SCITT
Internet-Draft
Intended status: Standards Track
Expires: 16 January 2026

H. Birkholz
Fraunhofer SIT
A. Delignat-Lavaud
C. Fournet
Microsoft Research
Y. Deshpande
ARM
S. Lasker
15 July 2025

An Architecture for Trustworthy and Transparent Digital Supply Chains
draft-ietf-scitt-architecture-15

Abstract

Traceability in supply chains is a growing security concern. While verifiable data structures have addressed specific issues, such as equivocation over digital certificates, they lack a universal architecture for all supply chains. This document proposes a scalable architecture for single-issuer signed statement transparency applicable to any supply chain. It ensures flexibility, interoperability between different transparency services, and compliance with various auditing procedures and regulatory requirements.

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-ietf-scitt-architecture/.

Discussion of this document takes place on the SCITT Working Group mailing list (mailto:scitt@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/scitt/. Subscribe at https://www.ietf.org/mailman/listinfo/scitt/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-scitt/draft-ietf-scitt-architecture.

Status of This Memo

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

This Internet-Draft will expire on 16 January 2026.

Copyright Notice

Table of Contents

1. Introduction

This document defines an architecture, a base set of extensible message structures, and associated flows to make signed content transparent via verifiable data structures maintained by corresponding transparency services. The goal of the transparency enabled by the Supply Chain Integrity, Transparency, and Trust (SCITT) architecture is to enhance auditability and accountability for single-issuer signed content (statements) that are about supply chain commodities (artifacts). Registering signed statements with a transparency service is akin to a notarization procedure. Transparency services perform notary operations, confirming a policy is met before recording the statement on the ledger. The SCITT ledger represents a linear and irrevocable history of statements made. Once the signed statement is registered, the transparency service issues a receipt, just as a notary stamps the document being notarized. Similar approaches have been implemented for specific classes of artifacts, such as Certificate Transparency [RFC9162]. The SCITT approach follows a more generic paradigm than previous approaches. This "content-agnostic" approach allows SCITT transparency services to be either integrated in existing solutions

or to be an initial part of new emerging systems. Extensibility is a vital feature of the SCITT architecture, so that requirements from various applications can be accommodated while always ensuring interoperability with respect to registration procedures and corresponding auditability and accountability. For simplicity, the scope of this document is limited to use cases originating from the software supply chain domain, but the specification defined is applicable to any other type of supply chain statements (also referred to as value-add graphs), for example, statements about hardware supply chains.

This document also defines message structures for signed statements and defines a profile for COSE receipts [I-D.draft-ietf-cose-merkle-tree-proofs], i.e., signed verifiable data structure proofs). These message structures are based on the Concise Binary Object Representation Standard [STD94] and corresponding signing is facilitated via the CBOR Object Signing and Encryption Standard [STD96]. The message structures are defined using the Concise Data Definition Language [RFC8610]. The signed statements and receipts are based on the COSE_Sign1 specification in Section 4.2 of [STD96]. As these messages provide the foundation of any transparency service implementation for global and cross-domain application interoperability, they are based on complementary COSE specifications, mainly [I-D.draft-ietf-cose-merkle-tree-proofs]. Therefore, support of COSE_Sign1 and extensibility of COSE Header Parameters are prerequisites for implementing the interoperable message layer included in this document.

In summary, this specification supports relying parties obtaining proof that signed statements were recorded and checked for their validity at the time they were registered. How these statements are managed or stored is out-of-scope of this document.

1.1. Requirements Notation

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.

2. Software Supply Chain Scope

To illustrate the applicability of the SCITT architecture and its messages, this section details the exemplary context of software supply chain (SSC) use cases. The building blocks provided by the SCITT architecture are not restricted to software supply chain use cases. Software supply chains serve as a useful application guidance and first usage scenario.

2.1. Generic SSC Problem Statement

Supply chain security is a prerequisite to protecting consumers and minimizing economic, public health, and safety threats. Supply chain security has historically focused on risk management practices to safeguard logistics, meet regulatory requirements, forecast demand, and optimize inventory. While these elements are foundational to a healthy supply chain, an integrated cyber security-based perspective of the software supply chains remains broadly undefined. Recently, the global community has experienced numerous supply chain attacks targeting weaknesses in software supply chains. As illustrated in Figure 1, a software supply chain attack may leverage one or more life-cycle stages and directly or indirectly target the component.

