Network Working Group
Internet-Draft
Intended status: Informational
Expires: 23 April 2026

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

20 October 2025

A Standard for Claiming Transparency and Falsifiability
draft-laurie-tmif-00

Abstract

This document specifies a transparency metadata interface format that allows a system to make claims about its levels of transparency and falsifiability.

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/sarahdeh/draft-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 23 April 2026.

Copyright Notice

Table of Contents

1. Introduction

As AI powered consumer facing features proliferate, AI service providers and users are grappling with increasing amounts of user discomfort over the increasing amounts of sensitive data powered by these systems. While this isn't a new problem, AI accelerates the need to process sensitive information in order to provide significant advances in utility. However, many consumers want to be reassured that there are strict limits on how their data is used, who can access it, and for what purpose. For that reason, we see an increasing trend to introduce transparency into privacy preserving systems, giving 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, be radically different, 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 falsifiability of the claims they make.

This document therefore defines a Transparency Metadata Interchange Format (TMIF) for describing levels of transparency achieved by such a system, in order to effectively allow independent auditors to determine the overall level of transparency that the system can claim.

A high degree of transparency means that end consumers, or their delegates, can assure themselves that a service provider's claims about the usage and handling of their data are very likely true because of the high technical bar that would have to be met to undermine those claims without discovery. It is also possible to incentivize third parties and independent researchers to focus their efforts towards finding falsifying examples of privacy and security claims.

2. Conventions and Definitions

Defining a target system for the transparency metadata interchange format

A system that is set up to provide transparency and falsifiability will generally include the following components, with characteristics as described, which may have transparency related details specified in the TMIF.

Service Provider

The entity that develops, deploys, and operates 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 Manufacturer

The hardware vendor that designs and manufactures the processor containing the TEE. This entity is a root of trust; it produces the TEE.

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.

Client

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

Server

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

Remote attestation

A cryptographic process that provides a remote client with verifiable proof of the TEE's state. The service generates a signed report, or attestation, that proves key information:

1. The hardware is a genuine TEE from a specific manufacturer.

2. The correct, unmodified application binaries are running within it.

3. The TEE's software version.

The attestation allows the client to verify the integrity of the trusted environment before provisioning it with any sensitive data.

See more in the section below on Transparency properties for who should read these attestations.

Root of Trust

In practice most TEEs will be hardware based, with the cryptographic keys fused into the CPU silicon during hardware manufacturing. This provides a high-degree of physical tamper resistance.

However, a software root of trust is also possible, therefore details of its assurance level should be specified as part of the TMIF format (e.g. has completed a vulnerability analysis at Evaluation Assurance Level (EAL) Level 5 or higher.).

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.

Isolation

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

3. Falsifiability of privacy claims

The aim of specifying the transparency metadata for a given system is to allow third parties 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 secretly. 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.

4. Examples of Claims

Claim: The service provider cannot access user data in plaintext.

How can it be falsified? - _Remote Attestation:_ Before a client sends any data, it can receive a cryptographic attestation (or signed report) from the TEE. This attestation proves which software is running inside the TEE. If the service provider were to run a modified version of the software to access plaintext data, the attestation would change, and a client could detect this and refuse to send data.

* _Binary Transparency:_ When a binary is publicly available, anyone can download it to perform reverse engineering and search for backdoors. A higher level of assurance is achieved when the

binary is open source and has a reproducible build. This allows anyone to build the binary from the public source code and confirm that the resulting binary matches the one being attested to by the TEE, providing strong confidence that the running code corresponds to the public source code.

Claim: Data that leaves the TEE is limited to privacy preserving, aggregated analytics, and it’s not possible to link it to a specific user.

How can it be falsified? - _Non-targetability:_ One could attempt to falsify this claim by analyzing the egressed data to demonstrate a method for de-anonymization and re-linking to an individual.

5. Usage of Transparency Metadata

We expect that users of the TMIF would comprise the claimant (e.g. a service provider), and a third party wishing to perform an evaluation of said claims.

