Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 23 August 2026

C. Drake
1id.com

19 February 2026

Hardware Attestation for Email Sender Verification
draft-drake-email-tpm-attestation-00

Abstract

This document defines a mechanism for email senders to include hardware attestation evidence in message headers, enabling receiving mail servers to cryptographically verify that an email was composed on a machine containing a genuine Trusted Platform Module (TPM) from a known manufacturer (Intel, AMD, Infineon, or similar). The verification chain runs from the email header directly to the TPM manufacturer's root Certificate Authority, requiring no trust in any intermediary identity service.

As a companion mechanism, this document also defines a privacy-preserving alternative using SD-JWT (Selective Disclosure JWT, RFC 9901) where the sender can prove specific claims about their hardware trust level without revealing their hardware identity.

Together, these mechanisms provide Sybil-resistant email authentication: each sender requires a unique physical security chip, making large-scale automated spam economically infeasible regardless of advances in artificial intelligence.

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, and other Internet protocols.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

This Internet-Draft will expire on 23 August 2026.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Requirements Language 4
  1.2. Terminology 4
2. Problem Statement 5
3. Applicability 6
4. Mode 1: Direct TPM Attestation 7
  4.1. Attestation Certificate Chain 7
  4.2. Header Field Definition 7
  4.3. Verification Algorithm 9
  4.4. Authentication-Results Integration 10
  4.5. Privacy Considerations for Direct Attestation 11
5. Mode 2: SD-JWT Trust Proof 11
  5.1. Overview 11
  5.2. Header Field Definition 11
  5.3. Verification Algorithm 12
  5.4. Authentication-Results Integration 13
6. Combined Mode 14
7. Sybil Resistance Properties 14
  7.1. Direct TPM Attestation (Mode 1) 14
  7.2. SD-JWT Trust Proof (Mode 2) 15
  7.3. Virtual TPMs 15
8. Interaction with Existing Email Authentication 15
  8.1. Multiple Headers and Forwarding 16
9. Implementation Status 16
  9.1. 1id.com Trust Registry 17

10. IANA Considerations 17
10.1. Header Field Registration 18
10.2. Authentication Method Registration 18
11. Security Considerations 18
11.1. Manufacturer CA Compromise 18
11.2. Replay Attacks 18
11.3. Compromised Endpoints and Privilege Requirements 19
11.4. Physical Attacks on TPMs 19
11.5. Virtual TPM Risk 20
11.6. Policy Abuse and Deanonymisation Risk 20
11.7. Header Stripping and Modification 20
12. Normative References 21
13. Informative References 21
Appendix A. Appendix: Economics of Hardware-Based Sybil Resistance 23
Appendix B. Appendix: TPM Manufacturer Root CAs 23
Appendix C. Appendix: Applicability to Other Transports 24
Appendix D. Acknowledgements 25
Author's Address 25

1. Introduction

Email authentication today relies on three complementary mechanisms: SPF [RFC7208] verifies the sending IP address, 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 physical infrastructure that composed or sent 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 physics, applying the remote attestation architecture of [RFC9334] to email: specifically, to the Trusted Platform Module (TPM) chip present in virtually every modern PC, server, and many embedded devices.

A TPM contains a unique Endorsement Key (EK) burned in at the factory by the chip manufacturer, with a certificate chaining to the manufacturer's root CA. This key cannot be extracted, cloned, or transferred. By signing email content with a TPM-resident key and including the certificate chain in the message headers, a 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 - the same CAs they already trust for Secure Boot, platform integrity, and other TPM-dependent operations. No new trust relationships are required.

This document defines two complementary mechanisms:

1. *Direct TPM Attestation* (Section 4): A CMS [RFC5652] signed-data structure containing the attestation signature and full certificate chain, carried in an email header. Verifiable directly against manufacturer root CAs. Optimised for deliverability.

2. *SD-JWT Trust Proof* (Section 5): A Selective Disclosure JWT [RFC9901] issued by a trust registry, where the sender selects which claims to reveal. Optimised for privacy in contexts where revealing hardware identity is undesirable.

While this document defines these mechanisms for email, the attestation formats themselves are transport-independent: the CMS attestation bundle and SD-JWT trust proof are self-contained data structures that can be carried in any protocol capable of transporting octet strings. Appendix C discusses applicability to HTTP and other protocols.

1.1. 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.

1.2. Terminology

TPM Trusted Platform Module, as specified by the Trusted Computing Group (TCG) [TCG-TPM2]. This document assumes TPM 2.0.

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.

AK Attestation Key. A signing key generated inside the TPM after deployment, certified by the EK (via TPM2_Certify) to prove it resides in the same TPM. Used for signing operations in this protocol.

