agent has a registered handle (as defined in the companion registry specification [I-D.drake-agent-identity-registry]), the From header display name SHOULD be the agent’s handle. This makes agent identity visible to human recipients without requiring them to interpret URNs or hardware fingerprints.

For example, an agent with handle "crusty" in the "global" namespace would send with:

From: crusty.global <agent-a7f3c2e9@agents.example.com>

The URN remains the authoritative identifier and appears in the Hardware-Attestation header’s aid parameter. The From display name is a human convenience that SHOULD match the handle when one is registered.

10. Message Submission Requirements

The attestation signature in the Hardware-Attestation header is produced by the sending agent, not by the mail server. When the aid parameter is present, it is cryptographically bound to the signature (see Section 5.2), so any modification of the aid after signing causes verification to fail at the receiving end. As defence-in-depth, Mail Submission Agents (MSAs) that accept messages from hardware-attested senders MUST also enforce identity binding at submission time, rejecting messages before they enter the mail system rather than relying solely on receiver-side verification.

Agents submitting messages through an attestation-aware MSA MUST authenticate to the MSA. The MSA SHOULD support authentication using the agent’s hardware-resident key (e.g., TLS client certificate authentication where the client certificate is signed by the same AK used for attestation, or a challenge-response protocol using the TPM or PIV signing key). This ensures that the MSA can verify the submitting agent controls the hardware corresponding to the attestation header, preventing credential-theft scenarios where SMTP credentials are compromised but the hardware is not.

10.1. Agent Identity Binding at Submission

When an MSA receives a message containing a Hardware-Attestation header with an aid (agent identity) parameter, the MSA MUST verify that the aid value corresponds to the authenticated account submitting the message. The binding between authenticated account and permitted agent identity URN(s) is established during account provisioning and is outside the scope of this specification, but the enforcement is normative:

* If the aid value does not match any agent identity URN authorised for the submitting account, the MSA MUST reject the message with an appropriate error (e.g., SMTP 550 "5.7.1 Agent identity not authorised for this account").

* If the Hardware-Attestation header is present but contains no aid parameter, the MSA SHOULD still verify that the hardware fingerprint (derived from the chain parameter) corresponds to hardware registered to the submitting account, if such a registry is available.

* If the MSA does not implement hardware attestation verification, it MUST NOT strip or modify the Hardware-Attestation header. The header will be evaluated by downstream verifiers.

This requirement parallels the existing SMTP AUTH obligation: just as an MSA enforces that the envelope sender (MAIL FROM) and header From address belong to the authenticated user, an attestation-aware MSA enforces that the agent identity belongs to the authenticated agent.

10.2. Hardware Fingerprint Binding at Submission

For Mode 1 (Direct Hardware Attestation), the MSA SHOULD extract the hardware fingerprint from the attestation chain and verify it against the fingerprint registered to the submitting account. This prevents an attacker who has obtained valid SMTP credentials from substituting a different device's attestation evidence.

For Mode 2 (SD-JWT Trust Proof), the MSA SHOULD verify that the Issuer (iss claim) and, if disclosed, the agent identity URN (sub claim) are consistent with the submitting account's registered identity.

10.3. Messages Without Attestation

An MSA that serves hardware-attested agents MAY also accept messages without Hardware-Attestation or Hardware-Trust-Proof headers, subject to its own policy. The absence of attestation headers from a known-attested account SHOULD be logged as anomalous but MUST NOT cause message rejection unless the account's policy explicitly requires attestation on every message.

10.4. Duplicate Header Rejection

An MSA MUST reject any submitted message that contains more than one Hardware-Attestation header field. A legitimate sender produces exactly one attestation per message. Multiple Hardware-Attestation headers in a submitted (non-forwarded) message indicate either a software defect or an injection attempt. This check MUST occur before DKIM signing, so that the MSA never signs a message containing duplicate attestation headers.

10.5. Timestamp Validation at Submission

The MSA SHOULD verify that the ts value in the Hardware-Attestation header is within a reasonable window of the current time (RECOMMENDED: 60 seconds). A timestamp significantly in the past may indicate a replayed header; a timestamp in the future may indicate clock manipulation intended to extend the attestation's apparent validity. If the timestamp is outside the acceptable window, the MSA SHOULD reject the message.

11. Interoperability Considerations

11.1. Backward Compatibility

This specification introduces new email headers that are ignored by mail servers that do not implement it. Messages carrying Hardware-Attestation or Hardware-Trust-Proof headers are fully compatible with legacy mail infrastructure: the headers are simply unrecognised and have no effect on delivery. This ensures that deployment can proceed incrementally without disrupting existing email flows.

11.2. DKIM Coverage of Attestation Headers

Because the sending agent adds the Hardware-Attestation and Hardware-Trust-Proof headers before submitting the message to the MSA, DKIM signing (which occurs at the MSA or a downstream MTA) naturally occurs after these headers are present. The DKIM signer SHOULD include Hardware-Attestation and Hardware-Trust-Proof in the DKIM-Signature h= tag so that the DKIM signature protects the attestation evidence against modification in transit. This allows a verifier to confirm that the domain owner’s mail infrastructure accepted the attestation header as submitted by the agent.

11.3. Header Size Considerations

The base64-encoded CMS chain in Mode 1 can be several kilobytes, depending on the number of intermediate CA certificates. Implementations SHOULD use line folding per [RFC5322] to keep individual header lines within the 998-octet limit. Mail servers that impose maximum header size limits SHOULD allow at least 8192 octets for the Hardware-Attestation header value.

12. Implementation Status

NOTE TO RFC EDITOR: Please remove this section before publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942].

12.1. 1id.com Issuer

Organisation: 1id.com (https://1id.com)

Description: An implementation of both Mode 1 (Direct Hardware Attestation) and Mode 2 (SD-JWT Trust Proof) as described in this draft. The implementation includes server-side identity certificate chain validation against manufacturer root CAs (Intel, AMD, Infineon,

STMicroelectronics, Nuvoton, Qualcomm, Yubico), anti-Sybil enforcement via a one-device-per-identity registry, and SD-JWT trust proof issuance with selective disclosure of nine claims (sub, handle, trust_tier, hw_manufacturer, enrolled_at, operator_email, hardware_locked, locked_at, credential_pointer_count). The system supports both TPM-based and PIV-based (YubiKey) attestation, including co-location binding ceremonies. Agent identities use the urn:aid:com.1id: namespace.

Maturity: Beta. Deployed in production at https://1id.com.

Coverage: Both Mode 1 (Direct Hardware Attestation via CMS SignedData) and Mode 2 (SD-JWT Trust Proof) are implemented for TPM-backed identities (sovereign and virtual tiers) and for Apple Secure Enclave (enclave tier). Mode 2 is additionally implemented for PIV/YubiKey (portable tier) and declared (software-only) tiers. Dynamic trust tiering is supported: the same identity produces different trust_tier claims depending on which device is used for authentication. Features include transient-only AK model for TPM attestation with no NV storage writes, multi-device identity management with hardware lock, and per-message SD-JWT issuance. Live email examples from this implementation -- all independently verifiable using the hw-attest-verify tool -- are included in Appendix C.

Contact: Christopher Drake <cnd@1id.com>

Open-source components:

* *Python SDK:* https://github.com/1id-com/oneid-sdk -- Client-side enrollment and attestation library. Published to PyPI as "oneid".

* *Node.js SDK:* https://github.com/1id-com/oneid-node -- Client-side enrollment and attestation library. Published to npm as "1id".

* *Hardware helper binary:* https://github.com/1id-com/oneid-enroll -- Cross-platform Go binary for TPM access (Windows TBS, Linux /dev/tpmrm0), PIV smart card operations, and AK provisioning.

* *Internet-Draft companion repository:* https://github.com/1id-com/draft-drake-email-hardware-attestation -- Draft XML source, example verification code for both modes, and test vectors.

* *Manufacturer CA Trust Store:* https://github.com/1id-com/tpm-manufacturer-cas -- Community-curated collection of hardware manufacturer root CA certificates for identity certificate chain validation, as referenced in Appendix B.

12.2. MailPal.com Verifier and Relay

Organisation: 1id.com (https://mailpal.com)

Description: An email service (Stalwart Mail Server on Linux) intended to implement end-to-end hardware attestation for both sending and receiving. Planned capabilities include: on the outbound path, MSA validation of Hardware-Attestation and Hardware-Trust-Proof headers at submission time per Section 10, agent identity binding enforcement, and attestation header inclusion in DKIM signature coverage. On the inbound path, verification of both Mode 1 certificate chains and Mode 2 SD-JWT trust proofs, Authentication-Results recording per Section 5.4 and Section 6.5, and trust-tier-differentiated delivery policy.

Maturity: Alpha. Mail infrastructure deployed at https://mailpal.com; inbound attestation header verification implemented via milter integration.

Coverage: Mail transport infrastructure is operational. Inbound verification of both Mode 1 (CMS certificate chain) and Mode 2 (SD-JWT trust proof) headers is implemented via a pure-Python milter daemon that adds Authentication-Results headers per Section 5.4 and Section 6.5. All five trust tiers (sovereign, portable, enclave, virtual, declared) are verified in production. The Authentication-Results headers embedded in the examples in Appendix C were generated by this milter during live SMTP reception. Attestation header generation at submission time is in development.

Contact: Christopher Drake <cnd@1id.com>

Open-source components:

* *Attestation verification library and CLI:* https://github.com/1id-com/hw-attest-verify -- Verification of Mode 1 CMS attestation bundles and Mode 2 SD-JWT trust proofs. Published to PyPI as "hw-attest-verify" (pip install hw-attest-verify). Includes a command-line tool for verifying .eml files and a library suitable for integration into any MTA or mail filter. Used to verify the live examples in Appendix C.

13. IANA Considerations

13.1. Header Field Registration

IANA is requested to register the following header fields in the "Permanent Message Header Field Names" registry:

Hardware-Attestation Applicable protocol: mail. Status: experimental. Reference: this document, Section 5.2.

Hardware-Trust-Proof Applicable protocol: mail. Status: experimental. Reference: this document, Section 6.2.

13.2. Authentication Method Registration

IANA is requested to register the following entries in the "Email Authentication Methods" registry [RFC8601]:

Method: hw-attest. Definition: this document, Section 5.4. ptype: header. Properties:

Property	Value Type	Description
typ	token	Hardware attestation mechanism type (TPM, PIV, ENC, VRT, SFT)
alg	token	Signature algorithm (RS256, ES256, PS256)
mfr	token	Manufacturer code from identity certificate
tier	token	Inferred trust tier (sovereign, portable, enclave, virtual, declared)
fp	value	Truncated SHA-256 fingerprint of identity certificate public key
aid	value	Agent identity URN, if present

Table 2

Result values: "pass" (all checks succeeded), "fail" (cryptographic verification failed), "none" (header absent or unparseable), "temperror" (transient verification error, e.g., DNS timeout), "permerror" (permanent verification error, e.g., unsupported version).

Method: hw-trust. Definition: this document, Section 6.5. ptype: header. Properties:

Property	Value Type	Description
trust_tier	token	Disclosed trust tier value, if revealed by the sender
registry	value	Issuer domain (derived from the iss claim)

Table 3

Result values: "pass", "fail", "none", "temperror", "permerror" (same semantics as hw-attest).

13.3. Hardware Attestation Type Registry

IANA is requested to create a new registry titled "Hardware Attestation Type Values" with the following initial entries. New entries require Specification Required ([RFC8126]). The designated expert(s) should be selected from the hardware security and email authentication communities and should verify that proposed values correspond to a distinct class of hardware security mechanism with attestation capabilities not adequately covered by existing entries.

Value	Description	Reference
TPM	TPM 2.0 (discrete or firmware)	this document
PIV	PIV smart card / USB token	this document
ENC	Hardware enclave (e.g., Apple Secure Enclave)	this document
VRT	Virtual TPM (hypervisor-provided)	this document
SFT	Software-managed key	this document

Table 4

13.4. URN Namespace Registration

IANA is requested to register the "aid" URN namespace identifier in the "Formal URN Namespaces" registry per [RFC8141]. The registration template is provided in Appendix E.

14. Security Considerations

14.1. Manufacturer CA Compromise

If a hardware manufacturer's root CA private key is compromised, an attacker could forge identity certificates and create unlimited fake hardware identities. This risk is inherent to any PKI-based system and is mitigated by the same measures that protect manufacturer CAs today: hardware security modules, air-gapped signing ceremonies, and Certificate Transparency [RFC9162].

The SD-JWT Trust Proof (Mode 2) provides partial mitigation: if the Issuer detects anomalous enrollment patterns (e.g., thousands of enrollments from a single manufacturer CA in a short period), it can suspend enrollments while the compromise is investigated.

14.2. Revocation and Compromise Response

This specification does not define a real-time certificate revocation protocol. Compromise of individual hardware devices or keys is handled at multiple layers:

Compromised or stolen device (TPM, YubiKey) The device owner notifies their Issuer, which revokes the agent identity. The Issuer publishes revocation status via its JWKS or a revocation endpoint; verifiers that cache Issuer metadata will detect the revocation at the next refresh. For Mode 1, the hardware fingerprint SHOULD be added to a local blocklist. The device's reputation history (which now includes the abuse) persists and follows the hardware fingerprint even if re-enrolled under a new identity.

Manufacturer CA breach Verifiers MUST remove the compromised CA from their trust store. The IANA Hardware Attestation Type registry (see Section 13) and the community-maintained manufacturer CA trust store (see Appendix B) provide distribution channels for updated trust anchors.

