# IETF122 - Bangkok
# STI-CT

**draft-ietf-wendt-stir-certificate-transparency-05**

**Chris Wendt, Rob Sliwa, Alec Fenichel, Vinit Anil Gaikwad**

# Understanding STI-CT: Using Certificate Transparency to protect the STI ecosystem

- **STI-CT** (Secure Telephony Identity Certificate Transparency) is an adaptation of the Certificate Transparency (CT) approach based on RFC6962 to the STI ecosystem.

- The primary focus is to protect the mis-issuance or spoofing of **SPC** (Service Provider Codes) or with **Delegate Certificates** that contain **Telephone Numbers** as part of the **TNAuthList** in an STI Certificate (RFC8226).

- STIR certificate eco-systems generally follow **prescriptive flows** and STIR has PASSporTs which are tokens that specifically reference certificates with a  mechanism to **deliver SCT within the certificate itself** (like a receipt of submitting to a CT log).

# Why STI-CT Improves Trust for SPC certs and TN level Delegate Certificates

In STIR, a service provider code certificate or a TN-based delegate certificate grants an entity the right to sign calls using a specific SPC or telephone number or range (via TNAuthList extension). When you rely on these certificates to authenticate signer identity, you must trust that:

- No one else can get a duplicate delegate certificate for the same code or number from another CA/SCA.
- No fraudulent or misconfigured certificate has been issued for the SPC or number without detection.

STI-CT solves these concerns in the following ways:

- All newly issued certificates are logged in a transparent, immutable Merkle Tree. For STI eco-system this would likely be public to authorized participants only.
- Service Provider or RespOrg participants or authorized participants on the behalf of interested Calling or Called party participants could watch the log to see if SPC or telephone numbers or ranges show up in unexpected certificates.
- Auditors can additionally verify the integrity of the log.

# Pre-Certificates

RFC6962 allows for submission to Certificate Transparency logs via either Certificate or Pre-Certificate.

STI-CT proposes to use Pre-Certificates only.

**What Is a Pre-Certificate?**

- A pre-certificate is an X.509 certificate body, TBSCertificate (To Be Signed Certificate) that includes a special CT poison extension that makes it invalid as a standalone end-entity certificate. Only after the pre-certificate is logged does the SCA remove or replace that poison extension, embed SCT, and finalize the certificate.

**Why do we only allow pre-certificates?**

- The use of pre-certificates mechanism allows for the inclusion of the SCT into the final certificate. Because existing STI ecosystems already have a well-defined mechanism for certificate distribution and validation and we want to maintain consistency with that. Requiring that an SCT be shared outside of the certificate would be hard to integrate into a mature ecosystem.

# How the Transparency Service Log Works

- In Certificate Transparency, the log is an **append-only** data structure. Whenever a new certificate is added, the log records in a **Merkle Tree**. Participants who have access to the log can request the log state, which is the root hash of the tree also referred to as STH (Signed Tree Head). The STH is signed by the log operator, and it serves as a commitment to the contents of the log.
- The whole premise is that the log is **immutable** and can be **trusted to not be modified**. This trust is verified by the **monitoring** and **auditing** of the log contents.

# STI-CT Participants

**Submitting STI-CA or STI-SCAs**

- These entities submit pre-certificates to the log as described in the APIs section. The returned Signed  Certificate Timestamp (SCT) can then be used to construct the final STI certificate, which includes one or more SCTs.