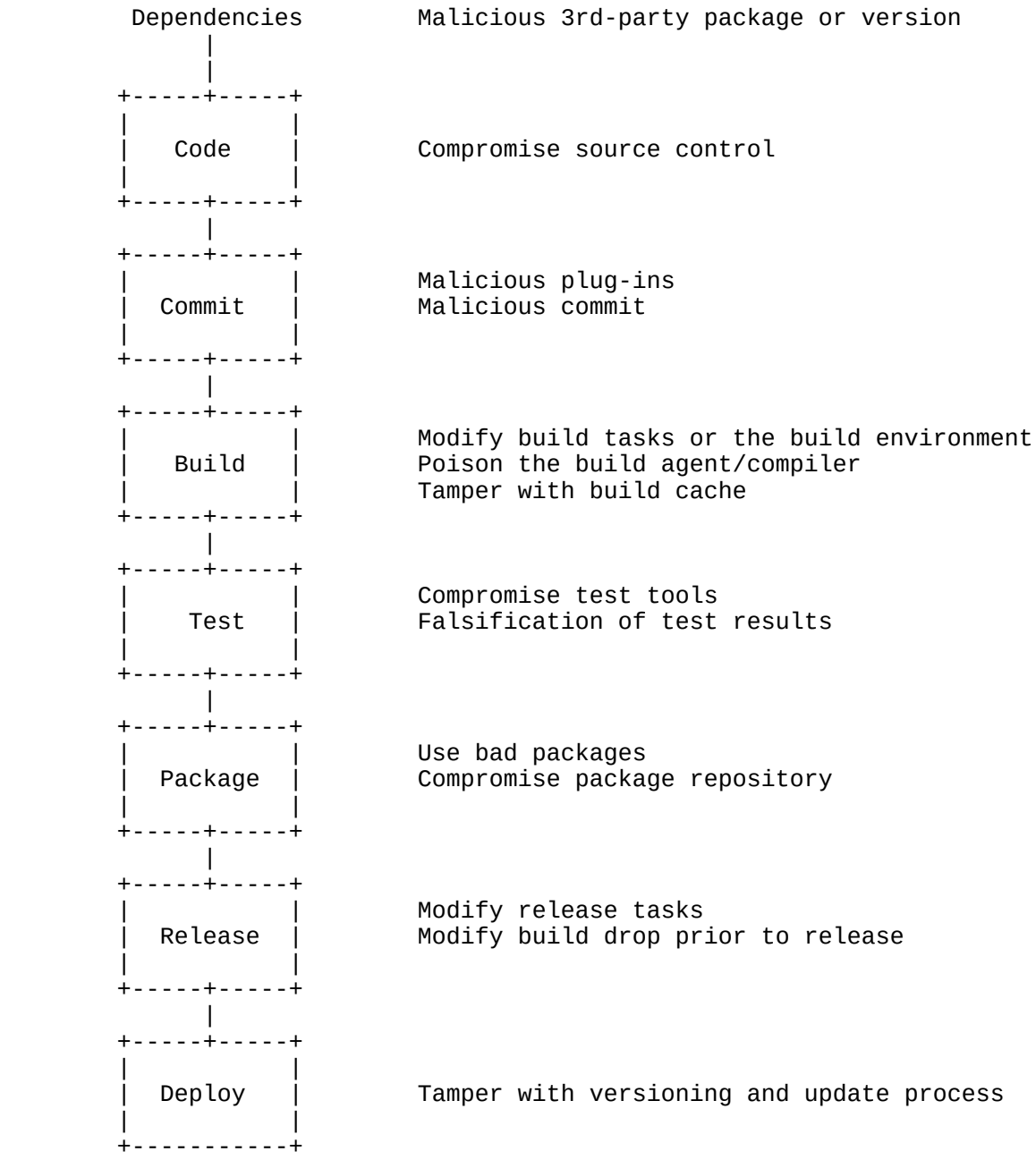


Figure 1: Example SSC Life-Cycle Threats

DevSecOps often depends on third-party and open-source software. These dependencies can be quite complex throughout the supply chain, so checking provenance and traceability throughout their lifecycle is difficult. There is a need for manageable auditability and accountability of digital products. Typically, the range of types of statements about digital products (and their dependencies) is vast, heterogeneous, and can differ between community policy requirements. Taking the type and structure of all statements about digital and products into account might not be possible. Examples of statements may include commit signatures, build environment and parameters, software bill of materials, static and dynamic application security testing results, fuzz testing results, release approvals, deployment records, vulnerability scan results, and patch logs. In consequence, instead of trying to understand and describe the detailed syntax and semantics of every type of statement about digital products, the SCITT architecture focuses on ensuring statement authenticity, visibility/transparency, and intends to provide scalable accessibility. Threats and practical issues can also arise from unintended side-effects of using security techniques outside their proper bounds. For instance digital signatures may fail to verify past their expiry date even though the signed item itself remains completely valid. Or a signature may verify even though the information it is securing is now found unreliable but fine-grained revocation is too hard.

Lastly, where data exchange underpins serious business decision-making, it is important to hold the producers of those data to a higher standard of accountability. The SCITT architecture provides mechanisms and structures for ensuring that the makers of authoritative statements can be held accountable and not hide or shred the evidence when it becomes inconvenient later.

The following use cases illustrate the scope of SCITT and elaborate on the generic problem statement above.

2.2. Eclectic SSC Use Cases

The three following use cases are a specialization derived from the generic problem statement above.

2.2.1. Security Analysis of a Software Product

A released software product is often accompanied by a set of complementary statements about its security properties. This gives enough confidence to both producers and consumers that the released software meets the expected security standards and is suitable to use.

Subsequently, multiple security researchers often run sophisticated security analysis tools on the same product. The intention is to identify any security weaknesses or vulnerabilities in the package.

Initially, a particular analysis can identify a simple weakness in a software component. Over a period of time, a statement from a third-party illustrates that the weakness is exposed in a way that represents an exploitable vulnerability. The producer of the software product provides a statement that confirms the linking of a software component vulnerability with the software product by issuing a product vulnerability disclosure report and also issues an advisory statement on how to mitigate the vulnerability. At first, the producer provides an updated software product that still uses the vulnerable software component but shields the issue in a fashion that inhibits exploitation. Later, a second update of the software product includes a security patch to the affected software component from the software producer. Finally, a third update includes a new release (updated version) of the formerly insecure software component. For this release, both the software product and the affected software component are deemed secure by the producer and consumers.

A consumer of a released software wants to:

* know where to get these security statements from producers and third-parties related to the software product in a timely and unambiguous fashion

* attribute them to an authoritative issuer

* associate the statements in a meaningful manner via a set of well-known semantic relationships

* consistently, efficiently, and homogeneously check their authenticity

SCITT provides a standardized way to:

* know the various sources of statements

* express the provenance and historicity of statements

* relate and link various heterogeneous statements in a simple fashion

* check that the statement comes from a source with authority to issue that statement

* confirm that sources provide a complete history of statements related to a given component

2.2.2. Promotion of a Software Component by Multiple Entities

A software component (e.g., a library or software product) released by a trusted producer is common practice for both open-source and commercial offerings. The released software component is accompanied by a statement of authenticity. Over time, due to its enhanced applicability to various products, there has been an increasing number of multiple providers of the same software component version on the internet.

Some providers include this particular software component as part of their release product/package bundle and provide the package with proof of authenticity using their issuer authority. Some packages include the original statement of authenticity, and some do not. Over time, some providers no longer offer the exact same software component source code but pre-compiled software component binaries. Some sources do not provide the exact same software component, but include patches and fixes produced by third-parties, as these emerge faster than solutions from the original producer. Due to complex distribution and promotion life-cycle scenarios, the original software component takes myriad forms.

A consumer of a released software wants to:

* understand if a particular provider is a trusted originating producer or an alternative party

* know if and how the source, or resulting binary, of a promoted software component differs from the original software component

* check the provenance and history of a software component’s source back to its origin

* assess whether to trust a component or product based on a downloaded package location and source supplier

SCITT provides a standardized way to:

* reliably discern if a provider is the original, trusted producer or is a trustworthy alternative provider or is an illegitimate provider

* track the provenance path from an original producer to a particular provider

* check the trustworthiness of a provider

* check the integrity of modifications or transformations applied by a provider

2.2.3. Software Integrator Assembling a Software Product for an Autonomous Vehicle

Software Integration is a complex activity. This typically involves getting various software components from multiple suppliers, producing an integrated package deployed as part of device assembly. For example, car manufacturers source integrated software for their autonomous vehicles from third parties that integrate software components from various sources. Integration complexity creates a higher risk of security vulnerabilities to the delivered software.

Consumers of integrated software want:

* all components presents in a software product listed

* the ability to identify and retrieve all components from a secure and tamper-proof location

* to receive an alert when a vulnerability scan detects a known security issue on a running software component

* verifiable proofs on build process and build environment with all supplier tiers to ensure end to end build quality and security

SCITT provides a standardized way to:

* provide a tiered and transparent framework that allows for verification of integrity and authenticity of the integrated software at both component and product level before installation

* notify software integrators of vulnerabilities identified during security scans of running software

* provide valid annotations on build integrity to ensure conformance

3. Terminology

The terms defined in this section have special meaning in the context of Supply Chain Integrity, Transparency, and Trust, which are used throughout this document. When used in text, the corresponding terms are capitalized. To ensure readability, only a core set of terms is included in this section.

The terms "header", "payload", and "to-be-signed bytes" are defined in [STD96].

The term "claim" is defined in [RFC8392].

Statement Sequence: a sequence of Signed Statements captured by a Verifiable Data Structure. see Verifiable Data Structure

Append-only Log: a Statement Sequence comprising the entire registration history of the Transparency Service. To make the Append-only property verifiable and transparent, the Transparency Service defines how Signed Statements are made available to Auditors.

Artifact: a physical or non-physical item that is moving along a supply chain.

Auditor: an entity that checks the correctness and consistency of all Transparent Statements, or the transparent Statement Sequence, issued by a Transparency Service. An Auditor is an example of a specialized Relying Party.

Client: an application making protected Transparency Service resource requests on behalf of the resource owner and with its authorization.

Envelope: metadata, created by the Issuer to produce a Signed Statement. The Envelope contains the identity of the Issuer and information about the Artifact, enabling Transparency Service Registration Policies to validate the Signed Statement. A Signed Statement is a COSE Envelope wrapped around a Statement, binding the metadata in the Envelope to the Statement. In COSE, an Envelope consists of a protected header (included in the Issuer’s signature) and an unprotected header (not included in the Issuer’s signature).

Equivocation: a state where a Transparency Service provides inconsistent proofs to Relying Parties, containing conflicting claims about the Signed Statement bound at a given position in the Verifiable Data Structure [EQUIVOCATION].

Issuer: an identifier representing an organization, device, user, or entity securing Statements about supply chain Artifacts. An Issuer may be the owner or author of Artifacts, or an independent third party such as an Auditor, reviewer or an endorser. In SCITT Statements and Receipts, the iss CWT Claim is a member of the COSE header parameter 15: CWT_Claims within the protected header of a COSE Envelope.

Non-equivocation: a state where all proofs provided by the Transparency Service to Relying Parties are produced from a Single Verifiable Data Structure describing a unique sequence of Signed Statements and are therefore consistent. Over time, an Issuer may register new Signed Statements about an Artifact in a Transparency Service with new information. However, the consistency of a collection of Signed Statements about the Artifact can be checked by all Relying Parties.