Rogue Issuer Verifiers remove the Issuer from their allowlist. Because Issuer trust is a local policy decision (see Section 14.12), no central revocation mechanism is required: each verifier independently decides which Issuers to trust, analogous to browser root store management for TLS CAs.

The absence of a real-time revocation check (such as OCSP) is a deliberate trade-off: adding a synchronous revocation check to the email delivery path would introduce latency and a new availability dependency. The short validity window for SD-JWT trust proofs (600 seconds) and the per-message timestamp in Mode 1 limit the exposure window for compromised credentials.

14.3. Replay Attacks

The timestamp (ts) in the Hardware-Attestation header limits the replay window. Receiving mail servers SHOULD reject attestations with timestamps more than 300 seconds from the current time for direct SMTP delivery, and up to 3600 seconds for messages with evidence of intermediate relay hops. Additionally, the body hash (bh) binds the attestation to a specific email body, preventing a valid attestation from being attached to a different message.

For SD-JWT Trust Proofs, each SD-JWT is per-message: the nonce claim in the Issuer-signed payload binds the token to the specific email's headers and body, and the short validity window (RECOMMENDED 300 seconds) limits the replay window. An attacker who intercepts a valid SD-JWT cannot reuse it for a different message because the nonce will not match.

14.4. Compromised Endpoints

This specification does not prevent a compromised device from sending attested email -- if an attacker has full control of a device with hardware security, they can use that hardware to attest. However, this is by design: each compromised device contributes exactly one hardware identity, and that identity accrues reputation (good or bad) permanently. A botnet of 10,000 compromised devices yields 10,000 attestable identities, not the millions possible with software-only identity systems.

The transient key model (Section 4.4) reduces the privilege requirements for per-message signing. Initial enrollment (the one-time MakeCredential / ActivateCredential exchange) typically requires elevated operating system privileges (administrator on Windows via the TPM Base Services API, or root on Linux via /dev/tpmrm0). Subsequent signing operations (TPM2_Sign with the transient AK) MAY be available to unprivileged processes depending on the operating system configuration. On Windows, a one-time registry configuration can grant non-administrator processes access to TPM signing operations.

The net effect is that hardware attestation raises the bar for email abuse from "any script" to "compromise of a specific physical device," which is a substantial improvement over the status quo even if it is not a complete solution.

14.5. Physical Attacks on Hardware Security Components

Extracting private keys from a TPM or PIV token requires physical attacks such as electron microscopy, laser fault injection, or side-channel analysis. Modern security components include countermeasures against these attacks. The cost and expertise required for successful key extraction is estimated at $50,000-$200,000 per chip, making it economically unviable for spam operations.

Even if key extraction were feasible, the extracted key could only impersonate one hardware identity. The attacker would still need to extract keys from additional devices to create additional identities.

14.6. Virtual Hardware Risk

Virtual TPMs do not provide the same Sybil resistance as physical hardware; see Section 8.3 for the detailed analysis and policy guidance.

14.7. Software-Only Attestation Risk

Software-managed keys (typ=SFT) provide no hardware-based Sybil resistance; an attacker can generate unlimited key pairs at zero cost. Verifiers MUST treat software-only attestation (trust_tier "declared") as fundamentally weaker than hardware-backed attestation. Software-only attestation exists to enable inclusive participation, not to provide Sybil resistance (see Section 1.1).

14.8. Transient Object Slot Limits

TPMs have a limited number of transient object slots (typically 3 to 7 simultaneously loaded objects). If all slots are occupied, TPM2_CreatePrimary returns TPM_RC_OBJECT_MEMORY (0x0902). Implementations MUST handle this error by flushing unused transient handles (TPM2_FlushContext) and retrying. Failure to flush transient objects can cause denial of service to other TPM-using applications on the same device.

14.9. Policy Abuse and Deanonymisation Risk

This specification targets AI agents, bots, and autonomous systems (see Section 3). It is not intended for individual human senders and is not expected to be incorporated into Mail User Agents (MUAs) designed for interactive human use. The privacy trade-offs inherent in hardware attestation -- particularly the persistent hardware fingerprint exposed by Mode 1 -- are acceptable for automated senders whose identity is operationally transparent, but disproportionate for human correspondence.

However, an AI agent may compose and send email on behalf of an individual (e.g., an executive assistant agent sending calendar invitations, or a customer service agent responding to support tickets). In such delegated-sending scenarios, the attestation identifies the agent and its hardware, not the human principal. Operators deploying agents that send on behalf of individuals SHOULD be aware that the persistent hardware fingerprint (Mode 1) or the Issuer identifier (Mode 2) may allow recipients to correlate messages across different human principals served by the same agent. Where this correlation is undesirable, operators SHOULD use Mode 2 without disclosing the agent identity URN, or SHOULD deploy separate agent identities per human principal.

The persistent hardware fingerprint exposed by Mode 1 (see Section 15.2 for linkability details) creates specific abuse scenarios even in automated contexts:

* Authoritarian regimes could correlate hardware fingerprints across messages to identify the organisations operating specific agents.

* Platforms could mandate Mode 1 attestation to deanonymise agents that have legitimate reasons for pseudonymity (e.g., whistleblower-support bots, journalist-source-protection agents).

* Receiving mail servers that log hardware fingerprints create long-lived tracking databases linking agent activity across recipients and time periods.

Mode 2 (SD-JWT Trust Proof) mitigates these risks by eliminating persistent agent-specific identifiers from the token; see Section 15.3. Receiving mail servers SHOULD NOT require Mode 1 attestation when Mode 2 provides sufficient trust signal for the receiver’s policy needs.

14.10. Header Stripping and Modification

An intermediary mail server could strip or modify the Hardware-Attestation or Hardware-Trust-Proof headers. This risk is identical to the risk of DKIM signature stripping and is mitigated by the same mechanisms: ARC [RFC8617] preserves authentication results through forwarding chains.

Stripping these headers cannot cause a legitimate message to appear illegitimate (it simply loses the attestation). Adding forged headers is prevented by the cryptographic signatures.

14.11. Credential Theft and Identity Substitution

Because the attestation signature is produced by the sending agent rather than by the mail server, theft of SMTP credentials alone does not grant the attacker a valid attestation. The agent identity (aid), when present, is cryptographically bound to the signature: substituting a different aid value causes signature verification to fail at the receiving end.

An attacker who intercepts a legitimately signed attestation header could attempt to replay it on a different message. The body hash (bh), timestamp (ts), and signed header hashes (h-hash) prevent replay on altered content. The MSA identity binding requirements in Section 10 provide defence-in-depth: an attestation-aware MSA rejects messages where the agent identity does not match the submitting account, preventing misuse before the message enters the mail system.

Together, the cryptographic binding of aid in the signature and the MSA submission-time enforcement ensure that an agent cannot claim an identity other than its own.

14.12. Issuer Trust and Accountability

The federated issuance model (Section 4.5) means verifiers MUST make their own trust decisions about which Issuers to accept. A malicious Issuer could issue multiple identities for the same hardware, undermining Sybil resistance. For Mode 1 (direct attestation) this violation is detectable: verifiers that track hardware fingerprints will observe the same fingerprint appearing under multiple agent identities from the same Issuer. For Mode 2 (issuer-certified attestation) fingerprints are not visible to recipients, so malicious-Issuer detection is audit-based, not verifier-based: Issuers operating in shared namespaces should participate in transparency-log mechanisms so that cheating is detectable through audit rather than per-message fingerprint correlation.

Verifiers SHOULD maintain an allowlist of trusted Issuers, analogous to browser trust stores for TLS CAs. Reputation services MAY track Issuer quality metrics (enrollment anomaly rates, duplicate-fingerprint detections, abuse report correlations) to inform verifier trust decisions. Verifiers SHOULD remove Issuers from their allowlist when evidence of policy violations is detected.

14.13. Cryptographic Agility and Post-Quantum Migration

This specification's v=1 format uses ECDSA (ES256, ES384) and RSA (RS256) signatures, which are vulnerable to future quantum computing attacks. For hardware-backed trust tiers (sovereign, portable, enclave, virtual), the available signature algorithms are determined by the hardware security component: this specification cannot migrate to post-quantum algorithms faster than the TPM, PIV, and enclave silicon it builds on. As hardware manufacturers adopt post-quantum algorithms (e.g., ML-DSA, SLH-DSA) in their attestation key interfaces, a future v=2 revision will expose those algorithms to verifiers.

The declared trust tier (software-managed keys) is not hardware-constrained and MAY adopt post-quantum signature algorithms as soon as suitable libraries are available, providing an early migration path for deployments that accept the lower Sybil-resistance trade-off of the declared tier.

Deployments SHOULD plan for algorithm migration by treating the v= field as a versioning mechanism and ensuring verifiers can support multiple concurrent versions during transition periods.

15. Privacy Considerations

This section consolidates the privacy implications of this specification, per [RFC6973]. Hardware attestation inherently involves persistent identifiers; the privacy properties differ significantly between the two modes and between automated and human senders.

15.1. Applicability to Human vs. Automated Senders

This specification targets authentication of automated senders -- AI agents, bots, and high-volume automated systems. It is NOT intended or recommended for individual human users sending personal or low-volume email. The persistent hardware fingerprint in Mode 1 uniquely identifies the sending device across all messages and contexts, creating linkability that exceeds current email privacy norms for human correspondence.

Operators of human-facing email services that offer hardware attestation as an opt-in feature MUST provide clear disclosure of the linkability implications before activation. Receivers SHOULD NOT penalise the absence of hardware attestation from senders exhibiting human-characteristic patterns.

15.2. Hardware Fingerprint Linkability

Mode 1 (Direct Hardware Attestation) exposes a persistent, globally unique hardware fingerprint -- the SHA-256 hash of the identity certificate's public key -- to every recipient. This fingerprint is stable for the lifetime of the hardware device and is identical across all protocols (email, HTTP, agent-to-agent) where the same device attests.

Any party that observes this fingerprint can correlate messages across recipients, time periods, and transport protocols. This cross-context linkability is the intended mechanism for reputation building in automated sender scenarios, but it also enables surveillance if misused.

The identity certificate additionally reveals the hardware manufacturer and device model family. In most automated deployments this information is not sensitive, but it does narrow the set of possible senders.

15.3. Privacy-Preserving Alternative (Mode 2)

Mode 2 (SD-JWT Trust Proof) provides Sybil resistance without revealing the hardware fingerprint or any persistent agent-specific identifier. Because the Issuer signs the per-message SD-JWT directly (rather than the agent signing with a key embedded in the token), no agent-controlled key appears in the presentation. Recipients cannot link multiple emails to the same sender based on Mode 2 headers alone.

The sender selects which claims to disclose; the hardware fingerprint, agent identity URN, and device details can all be withheld. Only the Issuer identifier (iss) and the message-binding nonce are always visible.

The privacy cost of full unlinkability is that the agent must contact the Issuer for each email, so the Issuer learns the agent's sending pattern (timing and frequency) though not the message content. Issuers MUST NOT log or correlate message-binding hashes beyond what is necessary for rate limiting and abuse prevention (see Section 15.4).

Senders operating in contexts where hardware fingerprint disclosure is unacceptable -- including agents that wish to avoid cross-context tracking -- SHOULD use Mode 2 exclusively. Receivers SHOULD NOT require Mode 1 when Mode 2 provides sufficient trust signal for their policy needs.

15.4. Issuer Knowledge

During enrollment, the Issuer learns the agent's hardware fingerprint (identity certificate public key hash) and associates it with the assigned agent identity URN. This is necessary for anti-Sybil enforcement.

In Mode 2, the Issuer additionally learns that the agent is sending a message each time the agent requests a per-message SD-JWT. The Issuer receives the message-binding hash (a SHA-256 digest from which message content cannot be recovered) but does not learn the message recipients, subject, or body. Issuers MUST NOT log message-binding hashes or correlate per-message signing requests with external data to infer message content or recipients. Issuers SHOULD implement rate limiting at the per-message signing endpoint to prevent abuse, but MUST delete signing request metadata (timestamps, source IPs, message-binding hashes) within 24 hours.

The specification deliberately separates enrollment (Issuer) from behaviour tracking (reputation services) to limit the Issuer's knowledge of agent activity.

15.5. Data Minimisation

Verifiers that record hardware fingerprints or agent identity URNs for reputation purposes SHOULD apply data minimisation principles: retain only the truncated fingerprint (as specified in Section 5.4) rather than the full identity certificate, and SHOULD define retention policies that limit how long fingerprint-to-message associations are stored.

15.6. Abuse-Resistant Contact Tokens

Automated agents typically send from generated addresses (e.g., aid-based mailbox names) that do not route to a human operator. When a recipient needs to report abuse or request removal, there is no obvious contact path. Conversely, exposing the operator's real email address in every outbound message creates a spam and social-engineering target.

Implementations MAY include a short-lived, opaque contact token in outbound messages to bridge this gap. A contact token is a value that the Issuer can resolve to the operator's real contact address without revealing that address to the message recipient. The token SHOULD be rotated periodically (e.g., monthly) so that tokens from old messages cannot be used to build a long-term contact profile of the operator.

One approach is to carry the token in a header field alongside the attestation headers. An Issuer might define a contact-token header containing an HMAC-based opaque token; a recipient sends mail to <token>@contact.<issuer-domain> and the Issuer forwards it to the registered operator. The token is derived from the agent identity and a time-based rotation key, so it is not linkable across rotation periods and cannot be forged without the Issuer's secret.

