Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Theodor Nolte

Certificate Transparency Deployment Study

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Theodor Nolte

Certificate Transparency Deployment Study

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas C. Schmidt
Zweitgutachter: Prof. Dr. Franz Korf

Eingereicht am: 1. November 2018

Theodor Nolte

Thema der Arbeit
Certificate Transparency Deployment Study

Stichworte
TLS, HTTPS, Web-PKI, Certificate Transparency

Kurzzusammenfassung
Um Webserver-Zertifikate auditierbar zu machen, erweitert Certificate Transparency (CT) das TLS-Ökosystem um sogenannte CT-Logs, welche ein nicht löschbares, öffentliches Verzeichnis darstellen. Das Hinzufügen eines Zertifikats in ein CT-Log wird durch einen sogenannten Signed Certificate Timestamp (SCT) quittiert. Mit dem Übermitteln von zugehörigen SCTs zusammen mit dem Webserver-Zertifikat wird die Auditierbarkeit beim TLS-Handshake nachgewiesen. In dieser Arbeit wird die Verbreitung und die zeitliche Entwicklung der Verbreitung von CT im produktiven Einsatz analysiert. Die Anzahl der Zertifikate in CT-Logs hat exponentiell zugenommen. Website-Support von CT hat in den vergangenen zwölf Monaten von 26% auf gegenwärtig 58% zugenommen für die populärsten Domains, die per HTTPS erreichbar sind.

Theodor Nolte

Title of the paper
Certificate Transparency Deployment Study

Keywords
TLS, HTTPS, Web-PKI, Certificate Transparency

Abstract
Certificate Transparency (CT) extends the TLS ecosystem by so-called CT logs which represent an append-only public register in order to make webserver certificates auditable. The adding of a certificate into a CT log will be receipt by a so-called Signed Certificate Timestamp (SCT). At the TLS handshake, together with the webserver certificate, the transmission of the corresponding SCTs prove the availability for auditing. In this thesis, we analyze the deployment of CT and its evolution over time. The number of certificates in CT logs have seen exponentional growth. Website support for CT has increased over a period of twelf month from 26% to currently 58% for the most popular domains which are accessible via HTTPS.

Contents

List of Tables

List of Figures

1 Introduction

The TLS ecosystem which provides for secure communication in the internet is based on trust of users in the certificate authorities (CAs) which issue certificates. In the past, there have been incidents where CAs misissued certificates which put this trust into question. The problem with this misissued certificates is that they can go undetected for a long time and some CAs have showed in the past that they have no interest in publishing this incidents.

The most spectacular incident of this kind was the break into the dutch DigiNotar CA in the second quarter of 2011 [20]. The attacker gained access for all servers which issued certificates and was able to issue at least 531 webserver certificates for example for popular domains such as `google.com`, `microsoft.com`, or `skype.com` [10]. A wildcard certificate for `*.google.com` was used for man-in-the-middle attacks mainly in Iran. The civil rights organisation Electronic Frontier Foundation suspects the Iranian government to be behind of this attacks [23]. It tooks more than five weeks betweeen the first attack on 2011-07-10 and the revocation of trust in web browsers in August and September of 2011.

This misissuances – due to mistakes or initiated by an attacker – are wrong behavior of the CAs. This concerns the main part of TLS. Because TLS is based in the trust that the CAs work correctly; the trust that the identities asserted by the certificates have been checked correctly by the CAs.

In order to remedy this lack of trust, Google pushed for Certificate Transparency by enforcing websites to support CT to be displayed as trustworthy in the Google browser Chrome. The base idea of Certificate Transparency (CT) is to have a public index of webserver certificates which makes them publicly auditable. The index is created by so-called CT logs where certificates will be published. This adds to the trust model of the TLS an new element, the *public control*. Everyone – CAs, domain owners, and others – can monitor CT issued certificates.

Since April 2018, Google Chrome enforces for all websites to support CT, or they would be displayed as unsure [15].

In this thesis, we analyze the deployment of CT and its development. In detail, our contributions read:

- For a conceptual and technical background of CT.
- For CT log evolution. In April when CT support becomes required by Google Chrome we can see an exponential growth of entries in CT logs.
- For webserver deployment of CT. We will see that the deployment of CT has been intensely grown in the last year and is now widely in use by the most of the webservers. Also, the diversity of the used CT logs have been improved.

