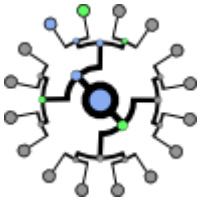

Certificate Transparency

Search this site

How Certificate Transparency Works

Certificate Transparency adds three new functional components to the current SSL certificate system:

- Certificate logs
- Certificate monitors
- Certificate auditors

These functional components represent discrete software modules that provide supplemental monitoring and auditing services. They are not a replacement for, or an alternative to, the current SSL certificate system. Indeed, these components do not change the fundamental chain-of-trust model that lets clients validate a domain and establish a secure connection with a server. Instead, these components augment the chain-of-trust model by providing support for public oversight and scrutiny of the entire SSL certificate system.

Basic Log Features

At the center of the Certificate Transparency system lie certificate logs. A certificate log is a simple network service that maintains a record of SSL certificates. Certificate logs have three important qualities:

- They're append-only--certificates can only be added to a log; certificates can't be deleted, modified, or retroactively inserted into a log.
- They're cryptographically assured--logs use a special cryptographic mechanism known as Merkle Tree Hashes to prevent tampering and misbehavior.
- They're publicly auditable--anyone can query a log and verify that it's well behaved, or verify that an SSL certificate has been legitimately appended to the log.

The number of logs does not need to be large: there need to be enough logs so that log failures or temporary outages are not a problem, but not so many that they become difficult to monitor--say, more than 10 but much less than 1000. Each log operates independently of the other logs (that is, there's no automatic replication among the logs).

The append-only nature of a log allows it to use a special type of cryptographic hash to prove that it's not corrupt

and that the log operator has not deleted or modified any certificates in the log. This special hash--known as a Merkle Tree Hash--also makes it possible for auditors to detect whether someone has forked a log or inserted back-dated certificates into a log. For more information about the hashing mechanism, see How Log Proofs Work.

Every certificate log must publicly advertise its URL and its public key (among other things). Anyone can interact with a log via HTTPS `GET` and `POST` messages.

Basic Log Operations

Anyone can submit a certificate to a log, although most certificates will be submitted by certificate authorities and server operators. When someone submits a valid certificate to a log, the log responds with a signed certificate timestamp (SCT), which is simply a promise to add the certificate to the log within some time period. The time period is known as the maximum merge delay (MMD).

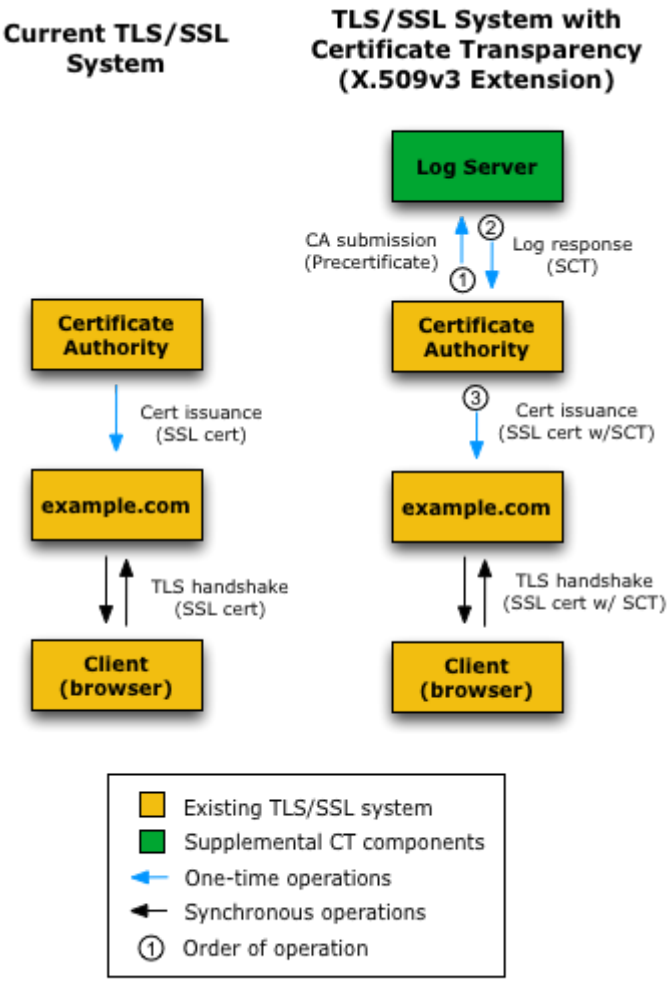

Figure 1

The MMD helps ensure that the log server adds the certificate to the log within a reasonable timeframe and doesn't block the issuance or use of the certificate, whilst allowing the log to run a distributed farm of servers for resilience and availability. The SCT accompanies the

certificate throughout the certificate's lifetime. In particular, a TLS server must deliver the SCT with the certificate during the TLS handshake.