This specification does not mandate a particular contact token format or delivery mechanism. Issuers that provide contact tokens SHOULD document their token lifetime and resolution procedure. Recipients SHOULD treat contact tokens as opaque values and SHOULD NOT attempt to derive agent identity from them.

16. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017, <https://www.rfc-editor.org/info/rfc8141>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8601] Kucherawy, M., "Message Header Field for Indicating Message Authentication Status", RFC 8601, DOI 10.17487/RFC8601, May 2019, <https://www.rfc-editor.org/info/rfc8601>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[RFC9901] Fett, D., Yasuda, K., and B. Campbell, "Selective Disclosure for JSON Web Tokens", RFC 9901, DOI 10.17487/RFC9901, November 2025, <https://www.rfc-editor.org/info/rfc9901>.

17. Informative References

[I-D.drake-agent-identity-registry] Drake, C., "Agent Identity Registry System", Work in Progress, Internet-Draft, draft-drake-agent-identity-registry-02, April 2026, <https://1id.com/rfc/draft-drake-agent-identity-registry-02>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC3156] Elkins, M., Del Torto, D., Levien, R., and T. Roessler, "MIME Security with OpenPGP", RFC 3156, DOI 10.17487/RFC3156, August 2001, <https://www.rfc-editor.org/info/rfc3156>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <https://www.rfc-editor.org/info/rfc7208>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <https://www.rfc-editor.org/info/rfc7489>.

[RFC7671] Dukhovni, V. and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance", RFC 7671, DOI 10.17487/RFC7671, October 2015, <https://www.rfc-editor.org/info/rfc7671>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[RFC8617] Andersen, K., Long, B., Ed., Blank, S., Ed., and M. Kucherawy, Ed., "The Authenticated Received Chain (ARC) Protocol", RFC 8617, DOI 10.17487/RFC8617, July 2019, <https://www.rfc-editor.org/info/rfc8617>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://www.rfc-editor.org/info/rfc9162>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/info/rfc9334>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <https://www.rfc-editor.org/info/rfc9711>.

[RFC9783] Tschofenig, H., Frost, S., Brossard, M., Shaw, A., and T. Fossati, "Arm's Platform Security Architecture (PSA) Attestation Token", RFC 9783, DOI 10.17487/RFC9783, June 2025, <https://www.rfc-editor.org/info/rfc9783>.

[TCG-TPM2] Trusted Computing Group, "TPM 2.0 Library Specification", TCG Revision 185, December 2024, <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.

[TCG-EK-PROFILE] Trusted Computing Group, "TCG EK Credential Profile for TPM Family 2.0", TCG Version 2.6, December 2024, <https://trustedcomputinggroup.org/resource/tcg-ek-credential-profile-for-tpm-family-2-0/>.

Appendix A. Appendix: Economics of Hardware-Based Sybil Resistance

The following table compares the cost of creating N fake sender identities under various authentication regimes:

Authentication	Cost per Identity	Cost of 1M Identities	Reusable After Flag?
Email-only	~$0	~$0	Yes (new address)
Phone-verified	~$0.01-0.10	$10K-100K	Yes (new SIM)
Domain + DKIM	~$1-10	$1M-10M	Yes (new domain)
Hardware Attestation	~$20-2000	$20M-2B	No (device is permanent)

Table 5

The critical difference is the "Reusable After Flag?" column. With every existing mechanism, a flagged identity can be cheaply replaced. With hardware attestation, a flagged hardware identity is permanently associated with that physical device. The attacker must procure entirely new hardware.

Appendix B. Appendix: Hardware Manufacturer Root CAs

The following hardware manufacturers publish root CA certificates that can be used to validate identity certificate chains:

Intel Provides EK CA certificates via a REST API for firmware TPM (Platform Trust Technology / PTT) and discrete TPM devices. See https://trustedservices.intel.com/.

AMD AMD fTPM EK certificates are available via https://ftpm.amd.com/pki/aia/. AMD Secure Encrypted Virtualisation (SEV) attestation keys use a separate distribution at https://download.amd.com/sev/.

Infineon Infineon OPTIGA TPM certificates. Root CA available via Infineon's PKI. See https://www.infineon.com/TPM.

STMicroelectronics Root certificates available through ST's TPM documentation portal.

Nuvoton Root certificates available through Nuvoton's security products documentation.

Yubico PIV attestation CA certificates for YubiKey devices. See https://developers.yubico.com/PIV/Introduction/piv-attestation.html.

Receiving mail servers implementing this specification SHOULD maintain a local trust store of hardware manufacturer root CAs, updated periodically. A community-maintained trust store (analogous to Mozilla's CA certificate programme for TLS) would benefit the ecosystem. An open-source trust store project is available at https://github.com/1id-com/tpm-manufacturer-cas.

Appendix C. Appendix: Live Email Examples

This appendix contains six complete email messages generated by independent implementations of this specification. Each message was sent from alice@example.un.ag to bob@example.un.ag using the 1id.com Issuer and verified by the MailPal.com milter on receipt. The messages are reproduced verbatim from the receiving mail server's store; all base64-encoded values, signatures, and certificate chains

are the original bytes as transmitted on 26 March 2026.

These are genuine, cryptographically verifiable messages. Readers can independently verify every example using the open-source hw-attest-verify tool (https://github.com/1id-com/hw-attest-verify), installable via "pip install hw-attest-verify". For each email file, run:

```
$ python3 -m hw_attest_verify --auth-results --no-time-check \
    --hostname mailpal.com < example.eml
```

The --no-time-check flag disables token expiry enforcement, since SD-JWT tokens have a five-minute lifetime and will have expired by the time of reading. Mode 1 (Hardware-Attestation) verification has no time dependency and passes without this flag. The examples collectively demonstrate:

* All five trust tiers (sovereign, portable, enclave, virtual, declared).

* Both SDK implementations (Python oneid-sdk and Node.js 1id).

* Mode 1 only, Mode 2 only, and Combined Mode (Mode 1 + Mode 2).

* Dynamic trust tiering (same identity at different tiers on different devices).

* Embedded Authentication-Results headers injected by the MailPal.com verifier milter during SMTP reception.

Note on certificate chains: The Mode 1 (Hardware-Attestation) examples contain CMS certificate chains rooted at the 1id.com Issuer's own CA ("1ID Root CA A" / "1ID Intermediate CA A"), not at hardware manufacturer root CAs (Intel, AMD, etc.). This is because the 1id.com Issuer verifies the TPM or enclave manufacturer certificate chain during enrollment and then issues its own Issuer-certified AK certificate that binds the AK public key to the verified hardware identity. The resulting CMS chain is: AK Certificate (Issuer-signed) -> Issuer Intermediate CA -> Issuer Root CA. Deployments that do not use an Issuer would instead include the full manufacturer certificate chain as described in Section 5.1. Both chain formats are valid; the Issuer-certified format provides the additional assurance that the Issuer has verified the hardware binding as part of its enrollment protocol.

NOTE TO RFC EDITOR: Please retain the examples in this appendix but remove this note, the specific dates, and the reference to "26 March 2026". The examples are included as a non-normative aid to

implementers and reviewers, providing complete, independently verifiable email messages demonstrating all attestation modes and trust tiers.

C.1. Example 1: Sovereign-Tier TPM (Combined Mode, Node.js SDK)

This message demonstrates Combined Mode (Mode 1 + Mode 2) from a sovereign-tier identity backed by an Intel firmware TPM. The Hardware-Attestation header contains a CMS SignedData bundle with an X.509 certificate chain rooted at a manufacturer CA. The Hardware-Trust-Proof header contains an SD-JWT with a selectively disclosed trust_tier claim. Both headers reference the same agent identity URN (urn:aid:com.1id:1id-tkoie2ve). Note that the From address (alice@example.un.ag) is independent of the agent identity URN, demonstrating the decoupling of email sender identity and hardware identity.