The remainder of this thesis is structured as follows. In chapter 2 on page 3 the basics of the Certificate Transparency ecosystem are shown and related work will be presented. In chapter 2.3 on page 20 we analyze the deployment of Certificate Transparency. Section 4.3 on page 35 is about the evolution of CT-logs. Section 3 on page 20 is about the webserver deployment of CT. Finally, this thesis concludes in chapter 5 on page 39 and gives an outlook.

2 Certificate Transparency and Related Work

Certificate Transparency (CT) [11, 8] extends the TLS ecosystem by so-called CT logs [9] which represent an append-only public register in order to make webserver certificates auditable. The publication of a certificate into a CT log will be receipt by a so-called Signed Certificate Timestamp (SCT). At the TLS handshake, together with the webserver certificate, the transmission of the corresponding SCTs prove the availability for auditing.

2.1 Functional Motivation

The HTTPS Public Key Infrastructure (web-PKI) without CT is based on the trust in the CAs to issue certificates without making errors. There is no organizational or technical mechanism to check for every certificate if it has been issued correctly.

CT mitigates this problem by making every webserver certificate auditable. The rationale of CT is that every issued webserver certificate (which supports CT) will be published. This enables everyone to check and verify the issuance of all certificates in the web-PKI. In particular domain owners could learn of every issued certificate of its domain. And CAs could check every certificate issued in its name.

CT extends the web-PKI by so-called CT logs where the webserver certificates will be published. Every certificate would be publishd into one ore more, usually at least into two CT logs. For the CT logs are no access restrictions, everyone can access them.

CT logs are append-only. This means there are technical mechanisms which makes it impossible to remove already published certificates or exchange them by others without noticing. Erroneous issued certificates are not hindered by CT. But they would be detected with a high probability by domain owners, the issuing CA, or by others. There already exist web-services which scan CT logged certificates systematically for misissuance [6].

CT extends the web-pki. No parts of the former web-pki are replaced. Nor is CT (just another) CA which signes every issued certificate again.

CT publication achieves:

1. No CA can issue a certificate without the fact that the domain owner could easily detect such a certificate. Because it is published in CT logs, he can search for it.

2. CAs and domain owners and also third parties can verify if certificates are wrongly issued.

3. Users can verify if the web certificate of a visited domain is published. Then, they can assume that the certificate due to its publication in CT logs has been checked by the belonging CA and the domain owner.

2.2 Technical Conceptuation

The main components of the TLS Public Key Infrastructure are the *CA* which issues certificates, the *webserver* which authenticates himself using certificates, and the *webbrowser* – acting as the users client – which checks the webservers identity by inspecting its certificate.

CT adds the component *CT log*, and two roles: *monitor* and *auditor* [11]. The CT log, monitor, and auditor are shown in figure 2.1.

An entity acting as a monitor checks if a CT log behaves correctly. For example, it verifies a new version of a CT log still contains all old entries. Also, a monitor knows every CT log entry and can watch for new CT log entries for a distinct domain. An example for such a monitor is Comodos crt.sh [6] (called *search*) which monitors all known CT logs.

An auditor verifies if a certificate is published in a CT log. To do this it needs the SCT of the CT log which belongs to the certificate.

Both checks – as a monitor and as an auditor – are based on cryptographic operations as we will see later in this chapter.

2.2.1 CT Log

A CT log publishes Webserver certificates. The CT log signs its publications and applies more cryptographic operations to prove it correct behavior. Therefore a CT log has a private and a public key.

The certificate entries are ordered by time and new entries will be appended only. This is named as the append-only property of a CT log. Periodically, for example every five minutes, the CT log publishes a new *version*, i.e. all newly added certificates become visible. When a certificate is accepted for publication, the CT log responds with a so-called Signed Certificate Timestamp (SCT). This SCT is a kind of a receipt, and is signed by the private key of the CT

