Public Notary Transparency
Internet Draft
Intended status: Standards Track
Expires: July 2016

S. Kent
D. Mandelberg
K. Seo
BBN Technologies

January 30, 2016

Certificate Transparency (CT) System Architecture
draft-kent-trans-architecture-02.txt

Abstract

This document describes the architecture for Certificate Transparency (CT) focusing on the Web PKI context. It defines the goals of CT and the elements that comprise the CT system. It also describes the major features of these elements. Other documents, cited in the References, establish requirements for these CT system elements and describe their operation in greater detail.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on July 30, 2016.

Table of Contents

1. Introduction

Certificate transparency (CT) is a set of mechanisms designed to deter, detect, and facilitate remediation of certificate mis-issuance (as defined below). CT deters mis-issuance by encouraging CAs to publish the certificates that they issue in a set of publically-accessible logs. Each log uses a Merkle tree design to ensure that it is an append-only database, and the log entries are digitally signed by the log operator. Monitoring of logs detects mis-issuance. Remediation of mis-issuance is effected via certificate revocation.

In the context of CT, the term mis-issuance refers to violations of either semantic or syntactic constraints associated with certificates [draft-trans-threat-analysis]. The fundamental semantic constraint for a (Web PKI) certificate is that it was issued to an entity that is authorized to represent the Subject name in the certificate. If any Subject Alternative Names (SANs) are present in the certificate, the entity also must be authorized to represent them. (It is also assumed that the entity requested the certificate from the CA that

issued it.) Throughout the remainder of this document we refer to a semantically mis-issued certificate as "bogus."

A certificate is characterized as syntactically mis-issued if it violates syntax constraints associated with the class of certificates that it purports to represent. Syntax constraints for certificates are established by certificate profiles, and typically are application-specific. For example, certificates used in the Web PKI environment might be characterized as domain validation (DV) or extended validation (EV) certificates. Certificates issued for use by applications such as IPsec or S/MIME have different syntactic constraints from those issued in the Web PKI context. Throughout the remainder of this document we refer to a syntactically mis-issued certificate as "erroneous." From a security perspective, erroneous certificates are not perceived as being as significant a concern as bogus certificates.

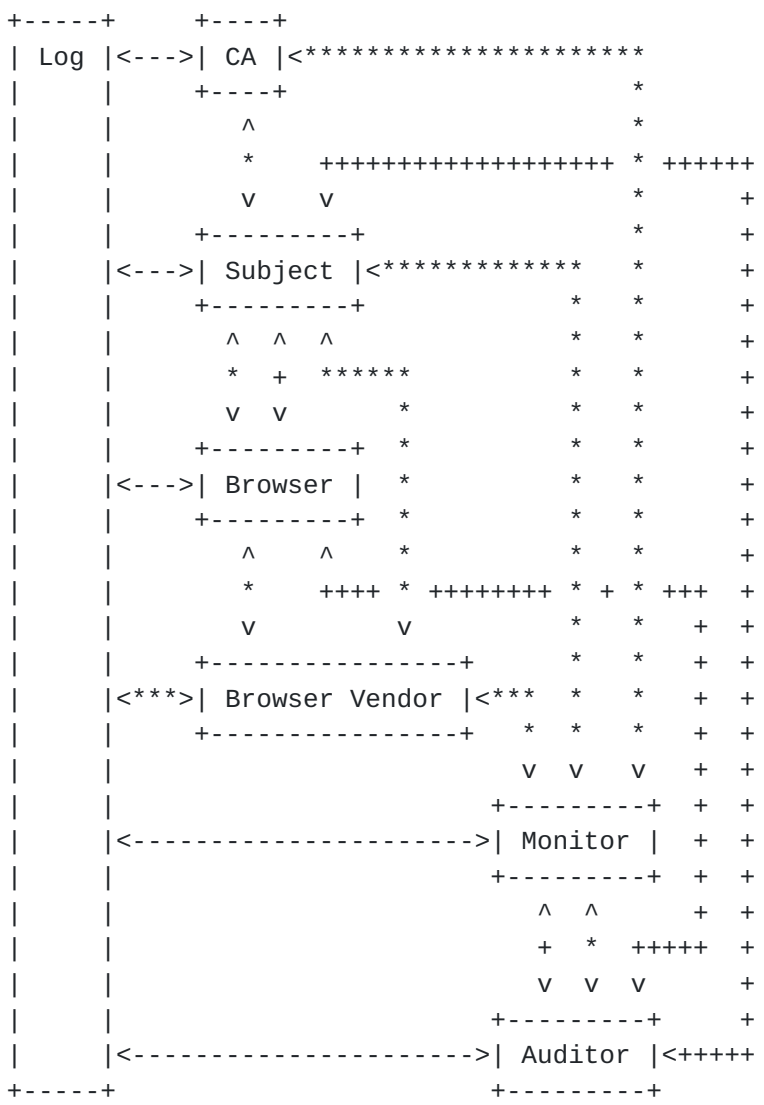
As noted above, CT deters mis-issuance by encouraging CAs to log the certificates that they issue. A CT log is a publicly auditable, append-only, database of issued certificates [6962-bis] based on a binary Merkle hash tree [Merkle]. Each CT log operates in a fashion that enables external entities (Auditors) to detect inconsistent behavior. As a result, logs need not be operated by trusted (third) parties. Some forms of log misbehavior require comparing information gleaned from multiple sources, e.g., using mechanisms such as the ones described in [Gossip]. If an Auditor detects misbehavior by the log, it will notify Monitors (described below) and Browser Vendors that it serves. In turn, the Monitors and Browser Vendors are expected to cease relying onlogs that repeatedlymisbehave in a fashion indicative of malice. (Ultimately, what constitutes malicious misbehavior will be determined by Monitors and Browser Vendors, and thus is outside the scope of this document.)