Certificate Transparency supports three methods for delivering an SCT with a certificate. Each is described below.

X.509v3 Extension

Certificate authorities can attach an SCT to a certificate using an X.509v3 extension. Figure 1 shows how this works. The certificate authority (CA) submits a precertificate to the log, and the log returns an SCT. The CA then attaches the SCT to the precertificate as an X.509v3 extension, signs the certificate, and delivers the certificate to the server operator.

This method does not require any server modification, and it lets server operators continue to manage their SSL certificates the same way they always have.

TLS Extension

Server operators can deliver SCTs by using a special TLS extension (see figure 2). In this case, the CA issues the certificate to the server operator, and the server operator submits the certificate to the log. The log sends the SCT to the server operator, and the server operator uses a TLS extension with type `signed_certificate_timestamp` to deliver the SCT to the client during the TLS handshake.

This method does not change the way a CA issues SSL certificates. However, it does require a server change to accommodate the TLS extension.

OCSP Stapling

Server operators can also deliver SCTs by using Online Certificate Status Protocol (OCSP) stapling (see figure 2). In this case, the CA simultaneously issues the certificate to the log server and the server operator. The server operator then makes an OCSP query to the CA, and the CA responds with the SCT, which the server can include in an OCSP extension during the TLS handshake.

This method allows CAs to take responsibility for the SCT but does not delay the issuance of the certificate, since the CA can get the SCT asynchronously. It does, however, require modification of the server to do OCSP stapling.