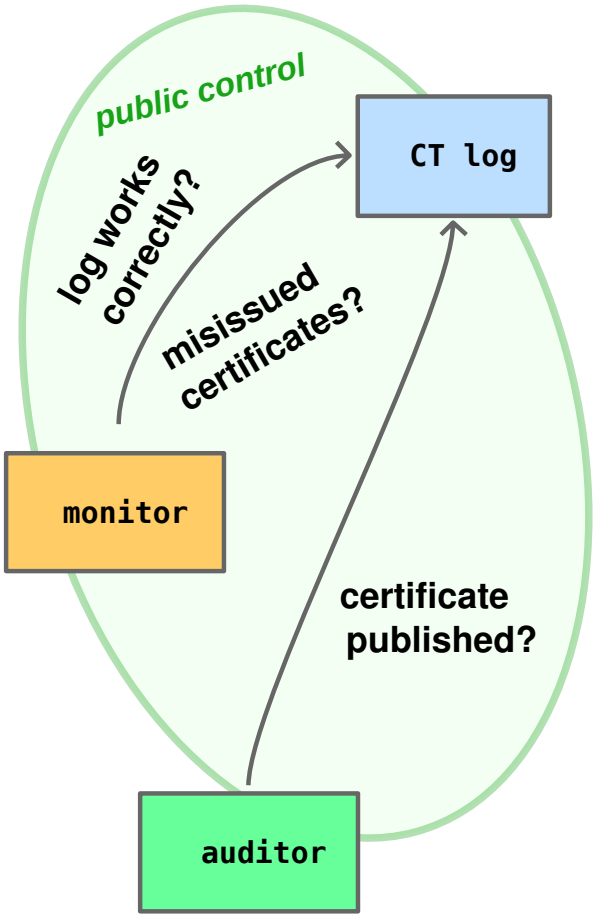

Figure 2.1: Components and roles of Certificate Transparency (CT)

log and can be verified by its public key. Because the private key is secret, only the CT log is capable to create a correct SCT.

The SCT distinctly points to the certificate entry. This entry not only contains the certificate itself but also its chain up to the root CA and some meta data such as the insertion time into the CT log.

The publication into CT logs is done anonymously and everyone can send a valid certificate for publication to a CT log. As a protection against certificate-spam CT logs only accept certificates which chains ends up by a root CA trusted by the web Public Key Infrastructure.

CT Log Proofs – CT Log from User Perspective

In contrast to a certificate authority, a CT log does not need to be *trusted* to work correctly. Instead, a CT log *proves* its integrity. Two kind of proves exist, the so-called *merkle audit proof*, and the *merkle consistency proof*.

Figure 2.2 shows an overview of the integration of CT into the TLS ecosystem. Typically, a CA and a domain owner who runs a webserver uses a monitor in order to learn for new certificates issued by a distinct CA or for a distinct domain (dotted green line between the CA and the monitor, and between the webserver and the monitor). A CA has a strong interest to know if it is compromised and if a certificate was issued in its name without permission (which would be the worst case for a CA). A domain owner with a webserver has a strong interest to know if a certificate was issued for its domain without permission. Such an misissued certificate can be used by an attacker to show an user a wrong webpage who browses to the domain. The dotted, green lines between the webserver and the auditor, and between the browser and the auditor mean that both, the domain owner and the user of the browser have an interest that the certificate is correctly published into the CT log.

The reason for a merkle audit proof is to answer an auditor as it were asking the question: *"Here is an SCT. Is the corresponding certificate published in this CT log?"* The merkle audit proof is a cryptographic proof of the publication of the certificate.

The monitor requests an merkle consistency proof as if it were asking the question: *"Works this CT log correctly, is its integrity correct?"* The most important aspect is that the CT log complies with the append-only property.

Merkle Hash Tree

Internally, the CT Log is build on a merkle hash tree [13]. A merkle hash tree (short: merkle tree) is a binary tree where an intermediate node is the hash of the concatenation of the two

