Network Working Group
Internet-Draft
Intended status: Informational
Expires: 5 December 2026

B. Laurie
T. Santoro
P. Anthonysamy
S. de Haas
A. Mathur
Google LLC

3 June 2026

A Standard for Claiming Transparency and Falsifiability
draft-laurie-tmif-01

Abstract

This document specifies a Transparency Metadata Interchange Format (TMIF) that allows a system to make standardized, verifiable claims about its levels of transparency and falsifiability. TMIF is designed to be agnostic to specific threat models or mitigations, serving purely as a communication vehicle for system providers to declare how their security and privacy controls can be verified by third-party evaluators.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-laurie-tmif/.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 December 2026.

Table of Contents

1. Introduction

As AI powered consumer facing features proliferate, AI experiences are leveraging increasing amounts of sensitive data to provide greater utility. This begs the question of how to assure the user around data privacy and how their data is processed by the AI. Along with users; regulators, device manufacturers and the larger ecosystem are all looking for ways to increase transparency around AI models. Privacy preserving systems give service providers a way to make strong claims about how data is handled, and users a way to independently verify those claims. Transparency approaches can, however, use different approaches, therefore some consistency in definitions and terminology is required in order to allow end users

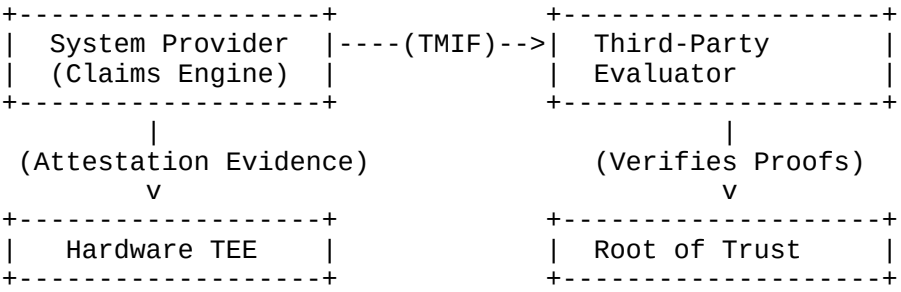
(or their delegates) to examine these systems in detail and assess the overall veracity of the claims they make.

This document therefore defines a Transparency Metadata Interchange Format (TMIF) for describing how transparency is achieved by such a system, in order to effectively allow evaluators to determine the overall level of transparency that the system can claim. TMIF is an agnostic standard for communicating the levels of transparency achieved by a system's components.

Crucially, TMIF separates the _communication_ of transparency from the _evaluation_ of threats and mitigations. It does not prescribe specific threat models (e.g., STRIDE) or scoring weights. Instead, it provides a standardized JSON structure for system providers to list mitigated threats (via standard URNs), describe the applied mitigations, and declare the transparency level (the verifiable "proof") of those mitigations. Evaluators, researchers, and other 3rd parties can then use these TMIF documents as inputs for their own qualitative or quantitative assessments.

A high degree of transparency means that end consumers, or their delegates, can assure themselves and have higher confidence around a service provider's claims about the usage and handling of their data. It is also possible to incentivize third parties and evaluators to focus their efforts towards finding falsifying examples of privacy and security claims.

While this setup bears significant relation to [I-D.ietf-rats-ar4si], this draft focuses specifically on communicating about this transparency property.



2. Conventions and Definitions

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 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.1. System Component Definitions

A system that is set up to provide transparency and falsifiability will generally include the following components, which may have details specified in the TMIF:

Accelerators When a TEE offloads computation to a hardware accelerator (e.g., a GPU, NPU), then in general the entire data pathway will be secured. This implies a mutually authenticated and encrypted channel between the host CPU’s TEE and a TEE within the accelerator itself. The accelerator would also have its own capacity for secure processing and attestation. The TMIF format should be able to describe all of these implementation details.