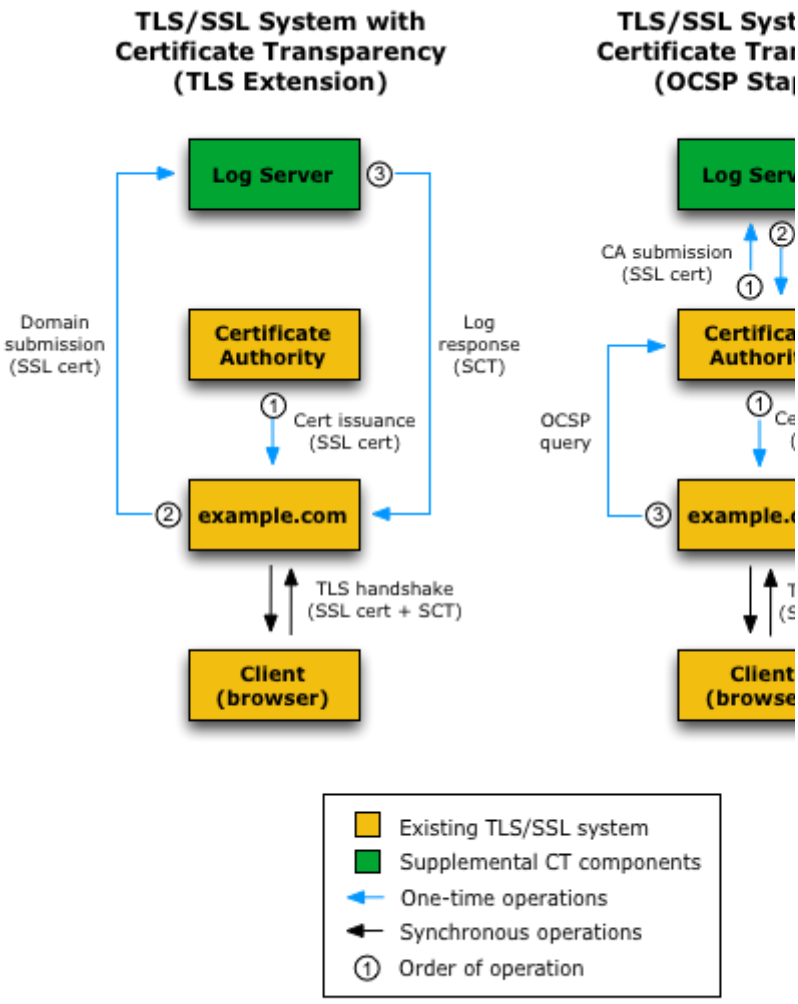

Figure 2

Basic Monitor and Auditor Operations

Monitors watch for suspicious certificates in logs, such as illegitimate or unauthorized certificates, unusual certificate extensions, or certificates with strange permissions (for example, CA certificates). Monitors also verify that all logged certificates are visible in the log. They do this by periodically fetching all the new entries that have been added to a log. As a result, most monitors have complete copies of the logs they monitor. If a log goes offline for a prolonged period of time, and a monitor has a copy of the log, then the monitor could act as a backup read-only log and provide log data to other monitors and auditors that are trying to query the log.

Auditors verify the overall integrity of logs. Some auditors can also verify whether a particular certificate appears in a log. They do this by periodically fetching and verifying log proofs. Log proofs are signed cryptographic hashes a log uses to prove it's in good standing. Every log must provide their log proofs on demand.

Auditors can use log proofs to verify that a log's new entries have always been added to the log's old entries, and that nobody has ever corrupted a log by retroactively

inserting, deleting, or modifying a certificate. Auditors can also use log proofs to prove that a particular certificate appears in a log. This is particularly important because the Certificate Transparency framework requires that all SSL certificates be entered into a log. If a TLS client determines (via an auditor) that a certificate is not in a log, it can use the SCT from the log as evidence that the log has not behaved correctly. For more information about log proofs, see How Log Proofs Work.

While log proofs allow an auditor or a monitor to verify that their view of a particular log is consistent with their past views, they also need to verify that their view of a particular log is consistent with other monitors and auditors. To facilitate this verification, auditors and monitors exchange information about logs through a gossip protocol. This asynchronous communication path helps auditors and monitors detect forked logs.

Typical System Configuration

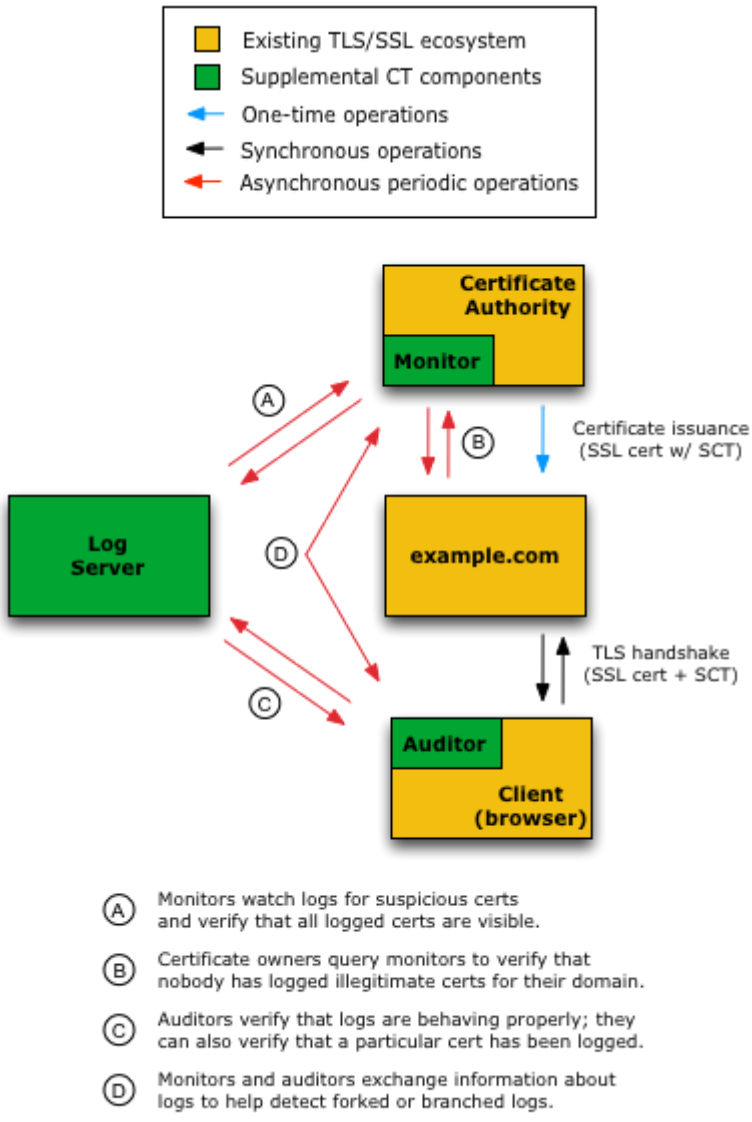

Figure 3

The Certificate Transparency framework doesn't prescribe any particular configuration or placement of monitors and