The claimant will perform the following functions: - Specification of the roots of trust, for example whether the RoT is hardware based, software based, or other, and what kind of system if so - Specification of the application level claims, for example that an application keeps data private from the service provide - Adding any additional fields that might help to increase the overall falsifiability level of the application level claims

The evaluator will assess the information provided by the TMIF and determine whether they trust the claimant based on the evidence provided. We expect the evaluator to calculate an algorithmic assessment based on the provided values. The evaluator can separately publish their own requirements for the claimant’s system.

6. Transparency Levels

We expect that the evaluators will calculate an algorithmic assessment that would class the service provider’s claims in different levels or buckets of transparency and falsifiability. The following levels are proposed 1. binary is publicly available 2. L1 + is executable 3. L2 + is reproducibly buildable 4. Source is available 5. Formal proof of security properties of the source/ binary are available

7. Transparency Metadata Interchange Format

This metadata interchange format allows for a way of computing an overall transparency level for a distributed system. Here is an example TMIF instance, as a JSON object:

{ "transparency_level": { "remote_attestation_roots_of_trust": ["amd_sev_snp", "nvidia_h100"], "application_level_claims": ["https://github.com/some-claim-index/oak/blob/main/docs/tr/claim/18136.md", "https://github.com/another-claim-index/claim/292382.md",], "transparency_level_lower_bound": 4, // more fields may be added here } }

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.). Fields are defined below.

Fields are defined below.

_remote_attestation_roots_of_trust_

For a single node, this is the root of trust that provides the attestation to which the encrypted and attested communication channel is bound. e.g. a hardware provider.

If the node is an accelerator, this is the manufacturer of the accelerator.

For the overall system, this field is the union of the roots of trust of the transitive closure of the connectivity graph of the system, starting from the entry point node.

In general, it is desirable for this set to be as small as possible, and not to include the service provider. Future versions of this document may introduce a more sophisticated semantics for the roots of trust, such as a boolean expression that combines entities with logical operators (AND, OR, k-of-n, etc.). TODO: describe key provisioning practices

_application_level_claims_

For a single node, this is the list of falsifiable claims attached to the application.

_transparency_level_lower_bound_

For a single node, this is the lower bound of the transparency level of all the published binaries.

For a graph, this is the lower bound of the transparency levels of all the nodes.

Optional Fields

The following transparency attributes are considered useful additional information about system transparency, and may be added as fields with corresponding values in the transparency message above.

Publication of binaries in a verifiable audit log

published_binaries Expected value: URI of the log

Binaries or binary hashes should be published in a verifiable audit log, such as Trillian. This allows for a record of releases to ensure that malicious code is not pushed to the server and then later changed to avoid discoverability.

Open Sourcing privacy-critical system components (policy enforcement components)

open_sourced_policy_enforcement Expected value: location of policy enforcement OSS

It is possible to prove certain attributes of a system without fully open sourcing the entire workload, but open sourcing key policy enforcing software. A straightforward example would be open sourcing a sandboxing system, which enforces the policy that data cannot egress.

Open Sourcing key system components (evaluation tools)

open_sourced_eval_tools Expected value: location of evaluation tools

It is also possible to make meaningful claims about a system by running an evaluation script that is open sourced against a closed source model or data set inside a TEE

Full OSS including workload

fully_open_sourced Expected value: location of system OSS

It is possible for the entire workload within the TEE to be fully open sourced. This may be desirable to prove adherence to policies that cannot be proved by a separate policy enforcement layer and instead are part of the workload itself.

Reproducible builds

reproducible_builds Expected value: boolean

In order to provide meaning to binaries published in a verifiable audit log, it is necessary to match them to associated source code. Therefore published source code that is intended as part of a 'proof' of system behavior must be reproducibly buildable to match the published binaries.

Publicly available remote attestation evidence

public_remote_attestation Expected value: path to attestation endpoint

In order to confirm that the binaries published in the verifiable audit log are the same binaries that run on the server, it is necessary to inspect the remote attestation from the TEE. At higher transparency levels this inspection should be possible for anyone, in principle, to complete, therefore it must be made possible for an attestation request to be made from any machine.

Auditability

data_egress_audit_logs Expected value: location of log

If highly privileged 'break glass' type access is used, this field can be used to declare an audit log that ensures that this type of access cannot be used secretly

8. Security Considerations

This section will certainly be filled in later as the discussion progresses.

9. IANA Considerations

This document has no IANA actions.

Acknowledgments

TODO acknowledge.

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