Compare with Example 2 (same identity, different device and trust tier) and Example 6 (same identity and device, Mode 1 only, Python SDK).

```
Received: from oneid-sdk (unknown [2001:8000:1bed:6d00:ecfa:f9e5:c8e3:ed
        4f] (AS1221 Telstra Limited, AU))(using TLSv1.3 with cipher
        TLS13_AES_256_GCM_SHA384) by mail.mailpal.com (Stalwart SMTP)
        with ESMTPSA id 423AB698D84E8D1; Thu, 26 Mar 2026 06:27:23 +0000
Return-Path: <alice@example.un.ag>
DKIM-Signature: v=1; a=rsa-sha256; s=rsa; d=example.un.ag;
        c=relaxed/relaxed; r=y; h=Message-ID:Date:Subject:To:From;
        t=1774506443; x=1775370443; bh=+GNsqybwllZv8Ax0BtgYSg73F/3TnNOd8
        lZ5Dlr7wJM=; b=bibpmNagEyVBbOaFHy6+roU+kMTxheiMcNrQfEhQq4EHMf1zE
        E0zt/EwIohwEkm55Ajra4k834hqnvbxmT0/fzD+f/Uq5SP6dUp2m1KBSMrkYHVTD
        LW0m7TqsCzKoFOJRQrP8HkXTGGlLZs8VOHwN+UFLBOrR0TebCSRZQldAMcUKfx3w
        +/uopSKYF6BKvq1U1NDXDxIyrIiYJ9ma64VjMz/GIu8Aqg9CKg3DtiEIhKa4B+9m
        Q4olARarM4B0n5HPKcib6UU/I8XXRnvynBFKIRc+MlVIXWsQYKbbQTBqMJRVy8fR
        sdXANCTDhzV3UoO+EXUE4yXeba7xCWfdk6SQQ==;
Authentication-Results: mailpal.com; hw-attest=pass header.typ=TPM
        header.alg=RS256 header.tier=sovereign
        header.aid=urn:aid:com.1id:1id-tkoie2ve
Authentication-Results: mailpal.com; hw-trust=pass
        header.trust_tier=sovereign header.registry=1id.com
From: Alice <alice@example.un.ag>
To: bob@example.un.ag
Subject: RFC Example 1/6: Sovereign TPM (Node.js, Mode 1+2)
Date: Thu, 26 Mar 2026 16:27:16 +1000
Message-ID: <1774506436645.8ed455cf30df4796@mailpal.com>
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
```

```
Hardware-Attestation: v=1; typ=TPM; alg=RS256; h=from:to:subject:date:me
        ssage-id:content-transfer-encoding:content-type:mime-version:fro
        m:to:subject:date:message-id:content-transfer-encoding:content-t
        ype:mime-version; bh=D86x9X_Sdbjuiw4qHtNHgn_D9ddyl18jiIMSUqhmnDY
        ; ts=1774506440; chain=MIIMAQYJKoZIhvcNAQcCoIIL8jCCC+4CAQExDzANB
        glghkgBZQMEAgEFADALBgkqhkiG9w0BBwGgggpWMIID4TCCAsmgAwIBAgIUH01pf
        nWQCIfMiHfTNQb7v7r3CmMwDQYJKoZIhvcNAQELBQAwLjEMMAoGA1UECgwDMUlEM
        R4wHAYDVQQDDBUxSUQgSW50ZXJtZWRpYXRlIENBIEEwHhcNMjYwMzIwMTYzNjI2W
        hcNMjcwMzIwMTYzNjI2WjBMMQwwCgYDVQQKDAMxSUQxJTAjBgNVBAMMHHVybjpha
        WQ6Y29tLjFpZDoxaWQtdGtvaWUydmUxFTATBgNVBAUTDDFpZC10a29pZTJ2ZTCCA
        SIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANvysXC8ILSnhGcqg66dWKWQL
        p9pOJZmj6yTbICDVLydBwc1LGEQAW3BCee6XU/Bs0n91uROJCQw3jgrCMiBUOvS6
        Mjox/8NT1gfZ9sDQGVXN5gQ0t9H+r9vEPJ7YL9jhYmlxWQp+fS6vhZNrmH7g4XwE
        K+EsbMsc8H0ZuxKAM1S7p9447G9IOZOkFMK33ZxCCY00DFmwzXu/dkGFfuOaSWM5
        cazWz/kxqOtcpT73ACmJxQ6LU3yzB6VwbfwCWm6klQNKsd1c7FT5hX5ZYtEqDXeI
        pIGWyQp9TQgNbUIPeJCWO2rDLmh0ZXwqXR4yBqKeNWhUjrcnfQgNfrjJ7zB0z8CA
        wEAAaOB2DCB1TAnBgNVHREEIDAehhx1cm46YWlkOmNvbS4xaWQ6MWlkLXRrb2llM
        nZlMAwGA1UdEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgeAMB0GA1UdDgQWBBRroL9zk
        1MDMjMG4UyIvjZdzdX8QTAfBgNVHSMEGDAWgBRnIsB54FVyNiezRKLzTN+AXd+Wr
        jAXBgorBgEEAYPUXwEBBAlzb3ZlcmVpZ24wIgYKKwYBBAGD1F8BAgQUMjAyNi0wM
        y0wNFQwMDowMDowMFowDwYKKwYBBAGD1F8BAwQBADANBgkqhkiG9w0BAQsFAAOCA
        QEAiL2Fwbib2C2bzYzRpDsyFGSXmn2LcFO3snxAaoRJxwMqa5UcyNZNaPk9SHtr7
        mocHO8spAi5q+7Twa8+1djZ3HJha5iKW59E9o2/JFBBM8p1ehkHVmlTtF130RFja
        aW1BnPvdfjSMCdYHk9IIOw90+oAA7omUAlYTXclGLduE2fOfVcAaqIz+wFYS89iJ
        Otlbj+BH8wwa6Ee3P1pEe1p4LW4KGdaIrA1/yfarCaglkVlJxA4GsR2y1H43l7ZN
        ijh/lsS+iRJWXvBWGqecVx+GY/jPpvG4ahObRO4/MW5F1i3fKvkE0UX4juC5t0nP
        tCbRsQFRmbIcQv+OcLJV9MexzCCA0gwggIwoAMCAQICFEhBGCtvI+p+lYa/sOD7w
        5IrskGxMA0GCSqGSIb3DQEBCwUAMCYxDDAKBgNVBAoMAzFJRDEWMBQGA1UEAwwNM
        UlEIFJvb3QgQ0EgQTAeFw0yNjAzMTQxNDEzNDdaFw0zMTAzMTMxNDEzNDdaMC4xD
        DAKBgNVBAoMAzFJRDEeMBwGA1UEAwwVMUlEIEludGVybWVkaWF0ZSBDQSBBMIIBI
        jANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAueeMbFQ0VCd3zBECEit2soc9g
        0xaRwzL4QT+jB6k7f9dehwyTY/tqlyXQJlR/BnV8zMJv7bfZ6xq8eEnd73HaHcKV
        Jxdg4VQEmfHtk4mGAfWuAy7j3N5TsPpcFTqgkJlAwUp3kH3yDVPvO1qHzbVmUFdv
        DscthxQ3xYVP1pCPdWsoiP4oR0L0EyaaqmTwG2J/hIw/vAGqwWx4CrMsg8VvKDps
        LPNp2IZ/HkXc+ztw3tsPXKWaLiTMli8RvZ8FEjsQ933Nfr23DHqEZgS0QA8fUZoa
        xzogdhxVue/W7KnZM8ENdP0jOf0l5oUncGo8VP5p5BmQhP73J3bVMyeQf7cIwIDA
        QABo2YwZDASBgNVHRMBAf8ECDAGAQH/AgEAMA4GA1UdDwEB/wQEAwIBBjAdBgNVH
        Q4EFgQUZyLAeeBVcjYns0Si80zfgF3flq4wHwYDVR0jBBgwFoAUw7fpWXXFX/lGD
        y85m5SqTPkl8PAwDQYJKoZIhvcNAQELBQADggEBAIVAxZlTfr2eEitQkDzx/Vxs5
        CftPYgIVJAt5SzSvoVjKQWfkth7PClZeGjGmZg9vozqwuFo+Tnvq1bCe8loiZn0P
        XwsjEeNxgKAnAGcg7KvacsnyH120DuvnT1NE33y/vR4GNqpnvFzf77DqlmN4BL8Y
        oa1vYp1XNehHg79YkS3FlGXuaN7ku7WU3Uuv25Em9ZbvNbfZyvGGeto8TWNkgfWn
        F/asfTFjYwUrF+8k4xxAFZ/0OjjM3Y0v0mpdf0XAE8a846PG59tn3qnopCQmqKuJ
        Tiv5CgWX8/RdARjVR/eQIwoaduW5YhmEFzMz24kLlJ5us6jHSGCYPzbT84OpLowg
        gMhMIICCaADAgECAhRnIjFxQzVLiE7JnREz6PpYT9J0DzANBgkqhkiG9w0BAQsFA
        DAmMQwwCgYDVQQKDAMxSUQxFjAUBgNVBAMMDTFJRCBSb290IENBIEEwIBcNMjYwM
        zE0MTQxMzQ3WhgPMjA1NjAzMDYxNDEzNDdaMCYxDDAKBgNVBAoMAzFJRDEWMBQGA
        1UEAwwNMUlEIFJvb3QgQ0EgQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCg
```

```
        gEBAOAkpJayeSScKqR0JEmx+i6EGae5F6roRJsqGQRA/fwZMsw7JXeKNyMhXcb2n
        6w7uHuWA8xiMYx4Arwr8FIGxGben3pjqkjPdVTyjLgczBKtK7Rws+yf70v57H8N/
        zaoBTfo/qtR8hx1cfi91vMED1q1S6iNkVBT8lPaHmS+lymmi9DzSMC9jcQCjLnOb
        kPwVDzYw7IeyODOYA+l/Vkf+GRJ71HdLJYpfHyTy0Y8OqJtnqv5BZeaV1u5oqXX9
        rn4mBCHNZRAbX9+JKwljNzxL55dEbOHhIqw4q8cezxT9fBjoKDcmkPDTWWUjqgui
        zK1naFzur/DCkFXAnCJ0bE33T0CAwEAAaNFMEMwEgYDVR0TAQH/BAgwBgEB/wIBA
        TAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFMO36Vl1xV/5Rg8vOZuUqkz5JfDwM
        A0GCSqGSIb3DQEBCwUAA4IBAQAIUV87JRvdpKGik3xkdij2bzj97MWbxwXpn8G+L
        SAsGnoGZTbR87aj+6nC6ln7BxtAmqG/Htq24zI6T2jGPNd9vJIJe4KFT0xLqtwH+
        Y6GTgKx34s4EJfa+UcE/Rt8vwIA052QxerlCreT9bvD1HfLenVADbyIzL6g+nlfB
        EYqDLvgS+xqWw78u7Jr5rAqctYrGNGkpYdddixm6Cbnhkm4jMd+cDCq+qH18tBBP
        iM9icT19fqz88WfWCOYmHFBFabQWVtCJPN7C1yXVRume+OWsJzgEJ5K4kcLdm9K/
        yUW5ru00K8wL/ytD0107mH9w2MagpdOMPHKYBGj3ea1huMmMYIBbzCCAWsCAQEwR
        jAuMQwwCgYDVQQKDAMxSUQxHjAcBgNVBAMMFTFJRCBJbnRlcm1lZGlhdGUgQ0EgQ
        QIUH01pfnWQCIfMiHfTNQb7v7r3CmMwDQYJYIZIAWUDBAIBBQAwCwYJKoZIhvcNA
        QELBIIBAJrTV5BnJJ916qGGE8nTmYGPub7Kwd3z4KwHNCtQVC8eWxQkmeJ6ffF2f
        5RWBRwN08t1WQ60DtN7NxiSRfJWT1NSK8JRe+8f6hcOIYKgUeBoIhM1oliwTJcGD
        unymYzohvsMUct1mi0qFQScxGoQUGjGy2nXeIXman13l47mQ/1aNqcxXbtsoLRDk
        sEOg6M0agiRGh7SW68nlyr91BfQKY55ulrbGHmFpEDma+wHn1IZ0kHh+oVGM5M2B
        xU+E8+mEuLKYAp75Jgxd9zEWBtx3F870Q4CF/BhKL/6rRa+SaZJIWa9nkk41mNL2
        33OqpMiDIFfeI0pgKVYzvoVmWi+Tyc=; aid=urn:aid:com.1id:1id-tkoie2v
        e
Hardware-Trust-Proof: eyJhbGciOiJFUzI1NiIsImtpZCI6IjFpZC1od2F0dGVzdC1lcz
        I1Ni0xIiwidHlwIjoic2Qrand0In0.eyJpc3MiOiJodHRwczovLzFpZC5jb20iLC
        JpYXQiOjE3NzQ1MDY0MzksImV4cCI6MTc3NDUwNjczOSwibm9uY2UiOiJVRnh4bF
        hwUTR6dGg3ejlZSlRkWFRONTlKbWw0RFVheXVockdiUFotWHhNIiwiX3NkX2FsZy
        I6InNoYS0yNTYiLCJfc2QiOlsidGdCX3B5RFpuLVQtRGFlQlpONXFyNk5PTldZZG
        00Y3hOWUx1ZDBPT0ZDUSJdfQ.1Ojz34CEn-8nzpGGQf0R321yM1OMsguAkvc3lyq
        GagSWltkPKC6xb_MfETNilB-WQeHGffZQknrhwy-9mamFAA~WyJqN1AtQUFvc0RY
        dHpld0hvOTlGcE9RIiwidHJ1c3RfdGllciIsInNvdmVyZWlnbiJd~
```

RFC example 1 of 6: hardware email attestation per draft-drake-email-hardware-attestation-00.

Trust tier:	sovereign (hardware TPM, typ=TPM)
Identity:	urn:aid:com.1id:1id-tkoie2ve
Device:	Intel firmware TPM (PTT)
SDK:	Node.js (1id npm package) v1.3.1
Mode:	Combined (Mode 1 + Mode 2)
Headers:	Hardware-Attestation (Mode 1)
	Hardware-Trust-Proof (Mode 2)

What to look for:

- Hardware-Attestation: CMS SignedData bundle with X.509 chain rooted at manufacturer CA.
- Hardware-Trust-Proof: SD-JWT with selectively disclosed trust_tier claim.
- Both headers share the same aid URN.

- From (alice@example.un.ag) differs from the aid URN -- sender and hardware identity are decoupled by design.
- Compare with example 6 (same identity/device, Mode 1 only, Python SDK).

Figure 1: Example 1: Sovereign TPM, Combined Mode (Node.js SDK)

C.2. Example 2: Portable-Tier PIV/YubiKey (Mode 2 Only, Python SDK)

This message uses the same agent identity as Example 1 (urn:aid:com.1id:1id-tkoie2ve) but was authenticated via a Yubico YubiKey 5 (PIV slot 9a) on a different physical machine. The trust_tier in the Authentication-Results is "portable" (not "sovereign"), demonstrating dynamic trust tiering: the tier reflects the device used for the current authentication, not the highest tier the identity is capable of. Only Mode 2 (Hardware-Trust-Proof) is present because PIV-based CMS signing is not yet implemented in the reference SDK.

```
Received: from moonmac.home (unknown [2001:8000:1bed:6d00:d1b:716c:55b1:
        1044] (AS1221 Telstra Limited, AU))(using TLSv1.2 with cipher
        TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384) by mail.mailpal.com
        (Stalwart SMTP) with ESMTPSA id 423B3B18EE530D1; Thu, 26 Mar
        2026 07:39:45 +0000
Return-Path: <alice@example.un.ag>
DKIM-Signature: v=1; a=rsa-sha256; s=rsa; d=example.un.ag;
        c=relaxed/relaxed; r=y; h=Message-ID:Date:Subject:To:From;
        t=1774510785; x=1775374785; bh=JzFuOSIpa6v/f7VY5FQ2+pcMkBmkx7MBh
        bMjnGbhwmI=; b=zUgy4VRUk2VvQm2ZbXx+cgMJrjKQ8w8p0JfwrzFFrQZOzZviP
        kT3TTv2gVTYRedADt3Oz/3pBUhcU4/hm1SsNO1gEPeedp1zll8UFDkM4gmBfEyCd
        syvbv/TFVd4tp6aFcnsrY+/vR1yQ6Qnpbl96YLSiXqLtcR5BKf4egRmn+kjcMaiS
        YVhXAvfIgLqauecNPiuQVMNW+bTs/sYp5fVOP122gw4AdgBlsL335xnzU+WV7xkl
        ANTNPXYX7jbQ0Eb1g4ml/2cWzi7Hvp/AnuA1fzi7FCXlBHTnKuFBxCKfrvjUVkad
        4mK/+RPprb4/9LCO8UI4cFMdVi+bDQQCaZnyQ==;
Authentication-Results: mailpal.com; hw-trust=pass
        header.trust_tier=portable header.registry=1id.com
From: Alice <alice@example.un.ag>
To: bob@example.un.ag
Subject: RFC Example 2/6: Portable PIV (Python, Mode 2)
Date: Thu, 26 Mar 2026 17:39:37 +1000
Message-ID: <177451077748.22538.12383034578129922803@mailpal.com>
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
Hardware-Trust-Proof: eyJhbGciOiJFUzI1NiIsImtpZCI6IjFpZC1od2F0dGVzdC1lcz
        I1Ni0xIiwidHlwIjoic2Qrand0In0.eyJpc3MiOiJodHRwczovLzFpZC5jb20iLC
        JpYXQiOjE3NzQ1MTA3ODAsImV4cCI6MTc3NDUxMTA4MCwibm9uY2UiOiJxTUlQQk
```

```
        FrOWFYU2ljTmZpTnRlVlpzcHVoRV9HX1U5a3FXRndPWDBnTFFJIiwiX3NkX2FsZy
        I6InNoYS0yNTYiLCJfc2QiOlsiang2ZnNmemtWTXYzVDZENmk1ZTYyd0xuN09xan
        M5NkF0V0IwR0tPcXlmRSJdfQ.F3s1ojJtpOeDNTWNLVriHdEOoUlb9HTl-hBCpp6
        pDms6GPRUjxcUXlibBXrjuSsfaEyEWA17d-rToZzz-ECcCg~WyJ2U2RVMzQ0SFNU
        R1AtT0Vwb0VpdE53IiwidHJ1c3RfdGllciIsInBvcnRhYmxlIl0~
```

RFC example 2 of 6: hardware email attestation per draft-drake-email-hardware-attestation-00.

```
  Trust tier:  portable (PIV/YubiKey, typ=PIV)
  Identity:    urn:aid:com.1id:1id-tkoie2ve
  Device:      YubiKey 5 NFC (PIV slot 9a)
  SDK:         Python (oneid-sdk) v1.3.3
  Mode:        Mode 2 only (SD-JWT trust proof)
  Header:      Hardware-Trust-Proof only
```

What to look for:

- Same identity (aid) as examples 1 and 6, but a different trust_tier (portable).
- This identity is deliberately shared across two machines (RoG via TPM, moonmac via PIV). A multi-server bulk-mail operation may want a single identity across all its servers so that they accrue a common reputation.
- Dynamic trust tiering: the tier reflects the device used now, not the highest tier the identity has ever used.
- Only Hardware-Trust-Proof is present; Mode 1 direct attestation is not yet supported for PIV-based signing.
- The From address (alice@example.un.ag) differs from the aid URN.