**Verification Services**

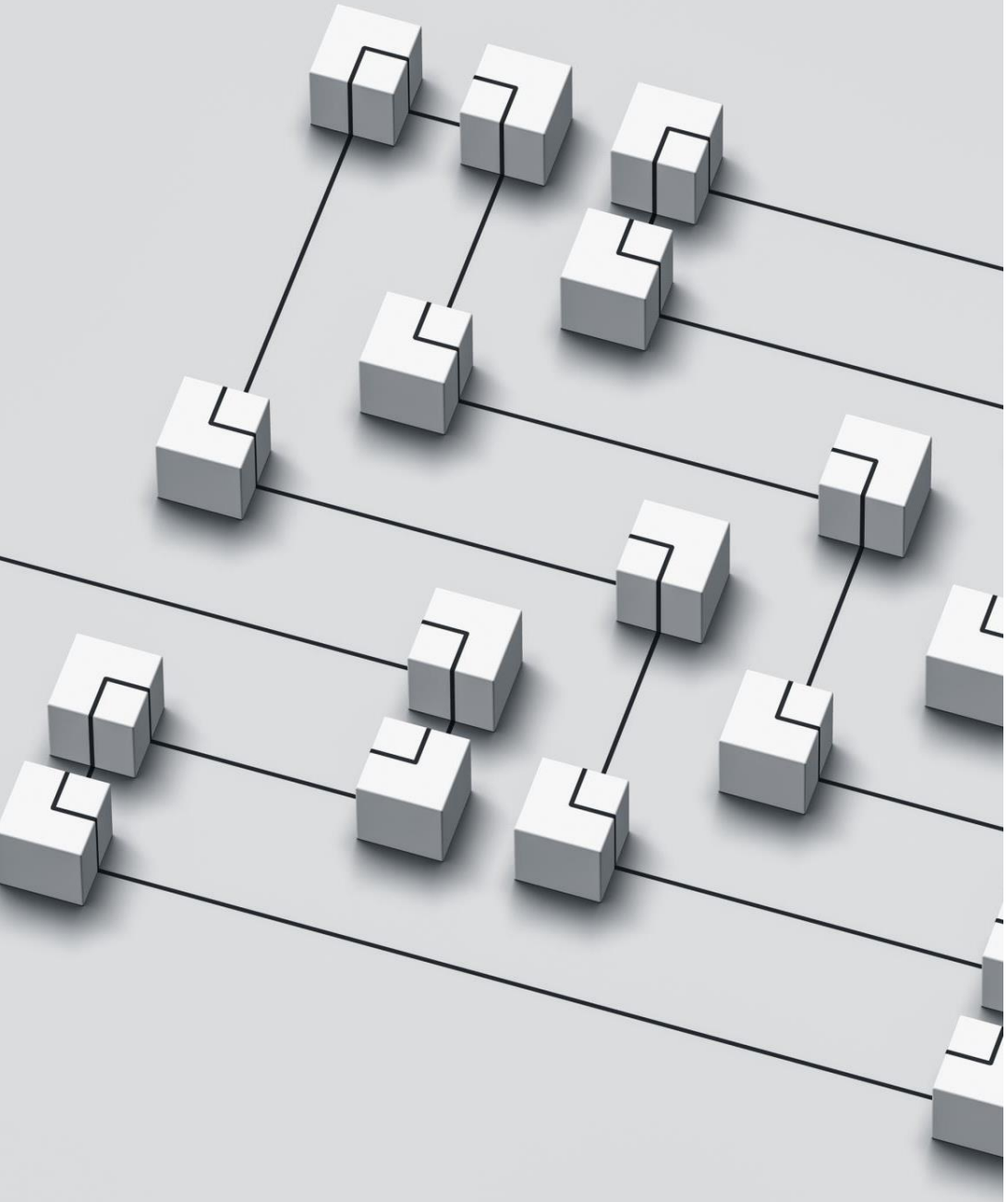
- The Verification Service receives the signed PASSporT token and verifies that the included SCTs in the certificate used to sign the PASSporT. VS should reject Certificates that do not have valid SCT(s) and fail PASSporT validation.

**Monitors**

- Monitors in the STI-CT framework play a crucial role in maintaining the integrity and trust of the ecosystem. They ensure that no certificates are mis-issued, particularly concerning the TNAuthList field, which lists the telephone numbers an entity is authorized to use.

**Auditors**

- Auditors are responsible for verifying the consistency and correctness of the log, ensuring that the log behaves according to the expected protocol. Auditors can operate as standalone services or as part of another client role, such as a monitor or an VS.



# STI-CT APIs Overview

- Provides interfaces for interacting with transparency logs for STIR ecosystem certificates.
- Operations include submission, verification, and audit functionalities for pre-certificate chains.
- Designed to ensure transparency and public auditability.