A bogus certificate that has been logged will be detected by an entity (a Monitor) that observes the log and that has knowledge of all legitimate certificates issued to the set of certificate Subjects that it serves. If a Monitor detects a log entry for a certificate that is inconsistent with the reference data for a Subject, the Monitor notifies the Subject. (A Subject may perform self-monitoring.) Thus Monitors implement the mis-issuance detection aspect of CT.

CAs are presumed to be deterred from logging mis-issued certificates, because of the implied reputational consequences. (The assumption is that a CA that is detected repeatedly mis-issuing certificates will, in time, be blacklisted by the Browser Vendors (who control the set of CAs that are accepted by Browsers).

Revocation of a bogus/erroneous certificate is the primary means of remedying mis-issuance. A browser vendor may distribute a "blacklist" of mis-issued certificates or a bad-CA-list of certificates of CAs that have mis-issued certificates. Browsers may then use such lists to reject certificates on the blacklist, or certificates issued by CAs whose certificates are on the bad-CA-list. This form of revocation, although not codified in IETF standards, is also a means of remediation for mis-issuance. Throughout the remainder of this document, references to certificate revocation as a remedy encompass these and analogous forms of revocation.

Figure 1 provides a top-level view of these elements of CT and their interactions.



Legend:
<---> Interface defined by CT
<***> Interface out of scope for CT
<+++> Proposed in Experimental Gossip Design

Figure 1 Elements of the CT Architecture

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Beneficiaries of CT

There are three classes of beneficiaries of CT: certificate Subjects, TLS Clients, and Certification Authorities (CAs). In the initial context of CT, the Web PKI, Subjects are web sites and TLS Clients are Browsers employing HTTPS to access these web sites. CAs are the issuers of certificates used in the Web PKI context.

A certificate Subject benefits from CT because CT enables Monitors to detect certificates that have been mis-issued in the name of that Subject. A Subject learns of a bogus/erroneous certificate (issued in its name), via a Monitor, as noted above. (The Monitor function may be provided by the Subject itself, i.e., self-monitoring, or by a third party trusted by the Subject.) When a Subject is informed of certificate mis-issuance by a Monitor, the Subject is expected to request/demand revocation of the bogus/erroneous certificate by the issuing CA and/or by the browser vendors (if the CA refuses to revoke the certificate).

A Subject also may benefit from the Monitor element of CT even if the Subject's legitimate certificate(s) has(have) not been logged. If the bogus/erroneous certificate is logged and if a Monitor has been provided with reference data from the Subject, then monitoring of logs for certificates issued in the Subject's name suffices to detect an instance of mis-issuance targeting the Subject. (If a CA operates a Monitor on behalf of its Subjects, then the CA has the requisite information to detect bogus/erroneous certificates in logs that it observes.)