Receipt: a cryptographic proof that a Signed Statement is included in the Verifiable Data Structure. See [I-D.draft-ietf-cose-merkle-tree-proofs] for implementations Receipts are signed proofs of verifiable data-structure properties. The types of Receipts MUST support inclusion proofs and MAY support other proof types, such as consistency proofs.

Registration: the process of submitting a Signed Statement to a Transparency Service, applying the Transparency Service's Registration Policy, adding to the Verifiable Data Structure, and producing a Receipt.

Registration Policy: the pre-condition enforced by the Transparency Service before registering a Signed Statement, based on information in the non-opaque header and metadata contained in its COSE Envelope.

Relying Party: a Relying Parties consumes Transparent Statements, verifying their proofs and inspecting the Statement payload, either before using corresponding Artifacts, or later to audit an Artifact's provenance on the supply chain.

Signed Statement: an identifiable and non-repudiable Statement about an Artifact signed by an Issuer. In SCITT, Signed Statements are encoded as COSE signed objects; the payload of the COSE structure contains the issued Statement.

Statement: any serializable information about an Artifact. To help interpretation of Statements, they must be tagged with a media type (as specified in [RFC6838]). A Statement may represent a Software Bill Of Materials (SBOM) that lists the ingredients of a software Artifact, an endorsement or attestation about an Artifact, indicate the End of Life (EOL), redirection to a newer version, or any content an Issuer wishes to publish about an Artifact. Additional Statements about an Artifact are correlated by the Subject Claim as defined in the IANA CWT [IANA.cwt] registry and used as a protected header parameter as defined in [RFC9597]. The Statement is considered opaque to Transparency Service, and MAY be encrypted.

Subject: an identifier, defined by the Issuer, which represents the organization, device, user, entity, or Artifact about which Statements (and Receipts) are made and by which a logical collection of Statements can be grouped. It is possible that there are multiple Statements about the same Artifact. In these cases, distinct Issuers (iss) might agree to use the sub CWT Claim to create a coherent sequence of Signed Statements about the same Artifact and Relying Parties can leverage sub to ensure completeness and Non-equivocation across Statements by identifying all Transparent Statements associated to a specific Subject.

Transparency Service: an entity that maintains and extends the Verifiable Data Structure and endorses its state. The identity of a Transparency Service is captured by a public key that must be known by Relying Parties in order to validate Receipts.

Transparent Statement: a Signed Statement that is augmented with a Receipt created via Registration in a Transparency Service. The Receipt is stored in the unprotected header of COSE Envelope of the Signed Statement. A Transparent Statement remains a valid Signed Statement and may be registered again in a different Transparency Service.

Verifiable Data Structure: a data structure which supports one or more proof types, such as "inclusion proofs" or "consistency proofs", for Signed Statements as they are Registered to a Transparency Service. SCITT supports multiple Verifiable Data Structures and Receipt formats as defined in [I-D.draft-ietf-cose-merkle-tree-proofs], accommodating different Transparency Service implementations.

4. Definition of Transparency

In this document, the definition of transparency is intended to build over abstract notions of Append-only Logs and Receipts. Existing transparency systems such as Certificate Transparency are instances of this definition. SCITT supports multiple Verifiable Data Structures, as defined in [I-D.draft-ietf-cose-merkle-tree-proofs].

A Signed Statement is an identifiable and non-repudiable Statement made by an Issuer. The Issuer selects additional metadata and attaches a proof of endorsement (in most cases, a signature) using the identity key of the Issuer that binds the Statement and its metadata. Signed Statements can be made transparent by attaching a proof of Registration by a Transparency Service, in the form of a Receipt. Receipts demonstrate inclusion of Signed Statements in the Verifiable Data Structure of a Transparency Service. By extension, the Signed Statement may say an Artifact (for example, a firmware

binary) is transparent if it comes with one or more Transparent Statements from its author or owner, though the context should make it clear what type of Signed Statements is expected for a given Artifact.

Transparency does not prevent dishonest or compromised Issuers, but it holds them accountable. Any Artifact that may be verified, is subject to scrutiny and auditing by other parties. The Transparency Service provides a history of Statements, which may be made by multiple Issuers, enabling Relying Parties to make informed decisions.

Transparency is implemented by providing a consistent, append-only, cryptographically verifiable, publicly available record of entries. A SCITT instance is referred to as a Transparency Service. Implementations of Transparency Services may protect their registered sequence of Signed Statements and Verifiable Data Structure using a combination of trusted hardware, consensus protocols, and cryptographic evidence. A Receipt is a signature over one or more Verifiable Data Structure Proofs that a Signed Statement is registered in the Verifiable Data Structure. It is universally verifiable without online access to the TS. Requesting a Receipt can result in the production of a new Receipt for the same Signed Statement. A Receipt’s verification key, signing algorithm, validity period, header parameters or other claims MAY change each time a Receipt is produced.

Anyone with access to the Transparency Service can independently verify its consistency and review the complete list of Transparent Statements registered by each Issuer.

Reputable Issuers are thus incentivized to carefully review their Statements before signing them to produce Signed Statements. Similarly, reputable Transparency Services are incentivized to secure their Verifiable Data Structure, as any inconsistency can easily be pinpointed by any Auditor with read access to the Transparency Service.

The building blocks defined in SCITT are intended to support applications in any supply chain that produces or relies upon digital Artifacts, from the build and supply of software and IoT devices to advanced manufacturing and food supply.

SCITT is a generalization of Certificate Transparency (CT) [RFC9162], which can be interpreted as a transparency architecture for the supply chain of X.509 certificates. Considering CT in terms of SCITT:

* CAs (Issuers) sign the ASN.1 DER encoded tbsCertificate structure to produce an X.509 certificate (Signed Statements)

* CAs submit the certificates to one or more CT logs (Transparency Services)

* CT logs produce Signed Certificate Timestamps (Transparent Statements)

* Signed Certificate Timestamps, Signed Tree Heads, and their respective consistency proofs are checked by Relying Parties

* The Verifiable Data Structure can be checked by Auditors

5. Architecture Overview

The SCITT architecture enables a loose federation of Transparency Services, by providing a set of common formats and protocols for issuing and registering Signed Statements and auditing Transparent Statements.

In order to accommodate as many Transparency Service implementations as possible, this document only specifies the format of Signed Statements (which must be used by all Issuers) and a very thin wrapper format for Receipts, which specifies the Transparency Service identity and the agility parameters for the Signed Inclusion Proofs. The remaining details of the Receipt’s contents are specified in [I-D.draft-ietf-cose-merkle-tree-proofs].

Figure 2 illustrates the roles and processes that comprise a Transparency Service independent of any one use case:

* Issuers that use their credentials to create Signed Statements about Artifacts

* Transparency Services that evaluate Signed Statements against Registration Policies, producing Receipts upon successful Registration. The returned Receipt may be combined with the Signed Statement to create a Transparent Statement.

* Relying Parties that:

  - collect Receipts of Signed Statements for subsequent registration of Transparent Statements;

  - retrieve Transparent Statements for analysis of Statements about Artifacts themselves (e.g. verification);

- or replay all the Transparent Statements to check for the consistency and correctness of the Transparency Service's Verifiable Data Structure (e.g. auditing)

In addition, Figure 2 illustrates multiple Transparency Services and multiple Receipts as a single Signed Statement MAY be registered with one or more Transparency Service. Each Transparency Service produces a Receipt, which may be aggregated in a single Transparent Statement, demonstrating the Signed Statement was registered by multiple Transparency Services.

The arrows indicate the flow of information.