Attester The entity (comprising both hardware and software elements) that creates _Evidence_ about its state, configuration, or environment. It exposes this cryptographically bound data to prove its trustworthiness to a remote peer. For example, in a mobile application context, the handset’s hardware-isolated Secure Element or Trusted Execution Environment (TEE) acts as the Attester to generate secure cryptographic proofs.

Attestation Results A structured, cryptographically signed statement emitted by a Verifier after evaluating an Attester’s Evidence. Attestation Results convey the Verifier’s formal appraisal of the Attester’s security posture and trustworthiness, enabling a Relying Party to make safe, risk-based decisions about whether to trust the Attester with sensitive data.

Claimant The entity (such as a device OEM or service provider) generating the TMIF document to assert the transparency of their system.

Client The device (e.g., a smartphone, laptop, or web browser) used to interact with the service. Includes software and hardware.

Evaluator A third party (such as the App Defense Alliance) that ingests the TMIF document, verifies the artifacts, and applies their own threat modeling and scoring methodologies to approve or deny the system.

Evidence A set of claims about an Attester's state, configuration, or execution environment that is cryptographically bound and signed by the Attester. Evidence is intended to be consumed by a Verifier, which evaluates its authenticity and structural integrity to determine whether the Attester can be considered trustworthy.

Falsifiability The design property ensuring that any accidental bug or malicious subversion of a system's stated privacy or security claims inherits a high risk of independent discovery. A claim is highly falsifiable if an evaluator can readily devise and execute tests to find definitive counterexamples to it.

Isolation TEE environments provide memory protection, even from software with higher privilege.

Remote attestation As per RATS Architecture [RFC9334], remote attestation consists of an interaction wherein an attester presents evidence about itself to enable a remote peer to decide whether or not to consider the attester trustworthy or not.

Relying Party The entity that ultimately consumes the Attestation Results from the Verifier to make an informed, risk-based decision. In this architecture, the Relying Party depends on these verified results to determine whether or not to trust the Attester with sensitive data, allow access to enterprise services, or execute specific application logic (such as processing protected user data in plaintext).

Server The machine that serves user requests. In a distributed system, this includes all machines that have access to the data.

Service Operator The entity that operates the service. This entity processes the data and considered an untrusted party in the security model, as the goal is to constrain access to user data in plaintext.

Service Provider The entity that develops, and deploys the service. This entity defines data processing logic and is considered an untrusted party in the security model, as the goal is to constrain access to user data in plaintext.

TEE A hardware-isolated secure processing environment that protects the confidentiality and integrity of code and data executing within it, and can produce attestations about the state of the secured environment.

For the purposes of this standard, we define a TEE as a secure area that keeps code and data loaded inside it, usually a hardware TEE as mentioned above. Code and data in TEEs are protected by confidentiality and integrity: data confidentiality prevents unauthorized entities from outside the TEE from reading data, while code integrity prevents code in the TEE from being replaced or modified by unauthorized entities. Crucially, an unauthorized entity can include the owner/maintainer of the code and data inside the TEE.

TEE Manufacturer The hardware vendor that designs and manufactures the processor containing the TEE.

Transparency The property of a system configuration, source code, or binary execution environment being inspectable, and verifiable by external parties. In TMIF, transparency is quantified by a tiered level system (1-5) representing the robustness of the available technical proof.

Trusted Computing Base As per the National Institute of Standards and Technology [NIST-TCB], we consider a trusted computing base to encompass the totality of protection mechanisms within a computer system, including hardware, firmware, and software, the combination responsible for enforcing a privacy policy. For the purposes of this document, transparency efforts will generally pertain only to the trusted computing base. All other components of a system do not need to be transparent in order for an evaluator to be able to assess a system's privacy and security claims.

Verifier A trusted core component that ingests the _Evidence_ generated by the Attester. It validates the cryptographic authenticity of the claims (checking signatures, certificates, and roots of trust) and evaluates them against specific appraisal policies. The Verifier then outputs an appraisal result, known as _Attestation Results_, which is a structured, trusted statement about the Attester's security posture.