A TLS client (Browser) benefits from CT if the TLS client rejects a mis-issued certificate, i.e., treats the certificate as invalid. A TLS client is protected from accepting a mis-issued certificate if that certificate is revoked, and if the TLS client checks the revocation status of the certificate. (A TLS client also is protected if a browser vendor "blacklists" a certificate or a CA as noted above.) A TLS client also may benefit from CT if the client validates a Signed Certificate Timestamp (SCT) [6962-bis] associated with a certificate, and rejects the certificate if the SCT is invalid.

CAs are also CT beneficiaries. If one CA issues a legitimate certificate to a Subject, and another CA issues a bogus certificate, the second certificate can be detected by a Monitor (if the bogus certificate has been logged). In this fashion the CA that issued the legitimate certificate benefits, since the bogus certificate is detected and, presumably revoked. Even the CA that issued the bogus certificate is a potential beneficiary. If the bogus certificate was issued as a result of an error or an (undetected) attack, CT can help

the CA become aware of the error or attack and act accordingly. This is presumed to be beneficial to the reputation of this CA.

3. The Elements of the CT Architecture

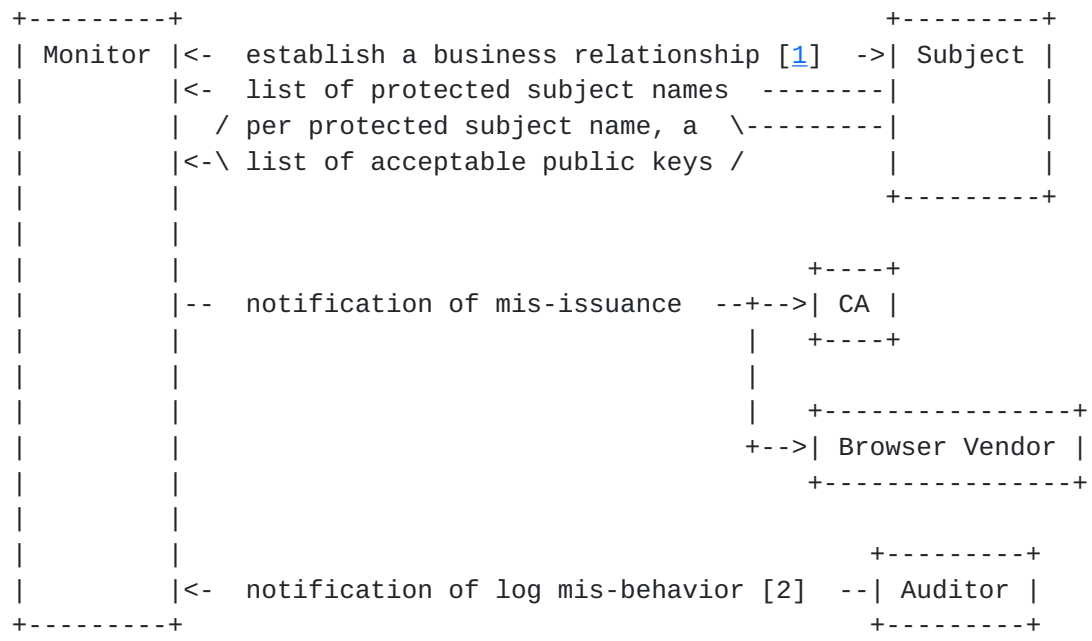
There are six elements of the CT architecture: logs, CAs, Monitors, Subjects, TLS clients (especially browsers and browser vendors), and Auditors. CAs, Subjects, and TLS clients are pre-existing elements affected by CT if they choose to participate. Because not all CAs, Subjects, and TLS clients may choose to participate in CT, these elements are qualified as "CT-aware" to distinguish them from existing instances of these types of Web PKI elements. Logs, Monitors, and Auditors are new elements introduced by CT and thus they are intrinsically CT "aware". Figure 2 shows how all of these elements interact with the central CT element, the log. Figure 3 shows how the pre-existing elements interact with one another under CT. Figure 4 shows the interactions of monitors and auditors that are not covered by Figure 2.

[1] The get-sth operation is performed periodically, and get-entries is performed each time a new STH is available.
[2] See Section 3.5 for privacy and performance caveats.
[3] If the Auditor stores copies of all Log entries, then this operation is not needed.

Figure 2 Interactions with a Log

[1] Not subject to standardization.
[2] Optionally including SCTs in an extension.
[3] Optional, via an OCSP response or in a TLS extension.