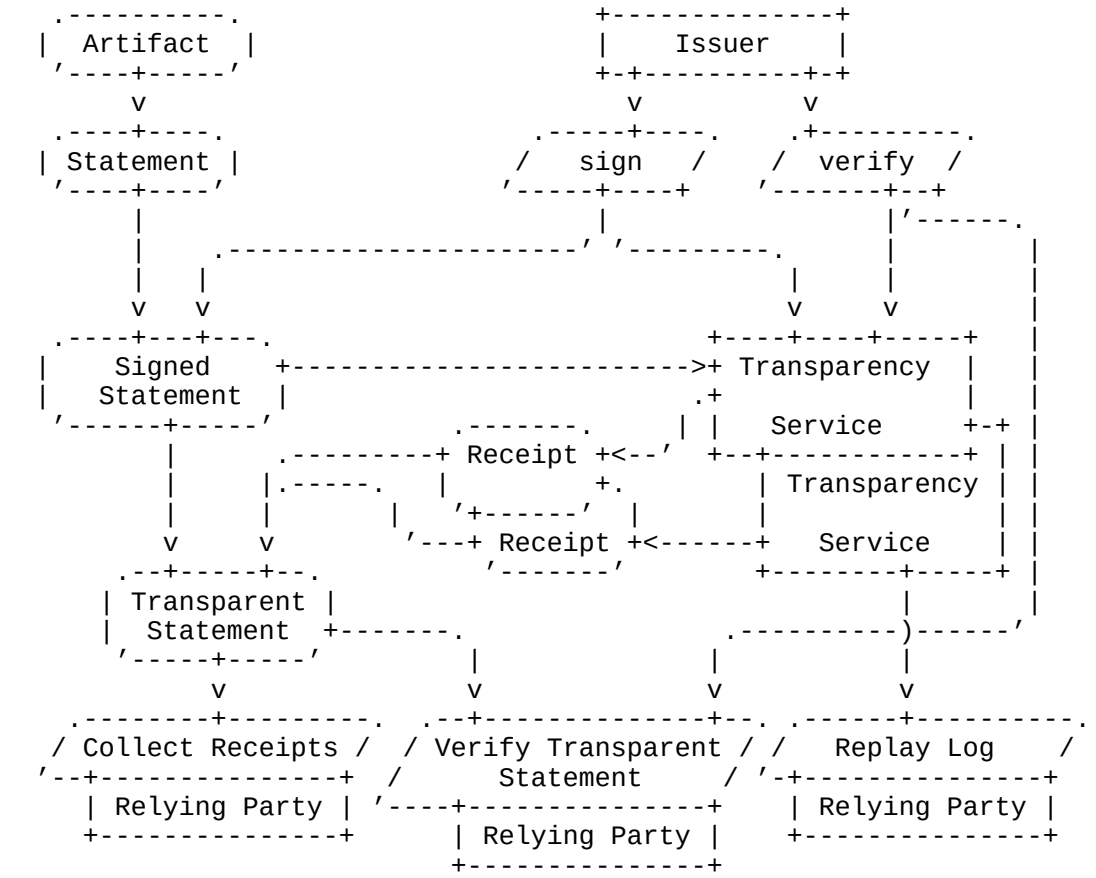


Figure 2: Relationship of Concepts in SCITT

The subsequent sections describe the main concepts, namely Transparency Service, Signed Statements, Registration, and Transparent Statements in more detail.

5.1. Transparency Service

Transparency Services MUST feature a Verifiable Data Structure. The Verifiable Data Structure records registered Signed Statements and supports the production of Receipts.

Typically a Transparency Service has a single Issuer identity which is present in the iss Claim of Receipts for that service.

Multi-tenant support can be enabled through the use of identifiers in the iss Claim, for example, ts.example may have a distinct Issuer identity for each sub domain, such as customer1.ts.example and customer2.ts.example.

5.1.1. Registration Policies

Registration Policies refer to additional checks over and above the Mandatory Registration Checks that are performed before a Signed Statement is registered to the Verifiable Data Structure. To enable audit-ability, Transparency Services MUST maintain Registration Policies.

Beyond the mandatory Registration checks, the scope of additional checks, including no additional checks, is up to the implementation.

This specification leaves implementation, encoding and documentation of Registration Policies and trust anchors to the operator of the Transparency Service.

5.1.1.1. Mandatory Registration Checks

During Registration, a Transparency Service MUST, at a minimum, syntactically check the Issuer of the Signed Statement by cryptographically verifying the COSE signature according to [STD96]. The Issuer identity MUST be bound to the Signed Statement by including an identifier in the protected header. If the protected header includes multiple identifiers, all those that are registered by the Transparency Service MUST be checked.

Transparency Services MUST maintain a list of trust anchors (see definition of trust anchor in [RFC4949]) in order to check the signatures of Signed Statements, either separately, or inside Registration Policies. Transparency Services MUST authenticate Signed Statements as part of a Registration Policy. For instance, a

trust anchor could be an X.509 root certificate (directly or its thumbprint), a pointer to an OpenID Connect identity provider, or any other COSE-compatible trust anchor.

When using X.509 Signed Statements, the Transparency Service MUST build and validate a complete certification path from an Issuer's certificate to one of the root certificates currently registered as a trust anchor by the Transparency Service. The protected header of the COSE_Sign1 Envelope MUST include either the Issuer's certificate as x5t or the chain including the Issuer's certificate as x5chain. If x5t is included in the protected header, an x5chain with a leaf certificate corresponding to the x5t value MAY be included in the unprotected header.

Registration Policies and trust anchors MUST be made Transparent and available to all Relying Parties of the Transparency Service by Registering them as Signed Statements on the Verifiable Data Structure.

The Transparency Service MUST apply the Registration Policy that was most recently committed to the Verifiable Data Structure at the time of Registration.

5.1.1.2. Auditability of Registration

The operator of a Transparency Service MAY update the Registration Policy or the trust anchors of a Transparency Service at any time.

Transparency Services MUST ensure that for any Signed Statement they register, enough information is made available to Auditors to reproduce the Registration checks that were defined by the Registration Policies at the time of Registration.

5.1.2. Initialization and Bootstrapping

Since the mandatory Registration checks rely on having registered Signed Statements for the Registration Policy and trust anchors, Transparency Services MUST support at least one of the three following bootstrapping mechanisms:

* Pre-configured Registration Policy and trust anchors;

* Acceptance of a first Signed Statement whose payload is a valid Registration Policy, without performing Registration checks

* An out-of-band authenticated management interface

5.1.3. Verifiable Data Structure

The security properties are determined by the choice of the Verifiable Data Structure ([I-D.draft-ietf-cose-merkle-tree-proofs]) used by the Transparency Service implementation. This verifiable data structure MUST support the following security requirements:

Append-Only: a property required for a verifiable data structure to be applicable to SCITT, ensuring that the Statement Sequence cannot be modified, deleted, or reordered.

Non-equivocation: there is no fork in the registered sequence of Signed Statements accepted by the Transparency Service and committed to the Verifiable Data Structure. Everyone with access to its content sees the same ordered collection of Signed Statements and can check that it is consistent with any Receipts they have verified.

Replayability: the Verifiable Data Structure includes sufficient information to enable authorized actors with access to its content to check that each data structure representing each Signed Statement has been correctly registered.

In addition to Receipts, some verifiable data structures might support additional proof types, such as proofs of consistency, or proofs of non-inclusion.

Specific verifiable data structures, such those describes in [RFC9162] and [I-D.draft-ietf-cose-merkle-tree-proofs], and the review of their security requirements for SCITT are out of scope for this document.

5.1.4. Adjacent Services

Transparency Services can be deployed along side other database or object storage technologies. For example, a Transparency Service that supports a software package management system, might be referenced from the APIs exposed for package management. Providing an ability to request a fresh Receipt for a given software package, or to request a list of Signed Statements associated with the software package.

6. Signed Statements

This specification prioritizes conformance to [STD96] and its required and optional properties. Profiles and implementation specific choices should be used to determine admissibility of conforming messages. This specification is left intentionally open to allow implementations to make Registration restrictions that make the most sense for their operational use cases.