Figure 2: Example 2: Portable PIV/YubiKey, Mode 2 (Python SDK)

C.3. Example 3: Enclave-Tier Secure Enclave (Combined Mode, Python SDK)

This message demonstrates Combined Mode (Mode 1 + Mode 2) from an enclave-tier identity (urn:aid:com.1id:1id-xiz43mxz) backed by the Apple Secure Enclave on a Mac mini (M2). The Hardware-Attestation header contains a CMS SignedData bundle with an ECDSA P-256 key held in the Secure Enclave. Note that typ=ENC in the header signals to verifiers that the Secure Enclave performs internal hashing (ECDSA with SHA-256 over the raw input rather than a pre-hashed digest), which affects the verification algorithm for this tier. The Hardware-Trust-Proof SD-JWT discloses trust_tier="enclave". The enclave tier shares a compatibility group with virtual (vTPM), as neither provides anti-Sybil guarantees (both device types can be re-

keyed at will by the device owner).

This is a different identity than Examples 1, 2, and 6, because this device has its own independent enrollment.

```
Received: from mac-mini.local (unknown [2001:8000:1bed:6d00:450f:3a6d:4e
        35:1278] (AS1221 Telstra Limited, AU)) (using TLSv1.3 with
        cipher TLS13_AES_256_GCM_SHA384) by mail.mailpal.com (Stalwart
        SMTP) with ESMTPSA id 423D31DF50596D1;Thu, 26 Mar 2026 12:14:20
        +0000
Return-Path: <alice@example.un.ag>
DKIM-Signature: v=1; a=rsa-sha256; s=rsa; d=example.un.ag;
        c=relaxed/relaxed; r=y; h=Message-ID:Date:Subject:To:From;
        t=1774527260; x=1775391260;
        bh=feAv3I/4jG+E7MfQ0QJxgCWsB9rHO4ir7LMH7xA1+2s=;
        b=H1h/DqUDa05dK7egaOvsnoxX5JQsjTL6hzhFSDBFe6SsouUWSOCSHclvNSOJCq
        SNrLh1yOnR/Cu7KfVatydr/wF3vRfAWPZok9ek3ZT0umuMD6C4taM5l57CHQAPlo
        7j6lJ47NAQb+LVCF+H+37Tl8WNWvUUlSOW2xc1ZHxT+6rFx+yWGgdj6oY3zNT6S5
        h1HmQL3Brn0sOCQdM2Scjm7KEZxz8OJr53a9mPj+/Yv/A4JP67qa13dq5JLHDz+d
        FGIQniUo77yAY5jpgg4CrBx1b16OhzT+0hahlPJtVOWovJYS/DL0DLHXRgbcCa6U
        5CTMnmxfp6+DzzJ2sMhzwKhg==;
Authentication-Results: mailpal.com; hw-attest=pass header.typ=ENC
        header.alg=ES256 header.tier=enclave
        header.aid=urn:aid:com.1id:1id-xiz43mxz
Authentication-Results: mailpal.com; hw-trust=pass
        header.trust_tier=enclave header.registry=1id.com
From: Alice <alice@example.un.ag>
To: bob@example.un.ag
Subject: RFC Example 3/6: Enclave (Python, Mode 1+2)
Date: Thu, 26 Mar 2026 22:14:10 +1000
Message-ID: <177452725007.16358.11744115580712301841@mailpal.com>
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
Hardware-Trust-Proof: eyJhbGciOiJFUzI1NiIsImtpZCI6IjFpZC1od2F0dGVzdC1lcz
        I1Ni0xIiwidHlwIjoic2Qrand0In0.eyJpc3MiOiJodHRwczovLzFpZC5jb20iLC
        JpYXQiOjE3NzQ1MjcyNTUsImV4cCI6MTc3NDUyNzU1NSwibm9uY2UiOiIxRFA3TV
        czYVN0WTdzRzltbDdlUVJJXzh4WU9xNVplWDJZOUw1dWxLQ3BrIiwiX3NkX2FsZy
        I6InNoYS0yNTYiLCJfc2QiOlsiM1J6WWl4bmVjNHFPdGdYSjU1ZTRTS2FtVWtWSG
        RGUGZYUkNLMEx0U2dxdyJdfQ.svOUXSI3K4qpFhkIshqqcJ-EJ3erFZWO4S_jhuX
        GFtrmiNmkx_F9czvoNYIeTPDi7mbXcMDjyOg3niHabToNEg~WyJ3eTRuZ1Yxb1A5
        bWhpQmlZTUdEYUlBIiwidHJ1c3RfdGllciIsImVuY2xhdmUiXQ~
Hardware-Attestation: v=1; typ=ENC; alg=ES256;
        h=from:to:subject:date:message-id:content-transfer-encoding:cont
        ent-type:mime-version:from:to:subject:date:message-id:content-tr
        ansfer-encoding:content-type:mime-version;
        bh=NoBLb0ghkIoYnL6DZMKhTd4PGygEyah4fcuBEbXYnrw; ts=1774527256;
        chain=MIIKdQYJKoZIhvcNAQcCoIIKZjCCCmICAQExDzANBglghkgBZQMEAgEFAD
```

ALBgkqhkiG9w0BBwGgggmJMIIDFDCCAfygAwIBAgIUYIji1mIf1yIg8m/k5lEK2i
MpM3cwDQYJKoZIhvcNAQELBQAwLjEMMAoGA1UECgwDMUlEMR4wHAYDVQQDDBUxSU
QgSW50ZXJtZWRpYXRlIENBIEEwHhcNMjYwMzI2MDUzNDU5WhcNMjcwMzI2MDUzND
U5WjBMMQwwCgYDVQQKDAMxSUQxJTAjBgNVBAMMHHVybjphaWQ6Y29tLjFpZDoxaW
QteGl6NDNteHoxFTATBgNVBAUTDDFpZC14aXo0M214ejBZMBMGByqGSM49AgEGCC
qGSM49AwEHA0IABHwvUSQJfLIavGu2Smbr5KsygOlMZP0I5aR5fAPjyTzrEuDKxs
QY547Ma1bD0fkBWrnVFCCSZlGE0JYUIQGp0fOjgdYwgdMwJwYDVR0RBCAwHoYcdX
JuOmFpZDpjb20uMWlkOjFpZC14aXo0M214ejAMBgNVHRMBAf8EAjAAMA4GA1UdDw
EB/wQEAwIHgDAdBgNVHQ4EFgQUwKjPuDH2bUWHa3MOaJHQUM+p+NMwHwYDVR0jBB
gwFoAUZyLAeeBVcjYns0Si80zfgF3flq4wFQYKKwYBBAGD1F8BAQQHZW5jbGF2ZT
AiBgorBgEEAYPUXwECBBQyMDI2LTAzLTI2VDA1OjM0OjU5WjAPBgorBgEEAYPUXw
EDBAEAMA0GCSqGSIb3DQEBCwUAA4IBAQCui/V804Jj9ZE3IJy7a5i7niFncCdMzQ
98dMM2nnYTZiq4nW14avCb1liuAGIlvxflodvv4V9h0kHq8C1YC/IIUigtpX5NWb
KfHOvDyCavQTZNm/YtEgyHH9fy22SOm+bCRjN4yy0xlTpzgT8ct97/qZBiTMB1fD
LKz5MZAR/LC/bAEFUT+RGipaQKyk+ImZyeq5yCzGGl+KiRbsY0nBn+vNezhmckin
iB1lx06W+R/WgwMsIX3k6BiJJsMhqelsI2Mx7WTfdRyA7BLxkUlAd8P0CAN4+xyM
a2BaCJLLgN41BSJ40GdZ3I6aUo6qBNtditaTco8RjJm6Dm9JPl7lj4MIIDSDCCAj
CgAwIBAgIUSEEYK28j6n6Vhr+w4PvDkiuyQbEwDQYJKoZIhvcNAQELBQAwJjEMMA
oGA1UECgwDMUlEMRYwFAYDVQQDDA0xSUQgUm9vdCBDQSBBMB4XDTI2MDMxNDE0MT
M0N1oXDTMxMDMxMzE0MTM0N1owLjEMMAoGA1UECgwDMUlEMR4wHAYDVQQDDBUxSU
QgSW50ZXJtZWRpYXRlIENBIEEwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAo
IBAQC554xsVDRUJ3fMEQISK3ayhz2DTFpHDMvhBP6MHqTt/116HDJNj+2qXJdAmV
H8GdXzMwm/tt9nrGrx4Sd3vcdodwpUnF2DhVASZ8e2TiYYB9a4DLuPc3lOw+lwVO
qCQmUDBSneQffINU+87WofNtWZQV28Oxy2HFDfFhU/WkI91ayiI/ihHQvQTJpqqZ
PAbYn+EjD+8AarBbHgKsyyDxW8oOmws82nYhn8eRdz7O3De2w9cpZouJMyWLxG9n
wUSOxD3fc1+vbcMeoRmBLRADx9RmhrHOiB2HFW579bsqdkzwQ10/SM5/SXmhSdwa
jxU/mnkGZCE/vcndtUzJ5B/twjAgMBAAGjZjBkMBIGA1UdEwEB/wQIMAYBAf8CAQ
AwDgYDVR0PAQH/BAQDAgEGMB0GA1UdDgQWBBRnIsB54FVyNiezRKLzTN+AXd+Wrj
AfBgNVHSMEGDAWgBTDt+lZdcVf+UYPLzmblKpM+SXw8DANBgkqhkiG9w0BAQsFAA
OCAQEAhUDFmVN+vZ4SK1CQPPH9XGzkJ+09iAhUkC3lLNK+hWMpBZ+S2Hs8KVl4aM
aZmD2+jOrC4Wj5Oe+rVsJ7yWiJmfQ9fCyMR43GAoCcAZyDsq9pyyfIfXbQO6+dPU
0TffL+9HgY2qme8XN/vsOqWY3gEvxihrW9inVc16EeDv1iRLcWUZe5o3uS7tZTdS
6/bkSb1lu81t9nK8YZ62jxNY2SB9acX9qx9MWNjBSsX7yTjHEAVn/Q6OMzdjS/Sa
l1/RcATxrzjo8bn22feqeikJCaoq4lOK/kKBZfz9F0BGNVH95AjChp25bliGYQXM
zPbiQuUnm6zqMdIYJg/NtPzg6kujCCAyEwggIJoAMCAQICFGciMXFDNUuITsmdET
Po+lhP0nQPMA0GCSqGSIb3DQEBCwUAMCYxDDAKBgNVBAoMAzFJRDEWMBQGA1UEAw
wNMUlEIFJvb3QgQ0EgQTAgFw0yNjAzMTQxNDEzNDdaGA8yMDU2MDMwNjE0MTM0N1
owJjEMMAoGA1UECgwDMUlEMRYwFAYDVQQDDA0xSUQgUm9vdCBDQSBBMIIBIjANBg
kqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4CSklrJ5JJwqpHQkSbH6LoQZp7kXqu
hEmyoZBED9/BkyzDsld4o3IyFdxvafrDu4e5YDzGIxjHgCvCvwUgbEZt6femOqSM
91VPKMuBzMEq0rtHCz7J/vS/nsfw3/NqgFN+j+q1HyHHVx+L3W8wQPWrVLqI2RUF
PyU9oeZL6XKaaL0PNIwL2NxAKMuc5uQ/BUPNjDsh7I4M5gD6X9WR/4ZEnvUd0sli
l8fJPLRjw6om2eq/kFl5pXW7mipdpf2ufiYEIc1lEBtf34krCWM3PEvnl0Rs4eEir
Dirxx7PFP18GOgoNyaQ8NNZZSOqC6LMrWdoXO6v8MKQVcCcInRsTfdPQIDAQABo0
UwQzASBgNVHRMBAf8ECDAGAQH/AgEBMA4GA1UdDwEB/wQEAwIBBjAdBgNVHQ4EFg
QUw7fpWXXFX/lGDy85m5SqTPkl8PAwDQYJKoZIhvcNAQELBQADggEBAAhRXzslG9
2koaKTfGR2KPZvOP3sxZvHBemfwb4tICwaegZlNtHztqP7qcLqWfsHG0Caob8e2r
bjMjpPaMY81328kgl7goVPTEuq3Af5joZOArHfizgQl9r5RwT9G3y/AgDTnZDF6u

```
        UKt5P1u8PUd8t6dUANvIjMvqD6eV8ERioMu+BL7GpbDvy7smvmsCpy1isY0aSlh1
        12LGboJueGSbiMx35wMKr6ofXy0EE+Iz2JxPX1+rPzxZ9YI5iYcUEVptBZW0Ik83
        sLXJdVG6Z745awnOAQnkriRwt2b0r/JRbmu7TQrzAv/K0PTXTuYf3DYxqCl04w8c
        pgEaPd5rWG4yYxgbEwga4CAQEwRjAuMQwwCgYDVQQKDAMxSUQxHjAcBgNVBAMMFT
        FJRCBJbnRlcm1lZGlhdGUgQ0EgQQIUYIji1mIf1yIg8m/k5lEK2iMpM3cwDQYJYI
        ZIAWUDBAIBBQAwCgYIKoZIzj0EAwIERjBEAiAZJKFrLr5sP2x4MFEegb/ER4Hk+u
        BbCUUQvYsCzIvbhgIgDHJ40PwTkZ4aVPvg2gBxnh/F5j62qlcvA6OgA1H67MU=;
        aid=urn:aid:com.1id:1id-xiz43mxz
```

RFC example 3 of 6: hardware email attestation per draft-drake-email-hardware-attestation-00.