3. Falsifiability of Privacy Claims

The aim of specifying the transparency metadata for a given system's components is to allow evaluators to assess the level of falsifiability of the system. A fully transparent system is designed in such a way to ensure that an accidental or malicious attempt to undermine privacy and security claims cannot be introduced without a risk of discovery. This is based on the principle that testing may show the presence of bugs or backdoors, but never their absence. The higher the falsifiability of a claim, the more likely it is for

someone to find a counterexample to it, in the event that claim were in fact false (e.g. because a bug or backdoor were introduced, accidentally or intentionally). In practice, a high degree of falsifiability also makes it difficult for an insider with highly privileged access to purposely subvert the claims of the system without risking discovery in doing so.

The chief criterion for inclusion of metadata in the algorithmic calculation of levels is its role in supporting claims that increase falsifiability.

3.1. Transparency Levels

We expect that evaluators will assess the claimant’s mitigations based on the level of "proof" provided. The following standardized transparency levels define the tiers of falsifiability:

Level	Name	Technical Evidence Requirement
1	Level 1	Binary is publicly available.
2	Level 2	Level 1 features + binary is executable.
3	Level 3	Source code is publicly available.
4	Level 4	Level 3 features + code is reproducibly buildable.
5	Level 5	Formal proof of security properties is available.

Table 1: Standardized Transparency Levels

4. Transparency Metadata Interchange Format (TMIF)

The Transparency Metadata Interchange Format is a JSON-based schema designed to communicate a system’s composition, the claims it makes against standardized threats, the mitigations implemented, and the transparency level of those mitigations.

To support complex architectures, TMIF documents catalog claims at the *component* level, but enforce an aggregate *system-wide transparency lower bound* (representing the "weakest link" of the system as a whole).

A system claiming compliance with this framework MUST expose a verifiable Transparency Metadata Interchange Format (TMIF) record.

Evaluators SHOULD be able to cryptographically verify these claims without out-of-band protocols.

4.1. Example TMIF Document

This section provides a sample TMIF document illustrating the component and claim structure.

```
{
  "system_identifier": "urn:example:system:paic-compute-v1",
  "system_version": "1.0.0",
  "valid_until": "2026-12-31T23:59:59Z",
  "transparency_level_lower_bound": 2,
  "components": [
    {
      "component_identifier": "urn:example:component:inference-node",
      "component_version": "1.0.0",
      "claims": [
        {
          "threat_identifier":
              "urn:ada:threat:information-disclosure:data-in-transit",
          "mitigation_description":
              "Workloads run in TEE VMs; data encrypted via TLS 1.3.",
          "transparency_level": 4,
          "artifacts": [
              "https://github.com/example/network/releases/tag/v1.0",
              "https://example-transparency-log.org/entries/12345"
          ]
        },
        {
          "threat_identifier": "urn:ada:threat:tampering:code-change",
          "mitigation_description":
              "Workload runs in a hardware TEE w/ Remote Attestation.",
          "transparency_level": 2,
          "artifacts": [
              "https://example.com/binaries/inference-node-v1.0.bin"
          ]
        }
      ]
    }
  ]
}
```

NOTE: This is currently specified in JSON format for the purposes of providing an example, but the preferred format will depend on emerging use cases and we invite further discussion on this particular point. Individual values will be defined elsewhere in a decentralized way (e.g. on separate websites / GitHub readmes, etc.).

4.1.1. TMIF Field Definitions

The fields used in the Transparency Metadata Interchange Format are defined below:

system_identifier (String) A unique URN or identifier for the overall system being assessed.

system_version (String) The version string of the overall system.

valid_until A timestamp indicating when the TMIF document ceases to be valid, expressed as an Internet date-time string in full-date "T" full-time format conforming to [RFC3339].

transparency_level_lower_bound (Integer) The lowest (weakest) transparency level found across all mitigations in all components of the system. This reflects the aggregate health of the system.

components (Array) A list of the distinct architectural components that make up the system (e.g., stateless inference service, stateful memory store).