auditors within the existing SSL certificate system. That said, some configurations are more common than others. In a typical configuration, a CA runs a monitor and a client (browser) runs an auditor (see figure 3). This configuration simplifies the messaging that's necessary for monitoring and auditing, and it lets certificate authorities and clients develop monitoring and auditing systems that meet the specific needs of their customers and users. Some of the processes that drive this configuration are described below.

Certificate Issuance

A CA obtains an SCT from a log server and incorporates the SCT into the SSL certificate using an X.509v3 extension (for more details on this process, see figure 1). The CA then issues the certificate (with the SCT attached) to the server operator. This method requires no server updates (all servers currently support X.509v3 extensions), and it lets server operators manage their certificates the same way they've always managed their SSL certificates.

TLS Handshake

During the TLS handshake, the TLS client receives the SSL certificate and the certificate's SCT. As usual, the TLS client validates the certificate and its signature chain. In addition, the TLS client validates the log's signature on the SCT to verify that the SCT was issued by a valid log and that the SCT was actually issued for the certificate (and not some other certificate). If there are discrepancies, the TLS client may reject the certificate. For example, a TLS client would typically reject any certificate whose SCT timestamp is in the future.

Certificate Monitoring

Most monitors will likely be operated by certificate authorities. This configuration lets certificate authorities build efficient monitors that are tailored to their own specific monitoring standards and requirements.

Certificate Auditing

Most auditors will likely be built into browsers. In this configuration, a browser periodically sends a batch of SCTs to its integrated auditing component and asks whether the SCTs (and corresponding certificates) have been legitimately added to a log. The auditor can then asynchronously contact the logs and perform the verification.

Other System Configurations

In addition to the typical configuration described above, where monitors and auditors are tightly integrated with existing TLS/SSL components, Certificate Transparency supports many other configurations. For example, monitors could operate as standalone entities, providing paid or unpaid services to certificate authorities and server operators (see figure 4). A monitor could also be run by a server operator, such as a large Internet entity like Google, Microsoft, or Yahoo. Likewise, an auditor could operate as a standalone service or it could be a secondary function of a monitor.

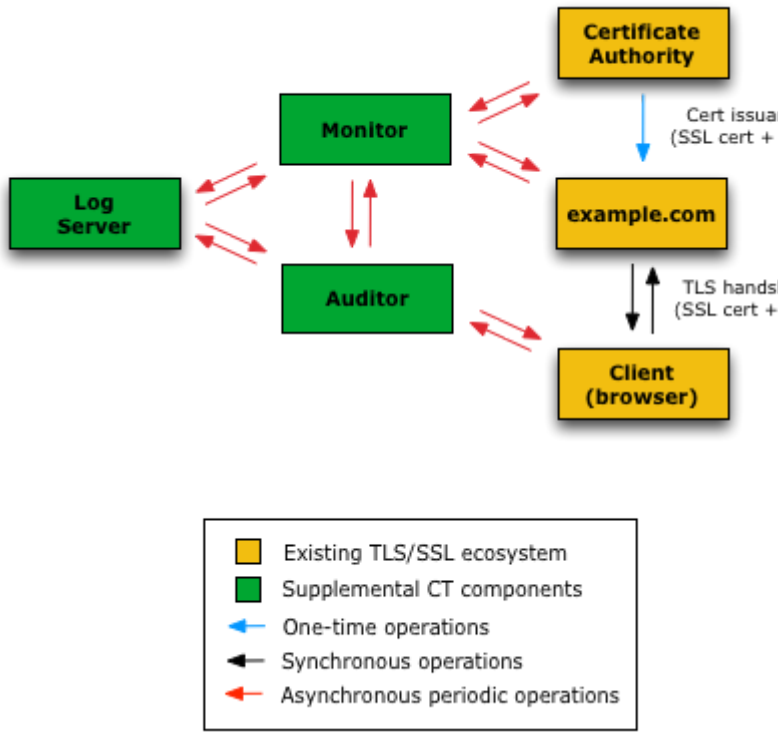

Figure 4