There are many types of Statements (such as SBOMs, malware scans, audit reports, policy definitions) that Issuers may want to turn into Signed Statements. An Issuer must first decide on a suitable format (3: payload type) to serialize the Statement payload. For a software supply chain, payloads describing the software Artifacts may include:

* [CoSWID]
* [CycloneDX]
* [in-toto]
* [SPDX-CBOR]
* [SPDX-JSON]
* [SLSA]
* [SWID]

Once all the Envelope headers are set, an Issuer MUST use a standard COSE implementation to produce an appropriately serialized Signed Statement.

Issuers can produce Signed Statements about different Artifacts under the same Identity. Issuers and Relying Parties must be able to recognize the Artifact to which the Statements pertain by looking at the Signed Statement. The iss and sub Claims, within the CWT_Claims protected header, are used to identify the Artifact the Statement pertains to. (See Subject under Section 3 Terminology.)

Issuers MAY use different signing keys (identified by kid in the protected header) for different Artifacts or sign all Signed Statements under the same key.

An Issuer can make multiple Statements about the same Artifact. For example, an Issuer can make amended Statements about the same Artifact as their view changes over time.

Multiple Issuers can make different, even conflicting Statements, about the same Artifact. Relying Parties can choose which Issuers they trust.

Multiple Issuers can make the same Statement about a single Artifact, affirming multiple Issuers agree.

Additionally, x5chain that corresponds to either x5t or kid identifying the leaf certificate in the included certification path MAY be included in the unprotected header of the COSE Envelope.

* When using x.509 certificates, support for either x5t or x5chain in the protected header is REQUIRED to implement.

* Support for kid in the protected header and x5chain in the unprotected header is OPTIONAL to implement.

When x5t or x5chain is present in the protected header, iss MUST be a string that meets URI requirements defined in [RFC8392]. The iss value's length MUST be between 1 and 8192 characters in length.

The kid header parameter MUST be present when neither x5t nor x5chain is present in the protected header. Key discovery protocols are out-of-scope of this document.

The protected header of a Signed Statement and a Receipt MUST include the CWT Claims header parameter as specified in Section 2 of [RFC9597]. The CWT Claims value MUST include the Issuer Claim (Claim label 1) and the Subject Claim (Claim label 2) [IANA.cwt].

A Receipt is a Signed Statement, (COSE_Sign1), with additional Claims in its protected header related to verifying the inclusion proof in its unprotected header. See [I-D.draft-ietf-cose-merkle-tree-proofs].

6.1. Signed Statement Examples

Figure 3 illustrates a normative CDDL definition [RFC8610] for the protected header and unprotected header of Signed Statements and Receipts.

The SCITT architecture specifies the minimal mandatory labels. Implementation-specific Registration Policies may define additional mandatory labels.

```
Signed_Statement = #6.18(COSE_Sign1)
Receipt = #6.18(COSE_Sign1)

COSE_Sign1 = [
  protected   : bstr .cbor Protected_Header,
  unprotected : Unprotected_Header,
  payload     : bstr / nil,
  signature   : bstr
]

Protected_Header = {
  &(CWT_Claims: 15) => CWT_Claims
  ? &(alg: 1) => int
  ? &(content_type: 3) => tstr / uint
  ? &(kid: 4) => bstr
  ? &(x5t: 34) => COSE_CertHash
  * int => any
}

CWT_Claims = {
  &(iss: 1) => tstr
  &(sub: 2) => tstr
  * int => any
}

Unprotected_Header = {
  ? &(x5chain: 33) => COSE_X509
  ? &(receipts: 394)  => [+ Receipt]
  * int => any
}
```

Figure 3: CDDL definition for Signed Statements and Receipts

Figure 4 illustrates an instance of a Signed Statement in Extended Diagnostic Notation (EDN), with a payload that is detached. Detached payloads support large Statements, and ensure Signed Statements can integrate with existing storage systems.

```
18(                                 / COSE Sign 1      /
    [
      h'a4012603...6d706c65',       / Protected        /
      {},                           / Unprotected      /
      nil,                          / Detached payload /
      h'79ada558...3a28bae4'        / Signature        /
    ]
)
```

Figure 4: CBOR Extended Diagnostic Notation example of a Signed Statement

Figure 5 illustrates the decoded protected header of the Signed Statement in Figure 4. It indicates the Signed Statement is securing a JSON content type, and identifying the content with the sub Claim "vendor.product.example".

```
{                                  / Protected        /
  1: -7,                           / Algorithm        /
  3: application/example+json,     / Content type     /
  4: h'50685f55...50523255',       / Key identifier   /
  15: {                            / CWT Claims       /
    1: software.vendor.example,    / Issuer           /
    2: vendor.product.example,     / Subject          /
  }
}
```

Figure 5: CBOR Extended Diagnostic Notation example of a Signed Statement's Protected Header

6.2. Registration of Signed Statements

To register a Signed Statement, the Transparency Service performs the following steps:

1. *Client authentication:* A Client authenticates with the Transparency Service before registering Signed Statements on behalf of one or more Issuers. Authentication and authorization are implementation-specific and out of scope of the SCITT architecture.

2. *TS Signed Statement Verification and Validation:* The Transparency Service MUST perform signature verification per Section 4.4 of [STD96] and MUST verify the signature of the Signed Statement with the signature algorithm and verification key of the Issuer per [RFC9360]. The Transparency Service MUST also check the Signed Statement includes the required protected headers. The Transparency Service MAY validate the Signed Statement payload in order to enforce domain specific registration policies that apply to specific content types.

3. *Apply Registration Policy:* The Transparency Service MUST check the attributes required by a Registration Policy are present in the protected headers. Custom Signed Statements are evaluated given the current Transparency Service state and the entire Envelope and may use information contained in the attributes of named policies.

4. *Register the Signed Statement*

5. *Return the Receipt*, which MAY be asynchronous from Registration. The Transparency Service MUST be able to provide a Receipt for all registered Signed Statements. Details about generating Receipts are described in Section 7.

The last two steps may be shared between a batch of Signed Statements registered in the Verifiable Data Structure.

A Transparency Service MUST ensure that a Signed Statement is registered before releasing its Receipt.

A Transparency Service MAY accept a Signed Statement with content in its unprotected header, and MAY use values from that unprotected header during verification and registration policy evaluation.

However, the unprotected header of a Signed Statement MUST be set to an empty map before the Signed Statement can be included in a Statement Sequence.

The same Signed Statement may be independently registered in multiple Transparency Services, producing multiple, independent Receipts. The multiple Receipts may be attached to the unprotected header of the Signed Statement, creating a Transparent Statement.

An Issuer that knows of a changed state of quality for an Artifact, SHOULD Register a new Signed Statement, using the same 15 CWT iss and sub Claims.

7. Transparent Statements

The Client (which is not necessarily the Issuer) that registers a Signed Statement and receives a Receipt can produce a Transparent Statement by adding the Receipt to the unprotected header of the Signed Statement. Client applications MAY register Signed Statements on behalf of one or more Issuers. Client applications MAY request Receipts regardless of the identity of the Issuer of the associated Signed Statement.

When a Signed Statement is registered by a Transparency Service a Receipt becomes available. When a Receipt is included in a Signed Statement a Transparent Statement is produced.

Receipts are based on Signed Inclusion Proofs as described in COSE Receipts [I-D.draft-ietf-cose-merkle-tree-proofs] that also provides the COSE header parameter semantics for label 394.

The Registration time is recorded as the timestamp when the Transparency Service added the Signed Statement to its Verifiable Data Structure.