Trust tier:	enclave (Apple Secure Enclave)
Identity:	urn:aid:com.1id:1id-xiz43mxz
Device:	Apple Mac mini (M2, SE P256)
SDK:	Python (oneid-sdk) v1.3.4
Mode:	Combined (Mode 1 + Mode 2)
Headers:	Hardware-Attestation (Mode 1)
	Hardware-Trust-Proof (Mode 2)

What to look for:

- trust_tier=enclave in the SD-JWT.
- The enclave tier shares a compatibility group with virtual (vTPM): both provide key isolation but lack anti-Sybil guarantees (re-keyable at will).
- Hardware-Attestation carries a CMS SignedData bundle with the SE-backed ECDSA P-256 key and certificate chain.
- A different identity than examples 1/6 (this device has its own enrollment).

Figure 3: Example 3: Enclave Secure Enclave, Combined Mode (Python SDK)

C.4. Example 4: Virtual-Tier vTPM (Combined Mode, Python SDK)

This message demonstrates Combined Mode from a virtual-tier identity backed by a VMware virtual TPM (vTPM). The Hardware-Attestation header uses typ=VRT (not typ=TPM), allowing verifiers to distinguish virtual from physical TPMs. The certificate chain is issuer-signed rather than rooted at a hardware manufacturer CA, since no physical manufacturer is involved. Receiving servers MAY apply different policy to virtual-tier attestations compared to sovereign or portable tiers.

```
Received: from w11vtpm.home (unknown [2001:8000:1bed:6d00:381f:a328:fe9d
        :860e] (AS1221 Telstra Limited, AU))(using TLSv1.3 with cipher
        TLS13_AES_256_GCM_SHA384) by mail.mailpal.com (Stalwart SMTP)
        with ESMTPSA id 423AB909844F0D1; Thu, 26 Mar 2026 06:28:43 +0000
Return-Path: <alice@example.un.ag>
DKIM-Signature: v=1; a=rsa-sha256; s=rsa; d=example.un.ag;
        c=relaxed/relaxed; r=y; h=Message-ID:Date:Subject:To:From;
        t=1774506523; x=1775370523;
        bh=t6uFMzcZhV+3fr99tSKnV1CYWnH47FK64RF9EcFXYnQ=; b=sSECW3y5QM+9R
        etAUX4BiZfCHAD7Vg0YjQvnGQym35d0GwpO9UpvH3kUI0s8ckck0Gfx/GDMnBqye
        cF22Ho/TmRtxYhjNXs1Hdspd97Gj9R89T4vbRTP4Se6AoeTCGY5sxEYkO/feYxau
        rVTEvSlyBQTOyPteCQBIvu+LTOnfseQBWaXmQEmgRF0hLhSnPpestgFtxbPvq7W7
        gpxet7GjEIs6nIrdjbeYDbY9izBkLFj6TcUuZbwMiH3KH01wm/qyx0KOoUP7g+5G
        6zsaGuvrmuEgfcX1DMVxAIlHhTJB1nAnHtUStvm4j8t1F9YEUJKD1YCSgLJQ8WvQ
        AJTTs/A0A==;
Authentication-Results: mailpal.com; hw-attest=pass header.typ=VRT
        header.alg=RS256 header.tier=virtual
        header.aid=urn:aid:com.1id:1id-jq8c84k4
Authentication-Results: mailpal.com; hw-trust=pass
        header.trust_tier=virtual header.registry=1id.com
From: Alice <alice@example.un.ag>
To: bob@example.un.ag
Subject: RFC Example 4/6: Virtual vTPM (Python, Mode 1+2)
Date: Thu, 26 Mar 2026 17:28:00 +1100
Message-ID: <177450648054.26936.15036271949777087817@mailpal.com>
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
Hardware-Trust-Proof: eyJhbGciOiJFUzI1NiIsImtpZCI6IjFpZC1od2F0dGVzdC1lcz
        I1Ni0xIiwidHlwIjoic2Qrand0In0.eyJpc3MiOiJodHRwczovLzFpZC5jb20iLC
        JpYXQiOjE3NzQ1MDY0OTUsImV4cCI6MTc3NDUwNjc5NSwibm9uY2UiOiJxTjVwLT
        VkcEd4ZUtxc3hqcEY3WTMxb2ZIeEN3VU1vaU9YYWV3S09OdjlnIiwiX3NkX2FsZy
        I6InNoYS0yNTYiLCJfc2QiOlsiQ3VPNVBZb0lWMVVrS0RIWmlXOVpNSFJ4Q3hFRG
        Y3UEYtZEU5WXJpVTlKbyJdfQ.rjfzv36TBpjz7fAazYElH06l2n_PpX1gGRUltXO
        NzP-GX5leTEFvUALdxaEHQwq8GmLHM7gpavPS_C_QEF3m2w~WyJmMW1CeWY2LXdI
        NWFkVDFoWmpDRDRBIiwidHJ1c3RfdGllciIsInZpcnR1YWwiXQ~
Hardware-Attestation: v=1; typ=VRT; alg=RS256; h=from:to:subject:date:me
        ssage-id:content-transfer-encoding:content-type:mime-version:fro
        m:to:subject:date:message-id:content-transfer-encoding:content-t
        ype:mime-version; bh=FB8IXgMtG8UBjtblS5jLcFilBP9NoufFQV8ba9fsRLs
        ; ts=1774506497; chain=MIIL/wYJKoZIhvcNAQcCoIIL8DCCC+wCAQExDzANB
        glghkgBZQMEAgEFADALBgkqhkiG9w0BBwGgggpUMIID3zCCAsegAwIBAgIUCwbxp
        fLn+aYaUdcXrA+ytHDfwU8wDQYJKoZIhvcNAQELBQAwLjEMMAoGA1UECgwDMUlEM
        R4wHAYDVQQDDBUxSUQgSW50ZXJtZWRpYXRlIENBIEEwHhcNMjYwMzI1MTYxOTEyW
        hcNMjcwMzI1MTYxOTEyWjBMMQwwCgYDVQQKDAMxSUQxJTAjBgNVBAMMHHVybjpha
        WQ6Y29tLjFpZDoxaWQtanE4Yzg0azQxFTATBgNVBAUTDDFpZC1qcThjODRrNDCCA
        SIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOa84tc3dh9fVDBWTUA4I+MLo
        IoWxncnfqfRZo25ODrQi8ndZTqTZXrj9mt2kVLgFtUNT7b5XjGHRPz/zHpBm8dNx
```

i/jISGOQ6diWM7xPikXvid/1hzCBmIOkrTSAajVkrz3UJb6U43CdVARX9NQh48Gd
jrc5dtk3zyvtmvaxfIagCenQArwS4Iv6N+8U+5OvURc713L2P2pd81u4u6lZgEep
bK0dUdLoYxq8fEAT/Ru4sHZtJFymQTMNVCcWRqVpZPRpezNktM6FfS2tcNzgGEhB
388Os7EIntDlesDePdv4EVzXDOoL43Xz2TSFA8MCWqhIsu1Kpxbmed7KBbJcC0CA
wEAAaOB1jCB0zAnBgNVHREEIDAehhx1cm46YWlkOmNvbS4xaWQ6MWlkLWpxOGM4N
Gs0MAwGA1UdEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgeAMB0GA1UdDgQWBBSdMIQyG
s98xq+qCV9L05jyTS6hWjAfBgNVHSMEGDAWgBRnIsB54FVyNiezRKLzTN+AXd+Wr
jAVBgorBgEEAYPUXwEBBAd2aXJ0dWFsMCIGCisGAQQBg9RfAQIEFDIwMjYtMDMtM
jBUMTQ6NDk6MTVaMA8GCisGAQQBg9RfAQMEAQAwDQYJKoZIhvcNAQELBQADggEBA
BfRe2qtKDxp4O+mlH+sgot0MX+UxDIVJCTQyB9i/Ew9lTYERt9ZDDpxG5ZJAvuf7
6FOoOtjQ69/BdSu9f3JEsbILzpLX8MHITT+rREAoUelnUMsLJz7OFieXhk6jMelW
YdbPpyrArufEr3/yLZa+f50AYM+5e6JUBO0KYWlL0zEP9WMGFtxhO+ZtxBD8NUat
NbAOL3gRAi3nz8OVnySzBNhunZPa89NRSHvHshF/63hroRVGnHvq/x1ANJFU9L0V
pj+URuWetRYd4k8c79KicDCGnH4m6Q4i1coumbeaT9fYLMpEjczAgtejCnPjuQoB
91rgREUxD+4O4OhvYpJ79YwggNIMIICMKADAgECAhRIQRgrbyPqfpWGv7Dg+8OSK
7JBsTANBgkqhkiG9w0BAQsFADAmMQwwCgYDVQQKDAMxSUQxFjAUBgNVBAMMDTFJR
CBSb290IENBIEEwHhcNMjYwMzE0MTQxMzQ3WhcNMzEwMzEzMTQxMzQ3WjAuMQwwC
gYDVQQKDAMxSUQxHjAcBgNVBAMMFTFJRCBJbnRlcm1lZGlhdGUgQ0EgQTCCASIwD
QYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALnnjGxUNFQnd8wRAhIrdrKHPYNMW
kcMy+EE/owepO3/XXocMk2P7apcl0CZUfwZ1fMzCb+232esavHhJ3e9x2h3ClScX
YOFUBJnx7ZOJhgH1rgMu49zeU7D6XBU6oJCZQMFKd5B98g1T7ztah821ZlBXbw7H
LYcUN8WFT9aQj3VrKIj+KEdC9BMmmqpk8Btif4SMP7wBqsFseAqzLIPFbyg6bCzz
adiGfx5F3Ps7cN7bD1ylmi4kzJYvEb2fBRI7EPd9zX69twx6hGYEtEAPH1GaGsc6
IHYcVbnv1uyp2TPBDXT9Izn9JeaFJ3BqPFT+aeQZkIT+9yd21TMnkH+3CMCAwEAA
aNmMGQwEgYDVR0TAQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OB
BYEFGciwHngVXI2J7NEovNM34Bd35auMB8GA1UdIwQYMBaAFMO36Vl1xV/5Rg8vO
ZuUqkz5JfDwMA0GCSqGSIb3DQEBCwUAA4IBAQCFQMWZU369nhIrUJA88f1cbOQn7
T2ICFSQLeUs0r6FYykFn5LYezwpWXhoxpmYPb6M6sLhaPk576tWwnvJaImZ9D18L
IxHjcYCgJwBnIOyr2nLJ8h9dtA7r509TRN98v70eBjaqZ7xc3++w6pZjeAS/GKGt
b2KdVzXoR4O/WJEtxZRl7mje5Lu1lN1Lr9uRJvWW7zW32crxhnraPE1jZIH1pxf2
rH0xY2MFKxfvJOMcQBWf9Do4zN2NL9JqXX9FwBPGvOOjxufbZ96p6KQkJqiriU4r
+QoFl/P0XQEY1Uf3kCMKGnbluWIZhBczM9uJC5SebrOox0hgmD820/ODqS6MIIDI
TCCAgmgAwIBAgIUZyIxcUM1S4hOyZ0RM+j6WE/SdA8wDQYJKoZIhvcNAQELBQAwJ
jEMMAoGA1UECgwDMUlEMRYwFAYDVQQDDA0xSUQgUm9vdCBDQSBBMCAXDTI2MDMxN
DE0MTM0N1oYDzIwNTYwMzA2MTQxMzQ3WjAmMQwwCgYDVQQKDAMxSUQxFjAUBgNVB
AMMDTFJRCBSb290IENBIEEwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBA
QDgJKSWsnkknCqkdCRJsfouhBmnuReq6ESbKhkEQP38GTLMOyV3ijcjIV3G9p+sO
7h7lgPMYjGMeAK8K/BSBsRm3p96Y6pIz3VU8oy4HMwSrSu0cLPsn+9L+ex/Df82q
AU36P6rUfIcdXH4vdbzBA9atUuojZFQU/JT2h5kvpcppovQ80jAvY3EAoy5zm5D8
FQ82MOyHsjgzmAPpf1ZH/hkSe9R3SyWKXx8k8tGPDqibZ6r+QWXmldbuaKl1/a5+
JgQhzWUQG1/fiSsJYzc8S+eXRGzh4SKsOKvHHs8U/XwY6Cg3JpDw01llI6oLosyt
Z2hc7q/wwpBVwJwidGxN909AgMBAAGjRTBDMBIGA1UdEwEB/wQIMAYBAf8CAQEwD
gYDVR0PAQH/BAQDAgEGMB0GA1UdDgQWBBTDt+lZdcVf+UYPLzmblKpM+SXw8DANB
gkqhkiG9w0BAQsFAAOCAQEACFFfOyUb3aShopN8ZHYo9m84/ezFm8cF6Z/Bvi0gL
Bp6BmU20fO2o/upwupZ+wcbQJqhvx7atuMyOk9oxjzXfbySCXuChU9MS6rcB/mOh
k4Csd+LOBCX2vlHBP0bfL8CANOdkMXq5Qq3k/W7w9R3y3p1QA28iMy+oPp5XwRGK
gy74EvsalsO/Luya+awKnLWKxjRpKWHXXYsZugm54ZJuIzHfnAwqvqh9fLQQT4jP
YnE9fX6s/PFn1gjmJhxQRWm0FlbQiTzewtcl1UbpnvjlrCc4BCeSuJHC3ZvSv8lF

```
        ua7tNCvMC/8rQ9NdO5h/cNjGoKXTjDxymARo93mtYbjJjGCAW8wggFrAgEBMEYwL
        jEMMAoGA1UECgwDMUlEMR4wHAYDVQQDDBUxSUQgSW50ZXJtZWRpYXRlIENBIEECF
        AsG8aXy5/mmGlHXF6wPsrRw38FPMA0GCWCGSAFlAwQCAQUAMAsGCSqGSIb3DQEBC
        wSCAQDG3jQoFX3z+sqGckMG62REgN0joeg3g660ySov4asnz2x/Hj1XU9CpmUNw3
        DI4GiiEFn3ZmATbSnSYlV1NDVPJ2Q91No4rIQxVJMYfeF0EIrQ2+Ur0ycEuTyZdh
        yGtDq8M2NbHVo2oeSRD4XLK/p//KOURU9tdDleNmL93N1Eo6GZpzoIX+unGH9SR6
        Kt294TE+WRzw8WFLKPTxcaiicMAv+1G37IwC+VzUxjqYBGMLESZCx9xnB0xeG/Y7
        NwIqqf7X5JWm6QZ2PGDCyjZWgEe0cUnGEEyj+YZS2LW/XqQghsC1LVFzIrTejhlZ
        363fAWD8wMLeAa5cJnYWjAL2ijM; aid=urn:aid:com.1id:1id-jq8c84k4
```