# STI-CT API – CA/SCA Submission

**Submit Pre-Certificate Chain**

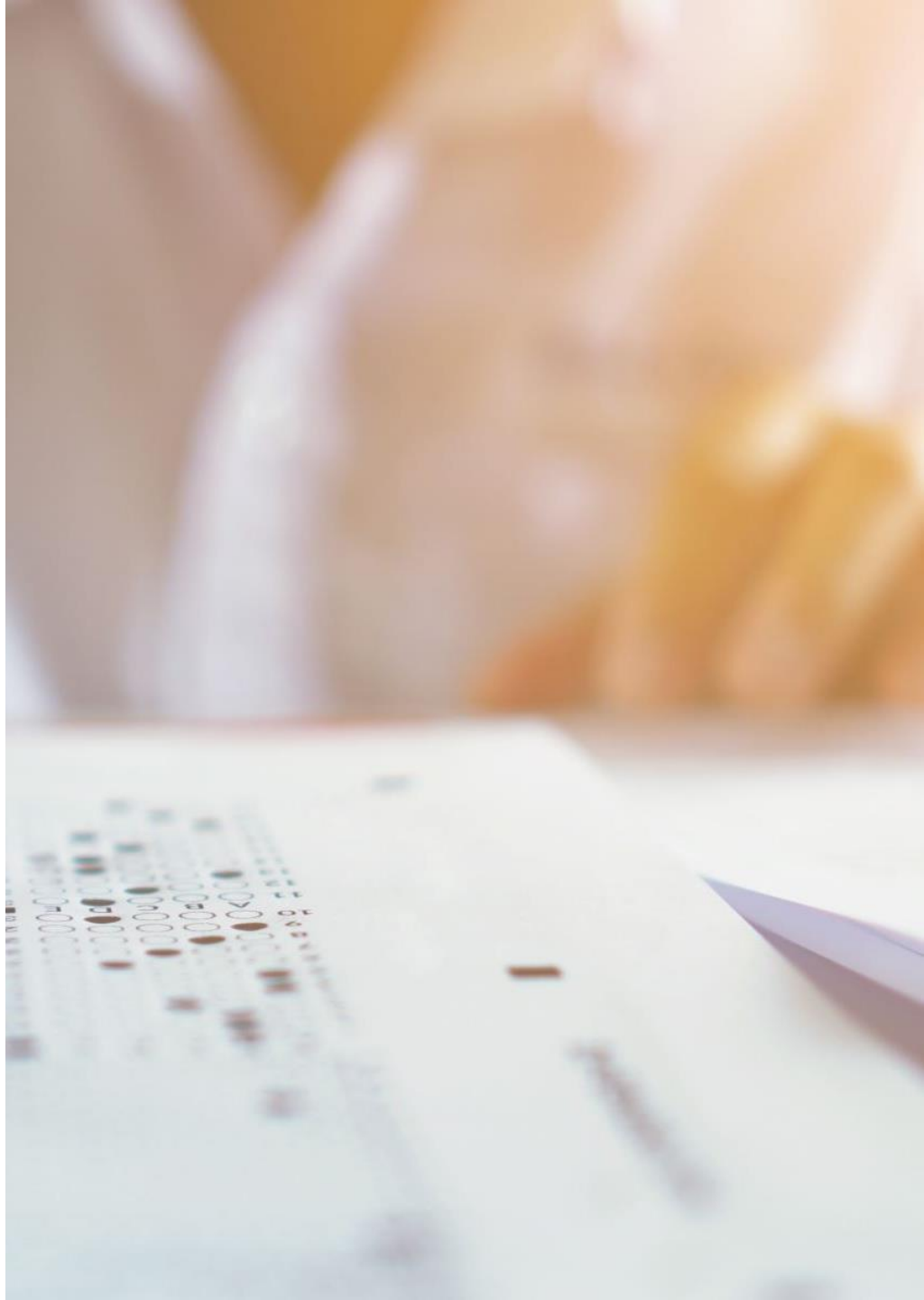
`POST /stict/v1/add-pre-chain`

**Purpose:** Submit STI pre-certificate chain to transparency logs.

**Returns:** Signed Certificate Timestamp (SCT) confirming log acceptance.

**Key Fields:**

- Request: Certificate chain (may exclude root).
- Response: SCT version, ID, timestamp, signature for inclusion in final STI certificate.



# STI-CT API – Get Log Status & Verification

### Get Merkle Tree Status

`GET /stict/v1/get-sth`

Provides log status (tree size, timestamp, root hash).

Used for verifying log integrity and monitoring growth.

### Get Accepted Root Certificates

`GET /stict/v1/get-roots`

Returns trusted root certificates for validating chains submitted to log.