Figure 3 Interfaces of Pre-existing Elements



[1] In the case of a self-monitor, the business relationship is trivial - the Subject and Monitor are the same organization.
[2] An entity performing the Monitor function MAY also choose to implement some of the Auditor functions. In that case the Monitor/Auditor interface is trivial. If the Auditor is separate, we note that there is no interface defined at the time of this writing.

Figure 4 Monitor and Auditor Interfaces

3.1. Logs

Logs are the central elements of the CT architecture. Logging of certificates enables Monitors to detect mis-issuance and, subsequently, to trigger Subjects to issue revocation requests to CAs and/or browser vendors and to notify CAs and browser vendors directly. Logging also deters mis-issuance, as noted above. The interfaces to a log are defined in [6962-bis], as are the details of how a log operates.

Briefly, a certificate chain (that must be verifiable under a trust anchor acceptable to the log) is submitted to a log by a CA, Subject or other party. The log creates an entry for the terminal certificate in the chain, and returns this Signed Certificate Timestamp (SCT) to the submitter. The SCT can be conveyed to a browser in one of three ways: it can be incorporated into a certificate by the CA that issues it, as described later. (A CA also may submit a so-called "pre-certificate" to a log, to acquire an SCT for inclusion in the certificate, prior to signing the certificate.) It also can be

conveyed explicitly in the TLS handshake or in OSCP data generated by a CA. The SCT is a token that can be verified by browsers to establish, to first order, that a certificate has been logged. See [6962-bis] for additional details of SCTs.

All clients that interact with a log require access to metadata associated with each log upon which they rely. This metadata includes the URL and public key for the log, the list of trust anchors accepted by the log, the hash and signature algorithms employed, etc. Log metadata is made available to log clients via out of band means that are generally outside the scope of the CT specifications. In the Web PKI context, CT assumes that browser vendors will make the necessary log metadata available to browsers via the same mechanisms used to convey trust anchor (and vendor-managed revocation data). Log metadata provided via this channel is not mutable by log operators (since it is part of browser configuration data), with one exception. When a log ceases operation, it publishes its final STH, enabling clients to verify previous log entries and to detect any (unauthorized) additions to the log. See [6962-bis] for additional details.

An open question is how other log clients receive the metadata they require to interact with the log in a predictable fashion. For example, a log may elect to check the syntax of certificates relative to [RFC5280], or it may skip some of all of the checks specified there. Absent a way to determine what checks a log will perform on submitted certificates, a CA (or other submitter) has no way to know whether a submitted certificate will be accepted by a given log. Similarly, a Monitor needs to acquire log metadata so that the Monitor can locate the log and verify the signatures on log entries.

3.2. CT-aware Certification Authorities (CAs)

A (CT-aware) CA interacts with a log to submit a certificate (or a pre-certificate) to create a log entry. (Most logged certificates are expected to be end-entity certificates, each associated with the web site that it represents. However, it also is possible to log a CA certificate under certain circumstances. See Section 3.2.3 of [6962-bis].) The pre-certificate capability is offered to facilitate rapid deployment of CT. It has the advantage that web sites need not make any software changes to acquire one or more SCTs, because the SCTs are embedded in the certificate itself. There is, however, a downside of embedding SCTs in certificates. If a log that provided an SCT is compromised or otherwise becomes unacceptable to browsers and Monitors, the certificate associated with that SCT will have to be re-issued with a replacement SCT. Thus, in the long term, other options for conveying an SCT, i.e., via the TLS handshake or in an

OCSP response (perhaps "stapled" into the handshake [RFC6961]), are preferred [TLS-Server].

A CA also may submit a "name-redacted" pre-certificate to a log. A name-redacted pre-certificate includes one or more "?" labels in lieu of DNS name components. See Section 4.2 of [6962-bis] for more details. Name-redaction is a feature of CT designed to enable an organization to request a CA to log its certificates without revealing all of the DNS name components in the certificate that will be matched to the log entry. This is an attractive feature for organizations that want to benefit from CT without revealing internal server names as a side effect of logging. An end-entity certificate that is to be treated as logged via this mechanism contains a critical (X.509v3) extension that indicates which labels have been redacted in the log entry. This extension is needed to enable TLS clients and Monitors to match a received certificate against the corresponding log entry in an unambiguous fashion. See Section <TBD> of [CA-Subject] for more details.

The CT architecture does not mandate a specific number of SCTs that should be associated with a certificate. Browser vendors might establish requirements for the minimum number of associated SCTs in different contexts, but such requirements are outside the scope of the CT architecture.