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

Trust Registry An entity that verifies TPM attestation evidence at enrollment time and subsequently issues trust assertions (e.g., SD-JWTs) about enrolled entities. A Trust Registry is OPTIONAL for Mode 1 (Direct TPM Attestation) and REQUIRED for Mode 2 (SD-JWT Trust Proof).

HSM Hardware Security Module. A physical device that safeguards cryptographic keys and provides cryptographic processing. TPMs are one category of HSM; portable smart cards (such as PIV-capable YubiKeys) are another. This document focuses on TPM-based attestation; companion documents may address other HSM types.

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 TPM chip. TPM chips 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 machines, each with a genuine TPM - a cost of approximately $500-2000 per identity (machine + TPM) compared to approximately $0 per identity with software-only approaches.

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

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 (EK public key hash) uniquely identifies the sending machine 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 a Trust Registry, which is not a typical consumer workflow.

Operators of human-facing email services MAY offer TPM 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 TPM 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 defined in this document - both the CMS attestation bundle and the SD-JWT trust proof - are self-contained data structures not inherently specific to email. They can be carried in HTTP headers, HTTP request bodies, WebSocket messages, or

any protocol supporting opaque octet strings. Appendix C discusses applicability to other transports. This document specifies the email transport binding as the primary application.

4. Mode 1: Direct TPM Attestation

This section defines a CMS-based attestation bundle containing a hardware-anchored signature and its full certificate chain. While presented here in the context of email headers (Section 4.2), the CMS SignedData structure is a self-contained, transport-independent data object; see Appendix C for applicability to other protocols.

4.1. Attestation Certificate Chain

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

1. *AK Certificate:* Created by TPM2_Certify, binding the AK public key to the EK-identified TPM. Contains the AK public key and the EK's certification signature.

2. *EK Certificate:* Issued by the TPM manufacturer at fabrication time. Contains the EK public key, TPM manufacturer identity, and TPM model 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 EK 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 TPM manufacturer root CAs are published at well-known URLs (e.g., Intel's Trusted Services API, AMD's Key Distribution Server).

4.2. Header Field Definition

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

```
TPM-Attestation = "TPM-Attestation" ":" SP tpm-attest-value CRLF

tpm-attest-value = tpm-version ";" SP
                   tpm-algorithm ";" SP
                   tpm-body-hash ";" SP
                   tpm-timestamp ";" SP
                   tpm-chain

tpm-version    = "v" "=" "1"
tpm-algorithm  = "alg" "=" ("RS256" / "ES256" / "PS256")
tpm-body-hash  = "bh" "=" base64url
tpm-timestamp  = "ts" "=" 1*DIGIT
tpm-chain      = "chain" "=" base64
```

Where:

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

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 RECOMMENDED for maximum compatibility with existing TPM deployments.

bh The base64url-encoded SHA-256 hash of the canonicalised email body, computed using the same canonicalisation algorithm as DKIM simple body canonicalisation Section 3.4.3 of [RFC6376].

ts Unix timestamp (seconds since 1970-01-01T00:00:00Z) at which the attestation signature was created.

chain A base64-encoded CMS SignedData structure [RFC5652] containing:

* The AK signature over the concatenation of the body hash and timestamp: Sign(AK, SHA-256(bh || ts))

* The AK certificate

* The EK certificate

* Any intermediate CA certificates

The CMS structure uses the SignedData content type (OID 1.2.840.113549.1.7.2). The encapContentInfo MUST be absent (detached signature); the signed content is the body hash concatenated with the timestamp, provided in the bh and ts fields.

4.3. Verification Algorithm

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

1. Parse the TPM-Attestation header field and extract v, alg, bh, ts, and chain. If parsing fails, the result is "none" (header malformed).

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

3. Compute the SHA-256 hash of the canonicalised email body using DKIM simple body canonicalisation. Compare with bh. If they differ, the result is "fail" (body modified).

4. 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. The ts value reflects when the sender created the attestation, not when the MTA transmitted the message; multi-hop delivery and temporary queue backlogs are common operational realities. If ts is outside the acceptable window, the result is "fail" (timestamp expired).

5. Decode the base64 chain value and parse as a CMS SignedData structure. Extract the signer certificate (AK certificate) and the certificate chain.

6. Validate the certificate chain:

    a. The AK certificate MUST contain a TPM2_Certify structure in an extension or as the subject, binding the AK public key to the EK.

    b. The EK 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 TPM manufacturer CA trust store.