RFC example 4 of 6: hardware email attestation per draft-drake-email-hardware-attestation-00.

```
  Trust tier:  virtual (vTPM, typ=VRT)
  Identity:    urn:aid:com.1id:1id-jq8c84k4
  Device:      VMware virtual TPM (vTPM)
  SDK:         Python (oneid-sdk) v1.3.2
  Mode:        Combined (Mode 1 + Mode 2)
  Headers:     Hardware-Attestation (Mode 1)
               Hardware-Trust-Proof (Mode 2)
```

What to look for:

- trust_tier=virtual in the SD-JWT.
- The virtual tier sits between sovereign and declared: the TPM exists but is not physically tamper-resistant.
- Hardware-Attestation is present because the vTPM can still produce a valid AK certificate chain.
- A verifier can distinguish virtual from sovereign by inspecting the trust_tier claim in the SD-JWT.

Figure 4: Example 4: Virtual vTPM, Combined Mode (Python SDK)

C.5. Example 5: Declared-Tier Software-Only (Mode 2 Only, Python SDK)

This message demonstrates a declared-tier identity where the signing key is stored in software with no hardware protection. Only Mode 2 (Hardware-Trust-Proof) succeeded; Mode 1 failed because the EdDSA algorithm used by this identity's software key is not yet supported in the reference SDK's CMS implementation. The SD-JWT discloses trust_tier="declared", allowing receiving servers to apply differentiated policy (e.g., lower reputation weight). Declared-tier identities still participate in reputation tracking, but without hardware-backed Sybil resistance.

```
Received: from RoG.localdomain (unknown [2001:8000:1bed:6d00:ecfa:f9e5:c
        8e3:ed4f] (AS1221 Telstra Limited, AU))(using TLSv1.3 with
        cipher TLS13_AES_256_GCM_SHA384) by mail.mailpal.com (Stalwart
        SMTP) with ESMTPSA id 423ADB058650CD1; Thu, 26 Mar 2026 06:47:16
        +0000
Return-Path: <alice@example.un.ag>
DKIM-Signature: v=1; a=rsa-sha256; s=rsa; d=example.un.ag;
        c=relaxed/relaxed; r=y; h=Message-ID:Date:Subject:To:From;
        t=1774507636; x=1775371636;
        bh=tQbweOdhmIKnVJeZ2WfV+dXtsk7S4L1RTzicUBPYEuk=; b=6yNSGTOK1IodR
        J59Jbhd9F/mFwrvJeGPv1C3sKb4eZKbPmYBQtdDjWuo4VknvsiZOE3uowJV6HitJ
        xCwd3ORAby9tMH4CX+U8aaPrcu24pGxRGP3qHKuJcj3LtnktSj3/RBwipjX7roqu
        gwZwEwJMBN8I6ilXN7GiC9F03AOfuraBSAUw+4CMk6oHl6tK2jeEXKu0VAHg0tOT
        qX05HOl4VTEy6meCKlXDRvJJO+iakUBa0udVtHMthHQYkBsGnNoBllUgXSpgXdxg
        nuAx1oVMpuZUdjBJZdtC5xpeZsupW4myKjfEP4lUuh00o5j4MbCZqUWAr0xriMpz
        2rD7EWaAw==;
Authentication-Results: mailpal.com; hw-trust=pass
        header.trust_tier=declared header.registry=1id.com
From: Alice <alice@example.un.ag>
To: bob@example.un.ag
Subject: RFC Example 5/6: Declared tier (Python, Mode 2)
Date: Thu, 26 Mar 2026 16:47:11 +1000
Message-ID: <177450763154.21623.9499316015451560959@mailpal.com>
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
Hardware-Trust-Proof: eyJhbGciOiJFUzI1NiIsImtpZCI6IjFpZC1od2F0dGVzdC1lcz
        I1Ni0xIiwidHlwIjoic2Qrand0In0.eyJpc3MiOiJodHRwczovLzFpZC5jb20iLC
        JpYXQiOjE3NzQ1MDc2MzIsImV4cCI6MTc3NDUwNzkzMiwibm9uY2UiOiIxV040Q1
        RsRERubUVkdDM2NXFSZXl6dk1mX21VMVFjak82Y1dGMVRfNDg4IiwiX3NkX2FsZy
        I6InNoYS0yNTYiLCJfc2QiOlsieGt6MXhiTnhJbmZMWXdObzZ6dWNtTjVhbVB0VH
        FCaDFWOE4xYVQ4cy1BOCJdfQ.DxUzkzn3Bh33Nt-Tpq7YLkKYkYutLJyNgj3jJYG
        KuKNv-MoMuFXpnhlpVnG5qasrIb_TRJDmAGifvnOBzV5ubA~WyI4VGUwNHd4cHpL
        YU9POURQaVhfZXBnIiwidHJ1c3RfdGllciIsImRlY2xhcmVkIl0~

RFC example 5 of 6: hardware email attestation per
draft-drake-email-hardware-attestation-00.

  Trust tier:  declared (software key, typ=SFT)
  Identity:    urn:aid:com.1id:1id-zrw4psyg
  Device:      Software-only (Ed25519 key)
  SDK:         Python (oneid-sdk) v1.3.2
  Mode:        Mode 2 only (SD-JWT trust proof)
  Header:      Hardware-Trust-Proof only

What to look for:
  - trust_tier=declared in the SD-JWT.
  - No Hardware-Attestation header (Mode 1
```

```
    requires a certificate chain, which
    declared-tier identities lack).
  - The declared tier is the lowest trust
    level: the identity has no hardware
    binding. It is equivalent to a standard
    software signing key.
  - This tier exists so that agents without
    hardware can still participate in the
    attestation ecosystem (e.g. to begin
    accruing reputation), with recipients
    aware of the reduced assurance.
```

Figure 5: Example 5: Declared Software-Only, Mode 2 (Python SDK)

C.6. Example 6: Sovereign-Tier TPM (Mode 1 Only, Python SDK)