Figure 6 illustrates a normative CDDL definition of Transparent Statements. See Figure 3 for the CDDL rule that defines 'COSE_Sign1' as specified in Section 4.2 of [STD96]

```
Transparent_Statement = #6.18(COSE_Sign1)

Unprotected_Header = {
  &(receipts: 394)  => [+ Receipt]
}
```

Figure 6: CDDL definition for a Transparent Statement

Figure 7 illustrates a Transparent Statement with a detached payload, and two Receipts in its unprotected header. The type of label 394 receipts in the unprotected header is a CBOR array that can contain one or more Receipts (each entry encoded as a .cbor encoded Receipts).

```
18(                                 / COSE Sign 1              /
    [
      h'a4012603...6d706c65',       / Protected                /
      {                             / Unprotected              /
        394:   [                    / Receipts (2)             /
          h'd284586c...4191f9d2'    / Receipt 1                /
          h'c624586c...8f4af97e'    / Receipt 2                /
        ]
      },
      nil,                          / Detached payload         /
      h'79ada558...3a28bae4'        / Signature                /
    ]
)
```

Figure 7: CBOR Extended Diagnostic Notation example of a Transparent Statement

Figure 8 one of the decoded Receipt from Figure 7. The Receipt contains inclusion proofs for verifiable data structures. The unprotected header contains verifiable data structure proofs. See the protected header for details regarding the specific verifiable data structure used. Per the COSE Verifiable Data Structure Registry documented in [I-D.draft-ietf-cose-merkle-tree-proofs], the COSE key type RFC9162_SHA256 is value 1. Labels identify inclusion proofs (-1) and consistency proofs (-2).

```
18(                                 / COSE Sign 1               /
    [
      h'a4012604...6d706c65',       / Protected                 /
      {                             / Unprotected               /
        -222: {                     / Proofs                    /
          -1: [                     / Inclusion proofs (1)      /
            h'83080783...32568964', / Inclusion proof 1         /
          ]
        },
      },
      nil,                          / Detached payload          /
      h'10f6b12a...4191f9d2'        / Signature                 /
    ]
)
```

Figure 8: CBOR Extended Diagnostic Notation example of a Receipt

Figure 9 illustrates the decoded protected header of the Transparent Statement in Figure 7. The verifiable data structure (-111) uses 1 from (RFC9162_SHA256).

```
{                                   / Protected                 /
  1: -7,                            / Algorithm                 /
  4: h'50685f55...50523255',        / Key identifier            /
  -111: 1,                          / Verifiable Data Structure /
  15: {                             / CWT Claims                /
    1: transparency.vendor.example, / Issuer                    /
    2: vendor.product.example,      / Subject                   /
  }
}
```

Figure 9: CBOR Extended Diagnostic Notation example of a Receipt's Protected Header

Figure 10 illustrates the decoded inclusion proof from Figure 8. This inclusion proof indicates that the size of the Verifiable Data Structure was 8 at the time the Receipt was issued. The structure of this inclusion proof is specific to the verifiable data structure used (RFC9162_SHA256).

```
[                                   / Inclusion proof 1         /
  8,                                / Tree size                 /
  7,                                / Leaf index                /
  [                                 / Inclusion hashes (3)      /
     h'c561d333...f9850597'         / Intermediate hash 1       /
     h'75f177fd...2e73a8ab'         / Intermediate hash 2       /
     h'0bdaaed3...32568964'         / Intermediate hash 3       /
  ]
]
```

Figure 10: CBOR Extended Diagnostic Notation example of a Receipt's Inclusion Proof

7.1. Validation

Relying Parties MUST apply the verification process as described in Section 4.4 of [STD96], when checking the signature of Signed Statements and Receipts.

A Relying Party MUST trust the verification key or certificate and the associated identity of at least one Issuer of a Receipt.

A Relying Party MAY decide to verify only a single Receipt that is acceptable to them and not check the signature on the Signed Statement or Receipts which rely on verifiable data structures which they do not understand.

APIs exposing verification logic for Transparent Statements may provide more details than a single boolean result. For example, an API may indicate if the signature on the Receipt or Signed Statement is valid, if Claims related to the validity period are valid, or if the inclusion proof in the Receipt is valid.

Relying Parties MAY be configured to re-verify the Issuer's Signed Statement locally.

In addition, Relying Parties MAY apply arbitrary validation policies after the Transparent Statement has been verified and validated. Such policies may use as input all information in the Envelope, the Receipt, and the Statement payload, as well as any local state.

8. Privacy Considerations

Interactions with Transparency Services are expected to use appropriately strong encryption and authorization technologies.

The Transparency Service is trusted with the confidentiality of the Signed Statements presented for Registration. Issuers and Clients are responsible for verifying that the Transparency Service's privacy and security posture is suitable for the contents of the Signed Statements they submit prior to Registration. Issuers must carefully review the inclusion of private, confidential, or personally identifiable information (PII) in their Statements against the Transparency Service's privacy posture.

In some deployments a special role such as an Auditor might require and be given access to both the Transparency Service and related Adjacent Services.

Transparency Services can leverage Verifiable Data Structures which only retain cryptographic metadata (e.g. a hash), rather than the complete Signed Statement, as part of a defense in depth approach to maintaining confidentiality. By analyzing the relationship between data stored in the Transparency Service and data stored in Adjacent Services, it is possible to perform metadata analysis, which could reveal the order in which artifacts were built, signed, and uploaded.

9. Security Considerations

SCITT provides the following security guarantees:

1. Statements made by Issuers about supply chain Artifacts are identifiable and can be authenticated

2. Statement provenance and history can be independently and consistently audited

3. Issuers can efficiently prove that their Statement is logged by a Transparency Service

The first guarantee is achieved by requiring Issuers to sign their Statements. The second guarantee is achieved by proving a Signed Statement is present in a Verifiable Data Structure. The third guarantee is achieved by the combination of both of these steps.

9.1. Ordering of Signed Statements

Statements are signed prior to submitting to a SCITT Transparency service. Unless advertised in the Transparency Service Registration Policy, the Relying Party cannot assume that the ordering of Signed Statements in the Verifiable Data Structure matches the ordering of their issuance.

9.2. Accuracy of Statements

Issuers can make false Statements either intentionally or unintentionally, registering a Statement only proves it was produced by an Issuer. A registered Statement may be superseded by a subsequently submitted Signed Statement from the same Issuer, with the same subject in the cwt_claims protected header. Other Issuers may make new Statements to reflect new or corrected information. Relying Parties may choose to include or exclude Statements from Issuers to determine the accuracy of a collection of Statements.

9.3. Key Management

Issuers and Transparency Services MUST:

* carefully protect their private signing keys
* avoid using keys for more than one purpose
* rotate their keys in well-defined cryptoperiods, see [KEY-MANAGEMENT]

9.3.1. Verifiable Data Structure

The security considerations for specific Verifiable Data Structures are out of scope for this document. See [I-D.draft-ietf-cose-merkle-tree-proofs] for the generic security considerations that apply to Verifiable Data Structure and Receipts.

9.4. Threat Model

This section provides a generic threat model for SCITT, describing its residual security properties when some of its actors (Issuers, Transparency Services, and Auditors) are either corrupt or compromised.

SCITT primarily supports checking of Signed Statement authenticity, both from the Issuer (authentication) and from the Transparency Service (transparency). Issuers and Transparency Services can both be compromised.

The SCITT Architecture does not require trust in a single centralized Transparency Service. Different actors may rely on different Transparency Services, each registering a subset of Signed Statements subject to their own policy. Running multiple, independent Transparency Services provides different organizations to represent consistent or divergent opinions. It is the role of the relying party to decide which Transparency Services and Issuers they choose to trust for their scenario.

In both cases, the SCITT architecture provides generic, universally-verifiable cryptographic proofs to individually blame Issuers or the Transparency Service. On one hand, this enables valid actors to detect and disambiguate malicious actors who employ Equivocation with Signed Statements to different entities. On the other hand, their liability and the resulting damage to their reputation are application specific, and out of scope of the SCITT architecture.

Relying Parties and Auditors need not be trusted by other actors. So long as actors maintain proper control of their signing keys and identity infrastructure they cannot "frame" an Issuer or a Transparency Service for Signed Statements they did not issue or register.

9.4.1. Cryptographic Agility

The SCITT Architecture supports cryptographic agility. There are no mandatory to implement signing algorithms for Signed Statements or Receipts.

9.4.2. Key Compromise

Revocation strategies for compromised keys are out of scope for this document. It is important for Issuers and Transparency Services to clearly communicate when keys are compromised, so that Signed Statements can be rejected by Transparency Services or Receipts can be ignored by Relying Parties.