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

7. Verify the CMS signature using the AK public key and the specified algorithm. The signed content is SHA-256(bh || ts). If signature verification fails, the result is "fail" (bad signature).

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

   * TPM manufacturer (from EK certificate issuer)

   * TPM model or family (from EK certificate extensions)

   * Hardware fingerprint: SHA-256 hash of the EK public key

4.4. Authentication-Results Integration

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

```
Authentication-Results: mx.example.com;
  tpm-attest=pass
    header.alg=RS256
    header.mfr=INTC
    header.hw=tpm2.0
    header.chip=sha256:a1b2c3d4...
```

Where:

tpm-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.alg The signature algorithm from the TPM-Attestation header.

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

header.hw The hardware type. "tpm2.0" for a hardware or firmware TPM. "vtpm2.0" for a hypervisor-provided virtual TPM.

header.chip The SHA-256 hash of the EK public key, truncated to the first 16 hex characters. This serves as a compact hardware fingerprint for reputation tracking. Full fingerprint SHOULD be available via an extended property.

4.5. Privacy Considerations for Direct Attestation

Direct TPM Attestation reveals the sender's hardware fingerprint (EK public key hash). This fingerprint is stable across all emails sent by the same machine, enabling linkability. This is a deliberate design choice: linkability is desirable for reputation systems (a hardware identity that sends spam can be permanently flagged) and is the mechanism by which Sybil resistance is achieved.

Senders who require unlinkability between messages SHOULD use Mode 2 (SD-JWT Trust Proof, Section 5) instead of or in addition to Direct TPM Attestation.

The EK certificate also reveals the TPM manufacturer and model family. In most deployments, this information is not sensitive. If the sender considers hardware provenance information sensitive, they SHOULD use Mode 2 exclusively.

5. Mode 2: SD-JWT Trust Proof

This section defines an SD-JWT-based trust proof. SD-JWT ([RFC9901]) is inherently transport-independent; see Appendix C for applicability beyond email.

5.1. Overview

SD-JWT Trust Proof provides an alternative attestation mechanism where a Trust Registry (an entity that has previously verified the sender's hardware attestation) issues a Selective Disclosure JWT [RFC9901] containing claims about the sender's trust classification. The claim structure is informed by the Entity Attestation Token (EAT) model [RFC9711].

The sender can then present this SD-JWT to email recipients with only the claims it chooses to disclose. For example, a sender might prove "my hardware trust level is sovereign" (indicating a genuine, non-virtual TPM was verified) without revealing its identity, hardware fingerprint, or registration date. The Trust Registry issuer (iss) is always visible in the SD-JWT (required for signature verification), but all other claims are selectively disclosable.

This mode requires the recipient to trust the Trust Registry's signing key (obtainable via the registry's JWKS endpoint). It does NOT require trust in any TPM manufacturer CA, as the Trust Registry has already performed that verification.

5.2. Header Field Definition

```
TPM-Trust-Proof = "TPM-Trust-Proof" ":" SP sd-jwt-compact CRLF

sd-jwt-compact = sd-jwt "~" *( disclosure "~" ) kb-jwt

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

The header value is a complete SD-JWT presentation as defined in [RFC9901], including optional disclosures and a Key Binding JWT (KB-JWT).

The SD-JWT payload MUST include the following claims:

iss The Trust Registry's identifier (a URI). Used to locate the Trust Registry's JWKS for signature verification.

iat Issuance time as a NumericDate.

exp Expiration time as a NumericDate. MUST NOT exceed 7 days after iat.

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

cnf Confirmation claim containing the sender's public key (JWK format). When the sender's key is TPM-bound, this enables holder binding: the recipient can verify that the presenter possesses the TPM-resident private key corresponding to this public key.

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

trust_tier The sender's hardware trust classification. Values defined by the Trust Registry (e.g., "sovereign" for genuine non-virtual TPM, "virtual" for hypervisor-provided TPM, "declared" for software-only).

sub The sender's identifier within the Trust Registry.

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

hsm_manufacturer The TPM manufacturer code (e.g., "INTC", "AMD").

enrolled_at The timestamp at which the sender enrolled with the Trust Registry.

5.3. Verification Algorithm

1. Parse the TPM-Trust-Proof header as an SD-JWT presentation per [RFC9901].

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

3. Fetch the issuer's JWKS from the well-known endpoint (e.g., {iss}/.well-known/jwks.json) or from a locally cached copy. The JWKS SHOULD be cached with 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.

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

7. If a Key Binding JWT (KB-JWT) is present and the SD-JWT contains a cnf claim, verify the KB-JWT signature against the cnf key. This proves the presenter holds the private key (which, if TPM-bound, proves the email was composed on the enrolled hardware).

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

5.4. Authentication-Results Integration