# STI-CT API – Audit APIs for Transparency

**Get Consistency Proof**

`GET /stict/v1/get-sth-consistency`

Confirms consistency between two log states.

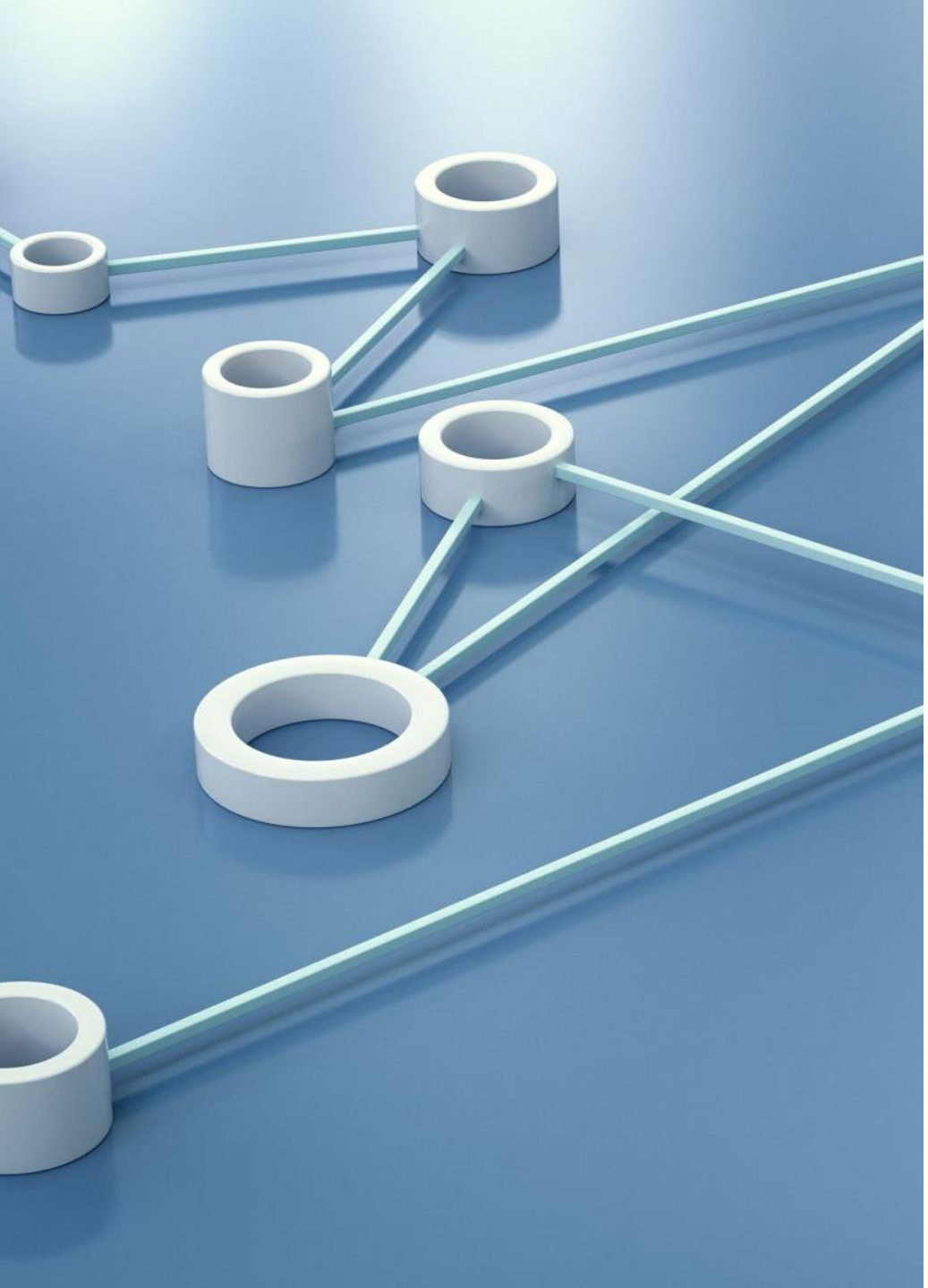
Demonstrates log’s append-only nature.

**Get Audit Proof by Leaf Hash**

`GET /stict/v1/get-proof-by-hash`

Verifies leaf inclusion within log by hash and tree size.

Includes an audit path for verification.