10. IANA Considerations

IANA is requested to register:

* the media type application/scitt-statement+cose in the "Media Types" registry, see below.

* the media type application/scitt-receipt+cose in the "Media Types" registry, see below.

10.1. COSE Receipts Header Parameter

394 is requested in [I-D.draft-ietf-cose-merkle-tree-proofs] and has received an early assignment.

10.2. Media Type application/scitt-statement+cose Registration

IANA is requested to add the following Media-Type to the "Media Types" registry [IANA.media-types].

Name	Template	Reference
scitt-statement+cose	application/scitt-statement+cose	Section 6 of RFCthis

Table 1: SCITT Signed Statement Media Type Registration

Type name: application
Subtype name: statement+cose
Required parameters: n/a
Optional parameters: n/a
Encoding considerations: binary (CBOR data item)
Security considerations: Section 9 of RFCthis
Interoperability considerations: none
Published specification: RFCthis
Applications that use this media type: Used to provide an identifiable and non-repudiable Statement about an Artifact signed by an Issuer.
Fragment identifier considerations: n/a
Additional information: Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): .scitt

Macintosh file type code(s): N/A
Person and email address to contact for further information: iesg@ietf.org
Intended usage: COMMON
Restrictions on usage: none
Author/Change controller: IETF

10.3. Media Type application/scitt-receipt+cose Registration

Name	Template	Reference
scitt-receipt+cose	application/scitt-receipt+cose	Section 7 of RFCthis

Table 2: SCITT Receipt Media Type Registration

Type name: application
Subtype name: receipt+cose
Required parameters: n/a
Optional parameters: n/a
Encoding considerations: binary (CBOR data item)
Security considerations: Section 9 of RFCthis
Interoperability considerations: none
Published specification: RFCthis
Applications that use this media type: Used to establish or verify transparency over Statements. Typically emitted by a Transparency Service, for the benefit of Relying Parties wanting to ensure Non-equivocation over all or part of a Statement Sequence.
Fragment identifier considerations: n/a
Additional information: Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): .receipt

Macintosh file type code(s): N/A
Person and email address to contact for further information: iesg@ietf.org
Intended usage: COMMON
Restrictions on usage: none
Author/Change controller: IETF

10.4. CoAP Content-Format Registrations

IANA is requested to register the following Content-Format numbers in the "CoAP Content-Formats" sub-registry, within the "Constrained RESTful Environments (CoRE) Parameters" Registry [IANA.core-parameters] in the 0-255 Range:

Content-Type	Content Coding	ID	Reference
application/scitt-statement+cose	-	103	RFCthis
application/scitt-receipt+cose	-	104	RFCthis

Table 3: SCITT Content-Formats Registration

11. Common Terminology Disambiguation

This document has been developed in coordination with the COSE, OAUTH and RATS WG and uses terminology common to these working groups.

This document uses the terms "Issuer", and "Subject" as described in [RFC8392], however the usage is consistent with the broader interpretation of these terms in both JOSE and COSE, and the guidance in [RFC8725] generally applies the COSE equivalent terms with consistent semantics.

The terms "verifier" and "Relying Party" are used interchangeably through the document. While these terms are related to "Verifier" and "Relying Party" as used in [RFC9334], they do not imply the processing of RATS conceptual messages, such as Evidence or Attestation Results that are specific to remote attestation. A SCITT "verifier" and "Relying Party" and "Issuer" of Receipts or Statements might take on the role of a RATS "Attester". Correspondingly, all RATS conceptual messages, such as Evidence and Attestation Results, can be the content of SCITT Statements and a SCITT "verifier" can also take on the role of a RATS "Verifier" to, for example, conduct the procedure of Appraisal of Evidence as a part of a SCITT "verifier"’s verification capabilities.

The terms "Claim" and "Statement" are used throughout this document, where Claim is consistent with the usage in [RFC9711] and [RFC7523], and Statement is reserved for any arbitrary bytes, possibly identified with a media type, about which the Claims are made.

The term "Subject" provides an identifier of the Issuer’s choosing to refer to a given Artifact and ensures that all associated Statements can be attributed to the identifier chosen by the Issuer.

In simpler language, a SCITT Statement could be some vendor-specific software bill of materials (SBOM), results from a model checker, static analyzer, or RATS Evidence about the authenticity of an SBOM

creation process, where the Issuer identifies themselves using the iss Claim, and the specific software that was analyzed as the Subject using the sub Claim.

In [RFC7523], the Authorization Server (AS) verifies Private Key JWT client authentication requests, and issues access tokens to clients configured to use "urn:ietf:params:oauth:client-assertion-type:jwt-bearer". This means the AS initially acts as a "verifier" of the authentication credentials in form of a JWT, and then later as an "Issuer" of access and refresh tokens. This mirrors how Signed Statements are verified before Receipts are issued by a Transparency Service. Note that the use of [RFC7523] is only one possible approach for client authentication in OAuth.

[FIPS.201] defines "assertion" as "A verifiable statement from an IdP to an RP that contains information about an end user".

[NIST.SP.800-63-3] defines "assertion" as "A statement from a verifier to an RP that contains information about a subscriber. Assertions may also contain verified attributes."

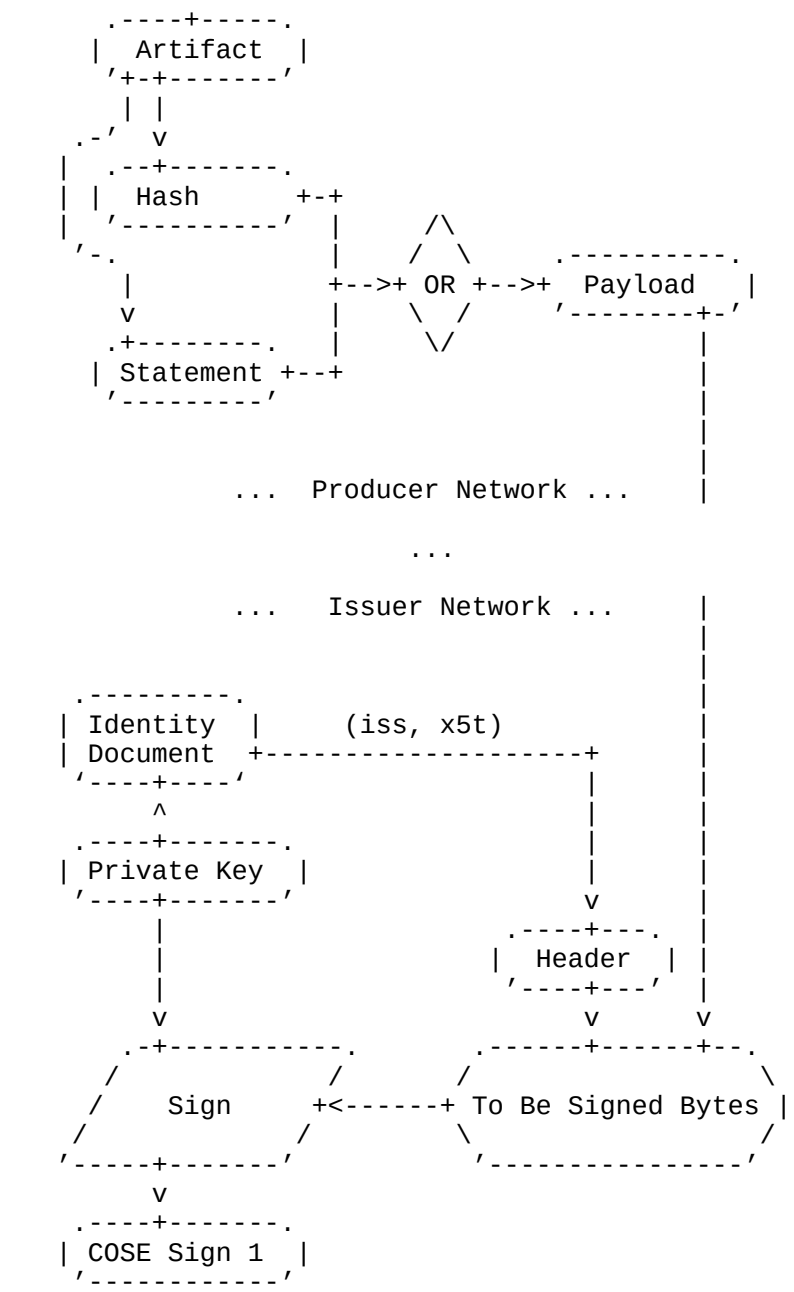
This document uses the term Statement to refer to potentially unsecured data and associated Claims, and Signed Statement and Receipt to refer to assertions from an Issuer, or the Transparency Service.