[CA-Subject] describes the requirements imposed on CT-enabled CAs.

3.3. Monitors

The primary role of a Monitor is to observe a set of logs, looking for log entries of interest. A Subject may act as a self-monitor, or may make use of the services of a third-party Monitor, as noted earlier.

In the self-monitoring context, log entries of interest are ones that contain a Subject or Subject Alternative Name (SAN) associated with the Subject's web site(s). (Name-constrained CA certificates and wildcard certificates also have to be examined to detect certificates that would match the end-entity certificates associated with a Subject's web sites.) Whenever a certificate of interest is detected, the Subject compares it with the public key information associated with its certificate(s). If there is a mismatch, this indicates that this logged certificate was mis-issued. The Subject contacts the CA that issued the certificate (using the Issuer name in the

certificate) and requests revocation of the mis-issued certificate, to resolve the problem. (The means by which a Subject determines how to contact a CA based on the issuer name is outside the scope of the CT architecture.) The means by which a Subject determines which set of logs to watch also is outside the scope of the CT architecture. It is anticipated that there will be a small number of logs that are widely used, and that the metadata for these logs will be available from browser vendors.

A third-party Monitor watches for certificates of interest to its clients. Each client of a third party Monitor supplies the Monitor with a list of Subject names and SANs associated with the client's web site(s), and public key information associated with each name. (As a special case, if a CA offers a Monitor service to its clients, then the CA/Monitor already has this information.) The Monitor watches a set of logs looking for entries that match the client certificates of interest. If it detects an apparent mis-issued certificate, the Monitor contacts the client and forwards the log entry, along with log metadata. The client (Subject) then follows the procedure noted above to request revocation of the mis-issued certificate.

Note that a Monitor does not try to detect mis-behavior by a log. That is the responsibility of an Auditor. [Monitor-Auditor] defines the requirements for a Monitor (self of third-party) and discusses additional operational details.

Note also that CT does not include any mechanisms designed to detect misbehavior by a Monitor. A self-Monitor does not require such mechanisms; Subjects who elect to rely upon third-party Monitors would benefit from such mechanisms. See [Monitor-Auditor] for the requirements imposed on Monitors by CT and for a more detailed description of how a Monitor operates.

3.4. CT-aware Subjects (TLS web servers)

A (CT-aware) Subject (e.g., a web site operator) can submit its certificate(s) to a log, and acquire an SCT for each certificate it submits (see Section 4.1 of [6962-bis]). There are three reasons why a Subject may choose to log its own certificate(s): (1) its CA did not embed an SCT in the certificate(s) it issued to the Subject, (2) the Subject wants to acquire SCTs from additional logs, or (3) the Subject wants the flexibility offered by conveying SCTs (from logs of its choosing) in the TLS handshake. [CA-Subject] describes the requirements imposed on Subjects for delivery of SCTs to CT-aware TLS clients.

Every Subject should either perform self-monitoring, or become a client of a third-party Monitor so that bogus certificates issued in the name of the Subject will be detected. When a Subject is notified of a bogus certificate issued in its name, the Subject contacts the CA that issued the certificate and requests that it be revoked, using whatever mechanisms the CA provides for such requests. The Subject may also contact browser vendors and ask that they put the certificate on a blacklist of mis-issued certificates or put the CA's certificate on a bad-CA-list, if the CA refuses to revoke the bogus certificate. [CA-Subject] describe the requirements established for for CT-aware Subjects.

3.5. CT-aware TLS clients (web browsers)

As noted in Section 2, a TLS client can benefit from CT even without actively participating. A Monitor will detect a mis-issued, logged certificate and notify the affected Subject. The Subject will, in turn attempt to trigger revocation by the CA that mis-issued the certificate in question, ultimately asking browser vendors to blacklist the certificate (or the CA) if revocation is not effected. Thus a TLS client that processes certificate revocation status data, e.g., CRLs, OCSP responses, will be protected from bogus certificates that have been logged, detected, and revoked.

If a TLS client required that every certificate it accepted was accompanied by an SCT, the client could have some confidence that the certificate had been logged. This would increase confidence that the certificate, if it were mis-issued, would have been revoked. However, there are two problems with mandating that every TLS client reject (treat as invalid) any certificate that is not accompanied by an SCT. First, such behavior does not accommodate incremental deployment of CT. Second, the mere presence of an SCT is not a guarantee that the certificate has been logged.