```
Authentication-Results: mx.example.com;
  tpm-trust=pass
    header.trust_tier=sovereign
    header.registry=registry.example.com
    header.kb=tpm-bound
```

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

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

header.registry The Trust Registry identifier (from the iss claim).

header.kb "tpm-bound" if the KB-JWT was verified against a cnf key known to be TPM-resident. "software" if the key binding used a software key. "none" if no KB-JWT was present.

6. Combined Mode

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

* Receivers that trust TPM manufacturer CAs but not the Trust Registry SHOULD verify TPM-Attestation and ignore TPM-Trust-Proof.

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

7. 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.

7.1. Direct TPM Attestation (Mode 1)

Each TPM contains exactly one EK, which produces exactly one EK certificate. The EK public key hash serves as a unique hardware fingerprint. A receiving mail server that tracks these fingerprints can enforce policies such as:

* Rate limiting per hardware fingerprint

* Reputation scoring per hardware fingerprint

* Blocking hardware fingerprints associated with abuse

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

7.2. SD-JWT Trust Proof (Mode 2)

Sybil resistance in Mode 2 depends on the Trust Registry's enrollment policy. A Trust Registry that issues trust_tier values indicating hardware verification (e.g., "sovereign") MUST verify hardware attestation at enrollment time and MUST maintain a uniqueness registry enforcing at most one identity per EK public key hash. Such a Trust Registry provides equivalent Sybil resistance to Mode 1, with the additional benefit that the hardware fingerprint is not revealed to email recipients.

Trust Registries 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.

A Trust Registry that does not perform hardware verification (e.g., one that enrolls software-only agents) provides no Sybil resistance and MUST NOT issue trust_tier values that imply hardware verification. The trust_tier claim distinguishes these cases, enabling recipients to weight trust appropriately.

7.3. Virtual TPMs

Virtual TPMs (vTPMs) provided by hypervisors (VMware, Hyper-V, QEMU) have valid EK certificates signed by the hypervisor vendor's CA, but the hypervisor operator can create arbitrarily many vTPMs. This weakens Sybil resistance.

Receiving mail servers SHOULD distinguish between physical TPMs and virtual TPMs. The EK certificate's issuer field identifies the manufacturer: VMware-issued EK certificates indicate a virtual TPM. The header.hw property in Authentication-Results ("tpm2.0" vs "vtpm2.0") communicates this distinction.

Mail servers MAY apply different policies to virtual TPM attestations (e.g., lower trust weighting, stricter rate limits) while still recognising them as superior to no attestation at all.

8. 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. TPM-Attestation

provides hardware-level signing. They protect different properties: DKIM proves domain authorisation; TPM-Attestation proves hardware identity. Both SHOULD be used.

DMARC [RFC7489] Ties SPF and DKIM together with policy. A future extension to DMARC could incorporate TPM 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 TPM-Attestation and TPM-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 TPM-Attestation solve different problems: S/MIME proves "this email address is controlled by someone who presented a valid certificate"; TPM-Attestation proves "this email was composed on genuine, unique hardware". They can coexist.

8.1. Multiple Headers and Forwarding

If multiple TPM-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 TPM-Attestation header. This is expected and analogous to DKIM signature breakage through body modification. Such intermediaries SHOULD preserve the original TPM-Attestation header (the verifier will record "fail" due to body hash mismatch) and MAY add their own TPM-Attestation header covering the modified body, if the intermediary itself has TPM capability. ARC [RFC8617] SHOULD be used to preserve the original authentication results from before the modification.

9. 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].

9.1. 1id.com Trust Registry

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

Description: A working implementation of both Mode 1 (Direct TPM Attestation) and Mode 2 (SD-JWT Trust Proof) as described in this draft. The implementation includes server-side EK certificate chain validation against manufacturer root CAs (Intel, AMD, Infineon, STMicroelectronics, Nuvoton, Qualcomm), anti-Sybil enforcement via a one-EK-per-identity registry, and SD-JWT trust proof issuance with selective disclosure. The system also supports PIV-based (YubiKey) attestation for portable hardware tokens.

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

Coverage: All protocol features described in this draft are implemented.

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".

* *TPM/PIV 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, privilege elevation, and AK provisioning.

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

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

10. IANA Considerations

10.1. Header Field Registration

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

TPM-Attestation Applicable protocol: mail. Status: standard. Reference: this document, Section 4.2.

TPM-Trust-Proof Applicable protocol: mail. Status: standard. Reference: this document, Section 5.2.

10.2. Authentication Method Registration

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

tpm-attest Definition: this document, Section 4.4. Applicable versions: 1.

tpm-trust Definition: this document, Section 5.4. Applicable versions: 1.

11. Security Considerations

11.1. Manufacturer CA Compromise