[NIST.SP.1800-19] defines "attestation" as "The process of providing a digital signature for a set of measurements securely stored in hardware, and then having the requester validate the signature and the set of measurements."

NIST guidance "Software Supply Chain Security Guidance EO 14028" uses the definition from [NIST_EO14028], which states that an "attestation" is "The issue of a statement, based on a decision, that fulfillment of specified requirements has been demonstrated.". In the RATS context, a "NIST attestation" is similar to a RATS "Endorsement". Occasionally, RATS Evidence and RATS Attestation Results or the procedures of creating these conceptual messages are referred to as "attestation" or (in cases of the use as a verb) "to attest". The stand-alone use of "attestation" and "to attest" is discouraged outside a well-defined context, such as specification text that highlights the application of terminology, explicitly. Correspondingly, it is often useful for the intended audience to qualify the term "attestation" to avoid confusion and ambiguity.

12. Signing Statements Remotely

Statements about digital Artifacts, containing digital Artifacts, or structured data regarding any type of Artifacts, can be too large or too sensitive to be send to a remote Transparency Services over the Internet. In these cases a Statement can also be hash, which becomes the payload included in COSE to-be-signed bytes. A Signed Statement (COSE_Sign1) MUST be produced from the to-be-signed bytes according to Section 4.4 of [STD96].



13. References

13.1. Normative References

[I-D.draft-ietf-cose-merkle-tree-proofs]
Steele, O., Birkholz, H., Delignat-Lavaud, A., and C. Fournet, "COSE Receipts", Work in Progress, Internet-Draft, draft-ietf-cose-merkle-tree-proofs-14, 11 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-merkle-tree-proofs-14>.

[IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <https://www.iana.org/assignments/core-parameters>.

[IANA.cwt] IANA, "CBOR Web Token (CWT) Claims", <https://www.iana.org/assignments/cwt>.

[IANA.media-types]
IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

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

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://doi.org/10.17487/RFC6838>.

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

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://doi.org/10.17487/RFC8392>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://doi.org/10.17487/RFC8610>.

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", RFC 9360, DOI 10.17487/RFC9360, February 2023, <https://doi.org/10.17487/RFC9360>.

[RFC9597] Looker, T. and M.B. Jones, "CBOR Web Token (CWT) Claims in COSE Headers", RFC 9597, DOI 10.17487/RFC9597, June 2024, <https://doi.org/10.17487/RFC9597>.

[STD94] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://doi.org/10.17487/RFC8949>.

[STD96] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://doi.org/10.17487/RFC9052>.

13.2. Informative References

[CoSWID] Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", RFC 9393, DOI 10.17487/RFC9393, June 2023, <https://doi.org/10.17487/RFC9393>.

[CycloneDX] "CycloneDX", n.d., <https://cyclonedx.org/specification/overview/>.

[EQUIVOCATION] Chun, B., Maniatis, P., Shenker, S., and J. Kubiatowicz, "Attested append-only memory: making adversaries stick to their word", Association for Computing Machinery (ACM), DOI 10.1145/1323293.1294280, ACM SIGOPS Operating Systems Review vol. 41, no. 6, pp. 189-204, October 2007, <https://doi.org/10.1145/1323293.1294280>.

[FIPS.201] "Personal identity verification (PIV) of federal employees and contractors", National Institute of Standards and Technology (U.S.), DOI 10.6028/nist.fips.201-3, January 2022, <https://doi.org/10.6028/nist.fips.201-3>.

[in-toto] "in-toto", n.d., <https://in-toto.io/>.

[KEY-MANAGEMENT] Barker, E. and W. Barker, "Recommendation for key management:: part 2 -- best practices for key management organizations", National Institute of Standards and Technology, DOI 10.6028/nist.sp.800-57pt2r1, May 2019, <https://doi.org/10.6028/nist.sp.800-57pt2r1>.

[NIST.SP.1800-19]
Bartock, M., Dodson, D., Souppaya, M., Carroll, D., Masten, R., Scinta, G., Massis, P., Prafullchandra, H., Malnar, J., Singh, H., Ghandi, R., Storey, L. E, Yeluri, R., Shea, T., Dalton, M., Weber, R., Scarfone, K., Dukes, A., Haskins, J., Phoenix, C., Swarts, B., and National Institute of Standards and Technology (U.S.), "Trusted cloud :security practice guide for VMware hybrid cloud infrastructure as a service (IaaS) environments", DOI 10.6028/NIST.SP.1800-19, NIST Special Publications (General) 1800-19, 20 April 2022, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-19.pdf>.

[NIST.SP.800-63-3]
Grassi, P. A, Garcia, M. E, Fenton, J. L, and NIST, "Digital identity guidelines: revision 3", DOI 10.6028/NIST.SP.800-63-3, NIST Special Publications (General) 800-63-3, 22 June 2017, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf>.

[NIST_EO14028]
"Software Supply Chain Security Guidance Under Executive Order (EO) 14028 Section 4e", 4 February 2022, <https://www.nist.gov/system/files/documents/2022/02/04/software-supply-chain-security-guidance-under-EO-14028-section-4e.pdf>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://doi.org/10.17487/RFC4949>.

[RFC7523] Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, <https://doi.org/10.17487/RFC7523>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://doi.org/10.17487/RFC8725>.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://doi.org/10.17487/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://doi.org/10.17487/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://doi.org/10.17487/RFC9711>.

[SLSA] "SLSA", n.d., <https://slsa.dev/>.

[SPDX-CBOR] "SPDX Specification", n.d., <https://spdx.dev/use/specifications/>.

[SPDX-JSON] "SPDX Specification", n.d., <https://spdx.dev/use/specifications/>.

[SWID] "SWID Specification", n.d., <https://csrc.nist.gov/Projects/Software-Identification-SWID/guidelines>.

Contributors

Orie Steele
Tradeverifyd
United States
Email: orie@or13.io

Orie contributed to improving the generalization of COSE building blocks and document consistency.

Amaury Chamayou
Microsoft
United Kingdom
Email: amaury.chamayou@microsoft.com

Amaury contributed elemental parts to finalize normative language on registration behavior and the single-issuer design, as well as overall document consistency

Dick Brooks
Business Cyber Guardian (TM)
United States

Email: dick@businesscyberguardian.com

Dick contributed to the software supply chain use cases.

Brian Knight
Microsoft
United States
Email: brianknight@microsoft.com

Brian contributed to the software supply chain use cases.

Robert Martin
MITRE Corporation
United States
Email: ramartin@mitre.org

Robert contributed to the software supply chain use cases.

Authors' Addresses

Henk Birkholz
Fraunhofer SIT
Rheinstrasse 75
64295 Darmstadt
Germany
Email: henk.birkholz@sit.fraunhofer.de

Antoine Delignat-Lavaud
Microsoft Research
21 Station Road
Cambridge
CB1 2FB
United Kingdom
Email: antdl@microsoft.com

Cedric Fournet
Microsoft Research
21 Station Road
Cambridge
CB1 2FB
United Kingdom
Email: fournet@microsoft.com

Yogesh Deshpande
ARM
110 Fulbourn Road
Cambridge
CB1 9NJ
United Kingdom
Email: yogesh.deshpande@arm.com

Steve Lasker
Email: stevenlasker@hotmail.com