- component_identifier (String) A unique URN identifying the specific component.
- component_version (String) The version of the component.
- claims (Array) A list of mitigation claims associated with this component.
  - threat_identifier (String) A standardized URN representing the specific threat being mitigated (e.g., referencing a canonical threat library maintained by an evaluator like the App Defense Alliance).
  - mitigation_description (String) A plain-text description of how the threat is mitigated (the technical solution).
  - transparency_level (Integer) The transparency level (1-5) representing the available proof for this mitigation.

artifacts (Array of Strings) A simple list of URIs pointing to the verifiable evidence for the transparency level (e.g., links to open-source repositories, transparency logs, binary downloads, or audit reports).

4.2. Usage of Transparency Metadata

The separation of concerns in TMIF allows it to be highly flexible:

1. *The Claimant (System Provider):* Identifies the threats relevant to their system (using evaluator-provided URNs), describes their mitigations, self-assesses their transparency level, and links to the artifact proofs in the TMIF document.

2. *The Evaluator (Auditing Body):* Parses the TMIF document, verifies the URIs in the artifacts array, and confirms the transparency_level. The evaluator then applies their own independent scoring methodology (e.g., mapping STRIDE threats to weights, multiplying weights by the TMIF transparency level) to arrive at an approval decision. IETF explicitly does not standardize the weights or the threat URNs, leaving that to the domain expertise of the evaluators.

5. Security Considerations

TMIF documents communicate security and privacy claims but do not themselves enforce them. The integrity and authenticity of a TMIF document are paramount; if an attacker can maliciously alter a TMIF payload in transit, an evaluator could be misled into trusting a compromised or subverted component.

5.1. Claim Tampering and Integrity Protection

To mitigate the risk of claim tampering, any system claiming compliance with this specification MUST ensure that TMIF payloads are cryptographically protected at the data layer.

* *Data Structure Protection:* TMIF payloads MUST be encapsulated in a format that supports cryptographic signatures and data integrity proofs. Implementations MUST use JSON Web Signatures (JWS) [RFC7515].

* *Signature Verification:* The Relying Party or Verifier MUST cryptographically validate the signature of the TMIF record against a known, trusted Root of Trust before parsing or appraising the metadata. Payloads with invalid, missing, or malformed cryptographic signatures MUST be rejected immediately.

* *Transport Security:* While payload-level signing is required, TMIF transport channels SHOULD additionally utilize Transport Layer Security (TLS) [RFC8446] to ensure confidentiality and prevent traffic analysis.

5.2. Replay Attack Mitigations

To prevent replay attacks where an historic, validly signed TMIF payload is substituted to mask a current compromised state, TMIF documents MUST include a cryptographically bound freshness mechanism.

* Implementations MUST include a ’valid_until’ timestamp parameter.

* Verifiers MUST reject TMIF payloads if the current system time is past the designated expiration window.

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. 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/rfc/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/rfc/rfc3339>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[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/rfc/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

7.2. Informative References

[I-D.ietf-rats-ar4si]
Voit, E., Birkholz, H., Hardjono, T., Fossati, T., and V. Scarlata, "Attestation Results for Secure Interactions", Work in Progress, Internet-Draft, draft-ietf-rats-ar4si-10, 18 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-rats-ar4si-10>.

[NIST-TCB] National Institute of Standards and Technology (NIST), "Glossary: Trusted Computing Base (TCB)", 2026, <https://csrc.nist.gov/glossary/term/trusted_computing_base>.

[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/rfc/rfc9334>.

Acknowledgments

Authors' Addresses

Ben Laurie
Google LLC
Email: benl@google.com

Tiziano Santoro
Google LLC
Email: tzn@google.com

Pauline Anthonysamy
Google LLC
Email: anthonysp@google.com

Sarah de Haas
Google LLC
Email: dehaass@google.com

Ankur Mathur
Google LLC
Email: ankurmathur@google.com