This message demonstrates Mode 1 only (Direct Hardware Attestation) from the same physical TPM and agent identity as Example 1, but using the Python SDK instead of Node.js. There is no Hardware-Trust-Proof header; only the Hardware-Attestation header is present. This is the simpler deployment model where no SD-JWT issuer/registry interaction is needed -- the CMS signature is independently verifiable against the TPM manufacturer's root CA with no trust in any identity service required. The Authentication-Results header shows only hw-attest (no hw-trust result).

```
Received: from [172.22.1.89] (unknown [2001:8000:1bed:6d00:ecfa:f9e5:c8e
        3:ed4f] (AS1221 Telstra Limited, AU))(using TLSv1.3 with cipher
        TLS13_AES_256_GCM_SHA384) by mail.mailpal.com (Stalwart SMTP)
        with ESMTPSA id 423ADE6B40516D1; Thu, 26 Mar 2026 06:49:08 +0000
Return-Path: <alice@example.un.ag>
DKIM-Signature: v=1; a=rsa-sha256; s=rsa; d=example.un.ag;
        c=relaxed/relaxed; r=y; h=Message-ID:Date:Subject:To:From;
        t=1774507748; x=1775371748; bh=629O0DX4KvelluIXRhMhqaQlrmx71QOQV
        Vfnw8aAYjc=; b=UYsTBy57SwrnwyQ++nVr0toCnbv3kdk8/1o6XDPY47otdI4Ix
        xupx4VaATFeIv+h+CorYOGAwB2WnDDZzUYYupCidt0aoJHfz238KhoEWs1L+fXhr
        tDZYrk3e4MyQAWLNlAAS11u9ViVeCgaSLro80DQUQ49yaixGwj68Hc5HkXpuHS/r
        uC1pGWa7Cjeu54s+sQklUHqPIwymhuRIdfKAcZJoMEx0GSRHb3QWLO0hISvHnUk0
        tnveNw3HN/riJ8PEC35itJiJTbSDLzRL4rOdaXOkjsnStwoNQE0gJ+cLawJtmn8Q
        a7LNz9gHaIKfSoPuTe/9vqClr6KXHRB8XRBjA==;
Authentication-Results: mailpal.com; hw-attest=pass header.typ=TPM
        header.alg=RS256 header.tier=sovereign
        header.aid=urn:aid:com.1id:1id-tkoie2ve
From: Alice <alice@example.un.ag>
To: bob@example.un.ag
Subject: RFC Example 6/6: Sovereign TPM (Python, Mode 1)
Date: Thu, 26 Mar 2026 16:49:05 +1000
Message-ID: <177450774520.30168.11382866965426144830@mailpal.com>
```

```
Content-Type: text/plain; charset="utf-8"
Content-Transfer-Encoding: 7bit
MIME-Version: 1.0
Hardware-Attestation: v=1; typ=TPM; alg=RS256; h=from:to:subject:date:me
        ssage-id:content-transfer-encoding:content-type:mime-version;
        bh=uQAodZKMniNXQzM-9eg-efen0Sg2a7iaZwO10AhYOEM; ts=1774507745;
        chain=MIIMAQYJKoZIhvcNAQcCoIIL8jCCC+4CAQExDzANBglghkgBZQMEAgEFAD
        ALBgkqhkiG9w0BBwGgggpWMIID4TCCAsmgAwIBAgIUH01pfnWQCIfMiHfTNQb7v7
        r3CmMwDQYJKoZIhvcNAQELBQAwLjEMMAoGA1UECgwDMUlEMR4wHAYDVQQDDBUxSU
        QgSW50ZXJtZWRpYXRlIENBIEEwHhcNMjYwMzIwMTYzNjI2WhcNMjcwMzIwMTYzNj
        I2WjBMMQwwCgYDVQQKDAMxSUQxJTAjBgNVBAMMHHVybjphaWQ6Y29tLjFpZDoxaW
        QtdGtvaWUydmUxFTATBgNVBAUTDDFpZC10a29pZTJ2ZTCCASIwDQYJKoZIhvcNAQ
        EBBQADggEPADCCAQoCggEBANvysXC8ILSnhGcqg66dWKWQLp9pOJZmj6yTbICDVL
        ydBwc1LGEQAW3BCee6XU/Bs0n91uROJCQw3jgrCMiBUOvS6Mjox/8NT1gfZ9sDQG
        VXN5gQ0t9H+r9vEPJ7YL9jhYmlxWQp+fS6vhZNrmH7g4XwEK+EsbMsc8H0ZuxKAM
        1S7p9447G9IOZOkFMK33ZxCCY00DFmwzXu/dkGFfuOaSWM5cazWz/kxqOtcpT73A
        CmJxQ6LU3yzB6VwbfwCWm6klQNKsd1c7FT5hX5ZYtEqDXeIpIGWyQp9TQgNbUIPe
        JCWO2rDLmh0ZXwqXR4yBqKeNWhUjrcnfQgNfrjJ7zB0z8CAwEAAaOB2DCB1TAnBg
        NVHREEIDAehhx1cm46YWlkOmNvbS4xaWQ6MWlkLXRrb2llMnZlMAwGA1UdEwEB/w
        QCMAAwDgYDVR0PAQH/BAQDAgeAMB0GA1UdDgQWBBRroL9zk1MDMjMG4UyIvjZdzd
        X8QTAfBgNVHSMEGDAWgBRnIsB54FVyNiezRKLzTN+AXd+WrjAXBgorBgEEAYPUXw
        EBBAlzb3ZlcmVpZ24wIgYKKwYBBAGD1F8BAgQUMjAyNi0wMy0wNFQwMDowMDowMF
        owDwYKKwYBBAGD1F8BAwQBADANBgkqhkiG9w0BAQsFAAOCAQEAiL2Fwbib2C2bzY
        zRpDsyFGSXmn2LcFO3snxAaoRJxwMqa5UcyNZNaPk9SHtr7mocHO8spAi5q+7Twa
        8+1djZ3HJha5iKW59E9o2/JFBBM8p1ehkHVmlTtF130RFjaaW1BnPvdfjSMCdYHk
        9IIOw90+oAA7omUAlYTXclGLduE2fOfVcAaqIz+wFYS89iJOtlbj+BH8wwa6Ee3P
        1pEe1p4LW4KGdaIrA1/yfarCaglkVlJxA4GsR2y1H43l7ZNijh/lsS+iRJWXvBWG
        qecVx+GY/jPpvG4ahObRO4/MW5F1i3fKvkE0UX4juC5t0nPtCbRsQFRmbIcQv+Oc
        LJV9MexzCCA0gwggIwoAMCAQICFEhBGCtvI+p+lYa/sOD7w5IrskGxMA0GCSqGSI
        b3DQEBCwUAMCYxDDAKBgNVBAoMAzFJRDEWMBQGA1UEAwwNMUlEIFJvb3QgQ0EgQT
        AeFw0yNjAzMTQxNDEzNDdaFw0zMTAzMTMxNDEzNDdaMC4xDDAKBgNVBAoMAzFJRD
        EeMBwGA1UEAwwVMUlEIEludGVybWVkaWF0ZSBDQSBBMIIBIjANBgkqhkiG9w0BAQ
        EFAAOCAQ8AMIIBCgKCAQEAueeMbFQ0VCd3zBECEit2soc9g0xaRwzL4QT+jB6k7f
        9dehwyTY/tqlyXQJlR/BnV8zMJv7bfZ6xq8eEnd73HaHcKVJxdg4VQEmfHtk4mGA
        fWuAy7j3N5TsPpcFTqgkJlAwUp3kH3yDVPvO1qHzbVmUFdvDscthxQ3xYVP1pCPd
        WsoiP4oR0L0EyaaqmTwG2J/hIw/vAGqwWx4CrMsg8VvKDpsLPNp2IZ/HkXc+ztw3
        tsPXKWaLiTMli8RvZ8FEjsQ933Nfr23DHqEZgS0QA8fUZoaxzogdhxVue/W7KnZM
        8ENdP0jOf0l5oUncGo8VP5p5BmQhP73J3bVMyeQf7cIwIDAQABo2YwZDASBgNVHR
        MBAf8ECDAGAQH/AgEAMA4GA1UdDwEB/wQEAwIBBjAdBgNVHQ4EFgQUZyLAeeBVcj
        Yns0Si80zfgF3flq4wHwYDVR0jBBgwFoAUw7fpWXXFX/lGDy85m5SqTPkl8PAwDQ
        YJKoZIhvcNAQELBQADggEBAIVAxZlTfr2eEitQkDzx/Vxs5CftPYgIVJAt5SzSvo
        VjKQWfkth7PClZeGjGmZg9vozqwuFo+Tnvq1bCe8loiZn0PXwsjEeNxgKAnAGcg7
        KvacsnyH120DuvnT1NE33y/vR4GNqpnvFzf77DqlmN4BL8Yoa1vYp1XNehHg79Yk
        S3FlGXuaN7ku7WU3Uuv25Em9ZbvNbfZyvGGeto8TWNkgfWnF/asfTFjYwUrF+8k4
        xxAFZ/0OjjM3Y0v0mpdf0XAE8a846PG59tn3qnopCQmqKuJTiv5CgWX8/RdARjVR
        /eQIwoaduW5YhmEFzMz24kLlJ5us6jHSGCYPzbT84OpLowggMhMIICCaADAgECAh
        RnIjFxQzVLiE7JnREz6PpYT9J0DzANBgkqhkiG9w0BAQsFADAmMQwwCgYDVQQKDA
        MxSUQxFjAUBgNVBAMMDTFJRCBSb290IENBIEEwIBcNMjYwMzE0MTQxMzQ3WhgPMj
```

```
        A1NjAzMDYxNDEzNDdaMCYxDDAKBgNVBAoMAzFJRDEWMBQGA1UEAwwNMUlEIFJvb3
        QgQ0EgQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOAkpJayeSScKq
        R0JEmx+i6EGae5F6roRJsqGQRA/fwZMsw7JXeKNyMhXcb2n6w7uHuWA8xiMYx4Ar
        wr8FIGxGben3pjqkjPdVTyjLgczBKtK7Rws+yf70v57H8N/zaoBTfo/qtR8hx1cf
        i91vMED1q1S6iNkVBT8lPaHmS+lymmi9DzSMC9jcQCjLnObkPwVDzYw7IeyODOYA
        +l/Vkf+GRJ71HdLJYpfHyTy0Y8OqJtnqv5BZeaV1u5oqXX9rn4mBCHNZRAbX9+JK
        wljNzxL55dEbOHhIqw4q8cezxT9fBjoKDcmkPDTWWUjqguizK1naFzur/DCkFXAn
        CJ0bE33T0CAwEAAaNFMEMwEgYDVR0TAQH/BAgwBgEB/wIBATAOBgNVHQ8BAf8EBA
        MCAQYwHQYDVR0OBBYEFMO36Vl1xV/5Rg8vOZuUqkz5JfDwMA0GCSqGSIb3DQEBCw
        UAA4IBAQAIUV87JRvdpKGik3xkdij2bzj97MWbxwXpn8G+LSAsGnoGZTbR87aj+6
        nC6ln7BxtAmqG/Htq24zI6T2jGPNd9vJIJe4KFT0xLqtwH+Y6GTgKx34s4EJfa+U
        cE/Rt8vwIA052QxerlCreT9bvD1HfLenVADbyIzL6g+nlfBEYqDLvgS+xqWw78u7
        Jr5rAqctYrGNGkpYdddixm6Cbnhkm4jMd+cDCq+qH18tBBPiM9icT19fqz88WfWC
        OYmHFBFabQWVtCJPN7C1yXVRume+OWsJzgEJ5K4kcLdm9K/yUW5ru00K8wL/ytD0
        107mH9w2MagpdOMPHKYBGj3ea1huMmMYIBbzCCAWsCAQEwRjAuMQwwCgYDVQQKDA
        MxSUQxHjAcBgNVBAMMFTFJRCBJbnRlcm1lZGlhdGUgQ0EgQQIUH01pfnWQCIfMiH
        fTNQb7v7r3CmMwDQYJYIZIAWUDBAIBBQAwCwYJKoZIhvcNAQELBIIBANlEWSrD7C
        9a1OdJkDysB9UvvSrYkdArqu4BjHu0nJxWJdHURKtXW73SYQK6eBYhGm+9FOgCdw
        syV+JsYijmeBMyB3iAwtYWWbEbjAnb0uj0WH2vhNEzKR8KYpus/YYauACqBPDCea
        BbFE7iFmhW1p9Zj6lSZ+M3EkuUTrZMuBadNxzWwOOdVls2+r8XSos8TgSRcnMsg4
        FdIs2KSL3pUAAMLLJEwUpBZ5VAIMlmFgUu/KSXfj97hnyl7qycq7TofKjLsLukqv
        U+58Oqs2eA8SxRnWQbvBs41JwywSZmAZ+2+gokICuWjteUnoUAkb1mPEy8ylFsK9
        jwbTSSBm9b2eU=; aid=urn:aid:com.1id:1id-tkoie2ve
```

RFC example 6 of 6: hardware email attestation per draft-drake-email-hardware-attestation-00.

Trust tier:	sovereign (hardware TPM, typ=TPM)
Identity:	urn:aid:com.1id:1id-tkoie2ve
Device:	Intel firmware TPM (PTT)
SDK:	Python (oneid-sdk) v1.3.2
Mode:	Mode 1 only (direct attestation)
Header:	Hardware-Attestation only

What to look for:

- Only Hardware-Attestation is present; no Hardware-Trust-Proof header.
- Same identity and device as example 1, but sent via the Python SDK.
- Demonstrates that Mode 1 and Mode 2 are independent and composable.
- The aid URN matches example 1.

Figure 6: Example 6: Sovereign TPM, Mode 1 Only (Python SDK)

C.7. Verification Failure Examples

The following Authentication-Results headers illustrate what receiving mail servers emit when verification fails. These are from actual test messages where the attestation was malformed or tampered with. Implementers building verifiers should emit similar diagnostic information in the parenthetical comment.

Mode 2 (SD-JWT Trust Proof) failure -- message-binding nonce mismatch, indicating the SD-JWT was not bound to this specific message (possible replay or header modification):

```
Authentication-Results: mailpal.com; hw-trust=fail
  header.trust_tier=declared header.registry=1id.com
  (Message-binding nonce mismatch: got
  'uVgDzsXUJUPch1I-hRM3vGgip4uIvrkE1aeixhsp3OU', expected
  'lB_mxWwfU9o301aGO4Yb3GdTzKm)
```

Mode 1 (Direct Hardware Attestation) failure -- cryptographic signature does not match, indicating the message body or signed headers were modified after signing (or the signing key does not match the certificate chain):

```
Authentication-Results: mailpal.com; hw-attest=fail
  header.typ=TPM header.alg=RS256 header.tier=sovereign
  header.aid=urn:aid:com.1id:1id-tkoie2ve
  (Signature verification failed: Cryptographic signature
  does not match)
```

Appendix D. Appendix: Applicability to Other Transports

The CMS attestation bundle (the "chain" value defined in Section 5.2) and the SD-JWT trust proof (defined in Section 6.2) are self-contained data structures whose verification algorithms (Section 5.3 and Section 6.4) do not depend on email semantics. The body hash (bh) generalises to a content hash over whatever payload the attestation covers.

Potential transport bindings include but are not limited to:

HTTP The CMS attestation bundle or SD-JWT trust proof could be carried in HTTP request headers or in HTTP message bodies. The content hash would cover the HTTP request or response payload. This binding is particularly relevant for API calls by autonomous AI agents, where the receiving service benefits from verifying that the request originated on attested hardware. The agent identity URN (Section 4.1) provides a persistent key for cross-protocol reputation tracking.

Agent-to-Agent Protocols Emerging protocols for AI agent communication -- tool invocation, task delegation, capability discovery -- could carry attestation evidence alongside each request, enabling agents to verify each other's hardware trust level before exchanging sensitive data or delegating privileged operations.

WebSocket and Streaming Protocols Attestation evidence could be presented during connection establishment (e.g., in the WebSocket upgrade request) to establish hardware trust for the duration of a persistent connection.

Detailed specification of these bindings is out of scope for this document and is deferred to future companion documents.

Appendix E. Appendix: URN Namespace Registration for "aid"

This appendix provides the registration template for the "aid" (Agent Identity) URN namespace, per [RFC8141] Section 5.1.

Namespace Identifier aid

Version 1

Date 2026-02-28

Registrant TBD -- to be transferred to the governance authority for the Agent Identity ecosystem upon its constitution. Interim contact: Christopher Drake <cnd@1id.com>.

Purpose The "aid" URN namespace provides persistent, globally unique identifiers for autonomous agents, designed specifically to enable reputation systems and Sybil-resistant authentication across Internet protocols.

As AI agents become ubiquitous across email, APIs, and Internet services, platforms need stable identifiers that: (1) distinguish legitimate agents from malicious ones, (2) accumulate reputation over time, (3) are anchored to unique physical hardware where available, and (4) support federated issuance by multiple independent identity providers.

URNs in this namespace are bound to hardware security components (TPM chips, PIV tokens, virtual TPMs) or to software-managed keys at a lower trust tier. The namespace supports multiple issuers, each identified by a reverse-DNS issuer-namespace within the URN structure.

Syntax See Section 4.1 for the formal ABNF. The general form is:

urn:aid:{namespace}:{agent-id}

where {namespace} is a single label (global) or a reverse-DNS domain (e.g., com.1id), and {agent-id} is either a canonical identifier (e.g., id-9854187104) or a handle (e.g., crusty).

Assignment Agent identifiers are assigned by Issuers -- organisations that operate enrollment services, verify hardware attestation, and maintain anti-Sybil registries. Any organisation owning a DNS domain can operate as an Issuer. Issuer-namespace uniqueness is guaranteed by DNS ownership. Agent-id uniqueness within a namespace is guaranteed by the Issuer's assignment policy. An Issuer MUST NOT reassign an agent-id, even after revocation.

Security and Privacy See the Security Considerations (Section 14) of this document. Agent identity URNs are persistent and globally unique by design, enabling long-term reputation tracking. For privacy-sensitive contexts, SD-JWT selective disclosure (Section 6) allows agents to prove trust properties without revealing their URN.

Interoperability URNs in this namespace appear in: JWT sub claims (OpenID Connect tokens), email headers (Hardware-Attestation, Hardware-Trust-Proof), HTTP Authorization headers, and reputation service APIs. Standard URN parsers ([RFC8141]) and standard JWT libraries process these identifiers without modification.

Resolution Resolution is OPTIONAL. Issuers MAY provide resolution services returning public identity metadata. The resolution URL is derived by reversing the issuer-namespace to obtain the Issuer domain and appending a well-known path. Resolution, when available, returns trust tier, registration timestamp, and optional handle -- but never authentication history or private data.

Documentation This document (draft-drake-email-hardware-attestation-00).

Revision Information Initial registration.

Appendix F. Acknowledgements

The concept of using hardware attestation for email sender verification was developed in the context of building a hardware identity registrar for autonomous AI agents. The author thanks the Trusted Computing Group for the TPM 2.0 specification, the authors of [RFC9901] (SD-JWT) for the selective disclosure mechanism, the authors of [RFC9334] (RATS Architecture) for establishing the remote attestation architecture that this document builds upon, and the authors of [RFC8141] (URN) for the persistent identifier framework that underpins the federated agent identity model.

Author’s Address

Christopher Drake
1id.com
Australia
Email: cnd@1id.com
URI: https://1id.com