To have high confidence that a certificate has been logged, a TLS client would have to verify that a log entry exists for the certificate. This requires acquisition of an inclusion proof from the log (see Section 4.5 of [6962-bis]). Requesting an inclusion proof directly from a log for a certificate discloses to a log that the TLS client is interested in the certificate in question. For a browser, this would disclose which web sites a user was visiting, a potential privacy concern for many users. Also, the data acquisition and processing might pose an unacceptable burden for some TLS clients, (e.g., browsers), and might not be performed in realtime anyway. Thus CT-aware TLS clients are not expected to fetch an inclusion proof in realtime, e.g., during TLS connection establishment. Such clients also are not expected to reject a certificate that has no associated

SCT, because there is no plan for incremental deployment of CT that accommodates such rejection in a backwards compatible fashion. Nonetheless, if an SCT is provided with a certificate, a CT-aware TLS client could verify the signature and the SCT data for the certificate in question. If performing these checks would not impose an undue burden on the TLS client, the checks would help detect errors in SCTs and provided feedback to log operators (via Subjects).

A TLS client that is a browser might discriminate against a certificate presented for a web site if the certificate is not accompanied by an SCT, e.g., providing an indication of this via the user interface. See [browser-vendor] for the requirements established for CT-aware browsers and browser vendors.

3.6. Auditors

Auditors perform checks intended to detect mis-behavior by logs. There are four log behavior properties that Auditors check:

1. The Maximum Merge Delay (MMD)

2. The STH Frequency Count

3. The append-only property

4. The consistency of the log view presented to all query sources

The first three of these checks are easily performed using existing log interfaces and log metadata (see [6962-bis]). For example, an Auditor could submit a certificate to a log and request an STH after the indicated MMD, to verify that the log is achieving its advertised MMD. The last check is more difficult to perform because it requires a way to share log responses among a set of CT elements, perhaps including browsers, web sites, Monitors, and Auditors, e.g., using so-called gossiping [Gossip]. There is as yet no standard for gossiping and thus the last check is NOT part of Auditor requirements at this time. See [Monitor-Auditor] for additional details of Auditor operation.

4. Security Considerations

CT is a system created to improve security for X.509 public key certificates, especially in the Web PKI context. An attack analysis [draft-trans-threat-analysis] examines the types of attacks that can be mounted against CT, to effect mis-issuance, and how CT addresses (or fails to address) each type of attack. That analysis is based on the architecture described in this document, and thus readers of this

document are referred to that one for a thorough discussion of the security aspects of CT. Briefly, CT logs represent a viable means of deterring semantic mis-issuance of certificates. Monitors are an effective way to detect semantic mis-issuance of logged certificates. The CT architecture enables certificate Subjects to request revocation of mis-issued certificates, thus remedying such mis-issuance. Residual vulnerabilities exist with regard to some forms of log and Monitor misbehavior, because the architecture does not include normative means of detecting such behavior. The current design also does not ensure the ability of Monitors to detect syntactic mis-issuance of certificates. This is because provisions for asserting the type of certificate being issued, for inclusion in an SCT, have not been standardized.

5. IANA Considerations

<TBD>

6. References

6.1. Normative References

[Merkle] Merkle, R. C. (1988). "A Digital Signature Based on a Conventional Encryption Function." Advances in Cryptology - CRYPTO '87. Lecture Notes in Computer Science 293. p. 369

[6962-bis] Laurie, B., Langley, A., Kasper, E., Messeri, E., and R. Stradling, "Certificate Transparency," draft-ietf-trans-rfc6962-bis-10 (work in progress), October 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, June 2013.

[RFC6961] Pettersen, Y., "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension," RFC 6961, June 2013.

6.2. Informative References

[draft-trans-threat-analysis] Kent, S., "Attack Model and Threat for Certificate Transparency," draft-ietf-trans-threat-analysis-03 (work in progress), October 2015.

[Gossip] Nordberg, L., Gillmor, D., and Ritter, T., "Gossiping in CT," draft-ietf-trans-gossip-01 (work in progress), October 2015.

[Monitor-Auditor] <TBD>

[CA-Subject] <TBD>

[browser-vendor] <TBD>

7. Acknowledgments

<TBD>

Authors' Addresses

Stephen Kent
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
US

Email: kent@bbn.com

David Mandelberg
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
US

Email: david@mandelberg.org

Karen Seo
BBN Technologies
10 Moulton St.
Cambridge, MA 02138
US

Email: kseo@bbn.com