# STI-CT API – Retrieve Entries and Proofs for Monitoring

### Get Entry and Proof

`GET /stict/v1/get-entry-and-proof`

Fetches a single entry and proof by leaf index.

Simplifies direct log entry verification.

### Get Log Entries

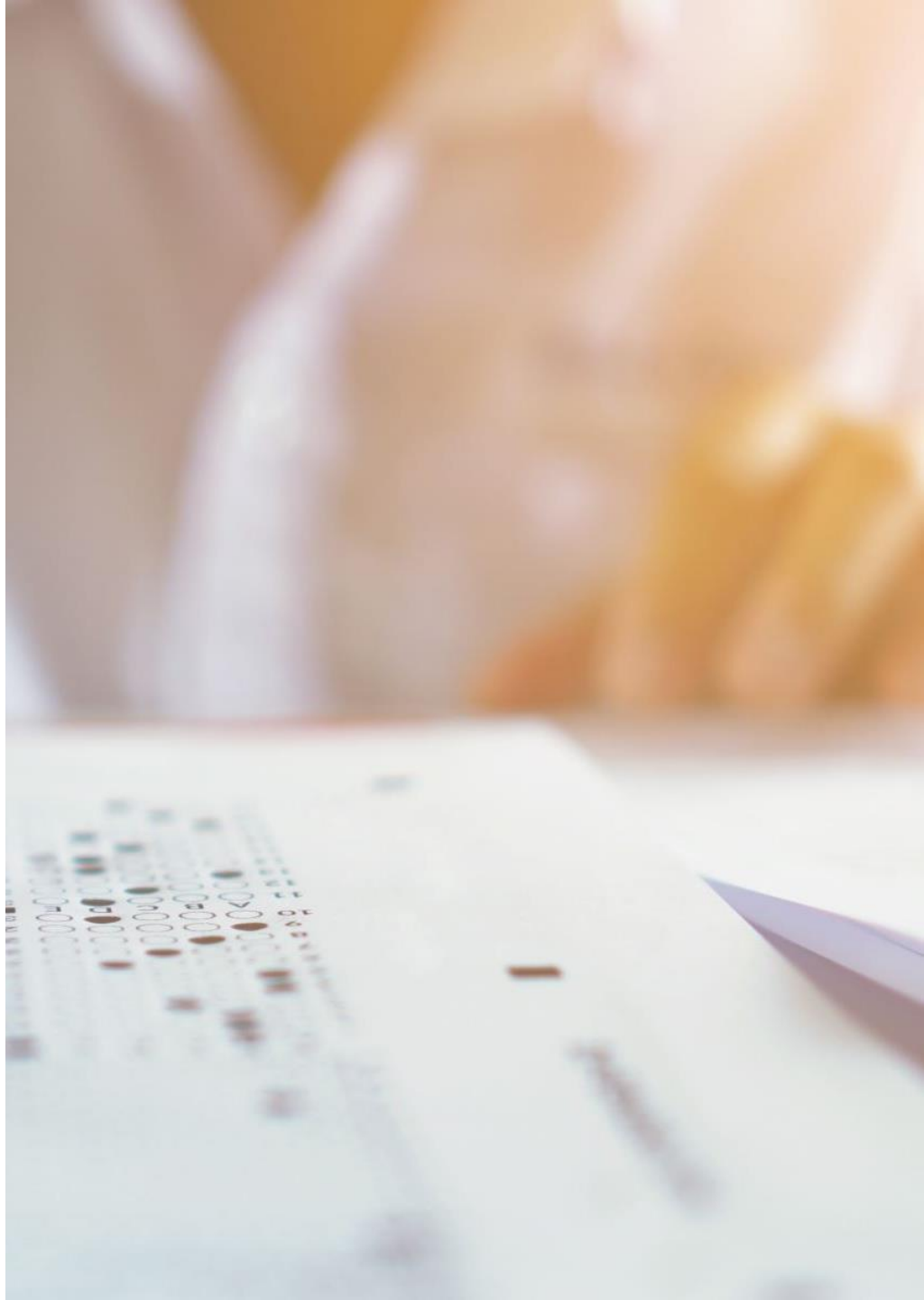
`GET /stict/v1/get-entries`

Retrieves specific log entries by range.

Useful for auditing and inspecting specific log segments.

# Draft Changes since IETF121

- Added Examples
- Pending change to remove role of Authentication Service participant



# Conclusion