If a TPM manufacturer’s root CA private key is compromised, an attacker could forge EK 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 Trust Registry 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. This is analogous to how Certificate Transparency monitors detect mis-issued TLS certificates.

11.2. Replay Attacks

The timestamp (ts) in the TPM-Attestation header limits the replay window. Receiving mail servers SHOULD reject attestations with timestamps more than 300 seconds from the current time. 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, the Key Binding JWT includes an iat claim and MAY include an aud claim binding the presentation to a specific recipient.

11.3. Compromised Endpoints and Privilege Requirements

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

Additionally, TPM operations provide a practical barrier against casual malware. Initial provisioning of an Attestation Key - the one-time setup step that creates the AK and certifies it against the EK - typically requires elevated operating system privileges (administrator on Windows via the TPM Base Services API, or root on Linux via /dev/tpmrm0). This means that unprivileged userspace malware (the most common class, distributed via phishing, drive-by downloads, and browser exploits) cannot provision new attestation keys, even if it runs on a machine with a TPM.

Once an AK has been provisioned, subsequent signing operations (TPM2_Sign) MAY be available to unprivileged processes depending on the TPM's authorization policy and operating system configuration. Implementations that wish to restrict signing to authorised software SHOULD configure appropriate TPM authorization policies (e.g., password or policy session requirements on the AK).

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

11.4. Physical Attacks on TPMs

Extracting private keys from a TPM requires physical attacks such as electron microscopy, laser fault injection, or side-channel analysis. Modern TPMs 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 TPMs to create additional identities.

11.5. Virtual TPM Risk

As noted in Section 7.3, virtual TPMs do not provide the same Sybil resistance as physical TPMs because the hypervisor operator can create arbitrary numbers of vTPMs. Receiving mail servers MUST be able to distinguish virtual from physical TPMs (via the manufacturer code in the EK certificate) and SHOULD apply appropriate policy differences.

11.6. Policy Abuse and Deanonymisation Risk

Mode 1 (Direct TPM Attestation) creates a persistent, globally unique hardware fingerprint that is visible to every recipient. While this is the intended mechanism for Sybil resistance and reputation building, it also creates risks if misused as a surveillance tool:

* Authoritarian regimes could correlate hardware fingerprints across messages to track individuals or organisations.

* Employers or platforms could mandate attestation to deanonymise senders who have legitimate reasons for pseudonymity.

* Receiving mail servers that log hardware fingerprints create long-lived tracking databases.

These risks are mitigated by the availability of Mode 2 (SD-JWT Trust Proof), which provides Sybil resistance without revealing the hardware fingerprint. Senders operating in contexts where hardware fingerprint disclosure is unacceptable SHOULD use Mode 2 exclusively.

Receiving mail servers SHOULD NOT require Mode 1 attestation when Mode 2 provides sufficient trust signal for the receiver’s policy needs. The Applicability section (Section 3) notes that this specification is designed for automated senders, not human users, further limiting the deanonymisation risk in practice.

11.7. Header Stripping and Modification

An intermediary mail server could strip or modify the TPM-Attestation or TPM-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, and DMARC policy can specify handling for messages that lose authentication in transit.

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.

12. 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>.

[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>.

[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>.

[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>.

13. Informative References

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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)
TPM Attestation	~$500-2000	$500M-2B	No (chip is permanent)

Table 1

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

Appendix B. Appendix: TPM Manufacturer Root CAs

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

Intel Provides EK CA certificates via a REST API. Root CA certificates available for download. See https://trustedservices.intel.com/.

AMD AMD Security Processor key distribution. Root certificates for fTPM EK validation. See 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.

Receiving mail servers implementing this specification SHOULD maintain a local trust store of TPM 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: Applicability to Other Transports

The CMS attestation bundle (the "chain" value defined in Section 4.2) and the SD-JWT trust proof (defined in Section 5.2) are self-contained data structures whose verification algorithms (Section 4.3 and Section 5.3) 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.

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 D. Acknowledgements

The concept of using hardware attestation for email sender verification was developed in the context of a hardware identity registrar for AI agents. The authors thank the Trusted Computing Group for the TPM 2.0 specification, the authors of RFC 9901 (SD-JWT) for the selective disclosure mechanism, and the IETF RATS and SEAT working groups for establishing the remote attestation architecture that this document builds upon.

Author’s Address

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