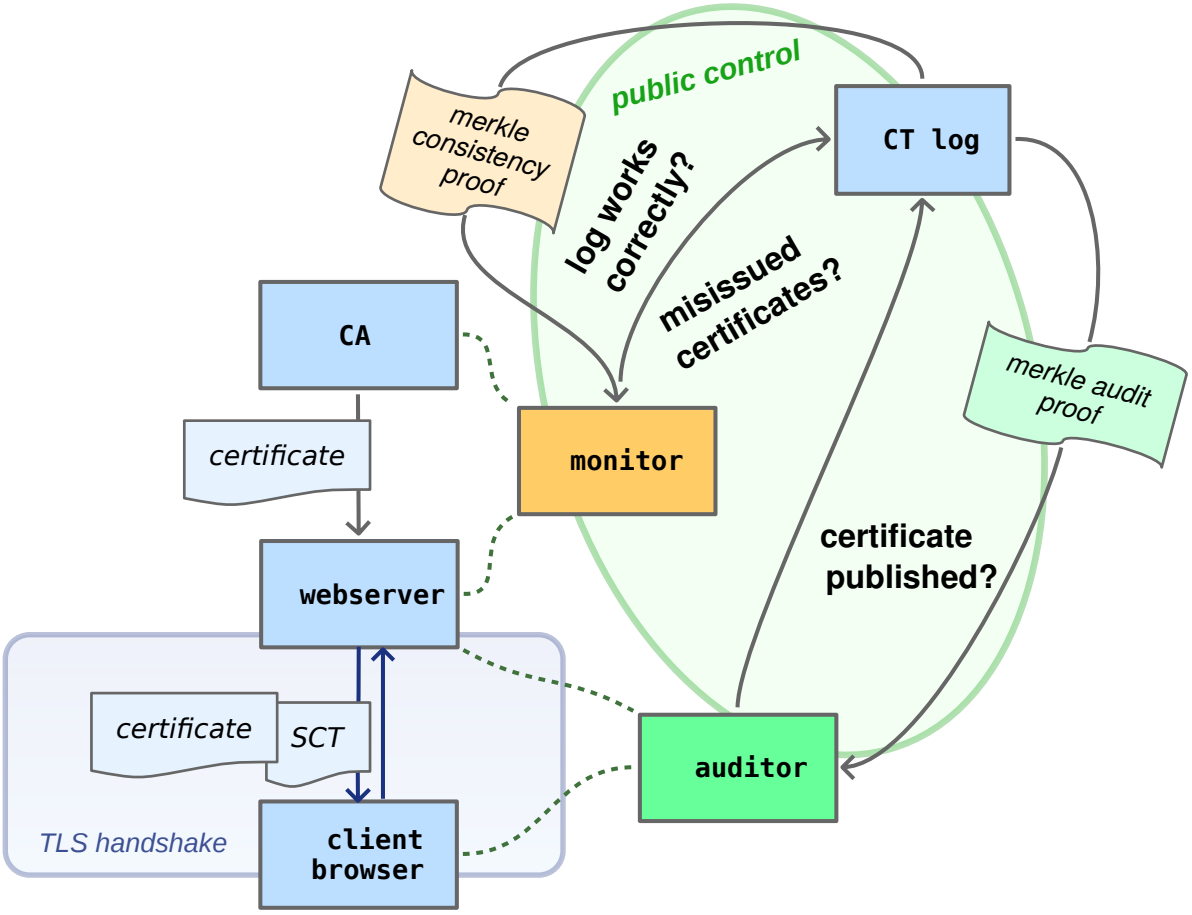

Figure 2.2: Interaction with components of Certificate Transparency and integration into the TLS ecosystem

children of that node. The leafs of the merkle tree are the hashes of the certificates (more exactly: the full certificate chain up to the root CA certificate).

The most easy way to cryptographically prove the publication of certificates by a log would be to hash the concatenation of all certificates and then sign this hash. To prove the publication of one single certificate by an auditor, the log would need to send all other certificates to the auditor. Then, the auditor could calculate the hash and could compare its result with the hash signed by the log. If any certificate differs it would lead to another hash. The time required for computing and the size of transferred data grows linear to the number of the certificate entries of the log. This corresponds to a complexity class of `O(n)`.

RFC 6962 [11] defines how to build the merkle tree of a CT log as follows:

Given a list of `n` entries:

```
C[n] := {c0, c1, ..., cn-1}.
```

The hash of an empty list is the hash of the empty byte-string:

```
MTH({}) := HASH()
```

The hash of a list with one entry, i.e. a leaf entry of the binary tree, is defined as:

```
MTH({c0}) := HASH(0x00 || c0)
```

The internal nodes of the binary tree are defined recursively. For `m > 1` let `k` be the largest power of 2 which is smaller than `m` (i.e. `k < m <= 2k`). Then is:

```
MTH(C[m]) := HASH(0x01 || MTH(C[0:k]) || MTH(C[k:m]))
```

Where `C[i:j]` is a list of `j-i` entries `ci` till `cj-1`:

```
C[i:j] := {ci, ci+1, ..., cj-1}
```

HASH is a hash algorithm which output length is always of `HASH_SIZE` bytes. Currently for CT only the hashing algorithm SHA-256 [1] is used. So the `HASH_SIZE` is `256/8 = 32`. The hash of the empty list then is: `MTH({}) = HASH() =` `e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`.

The node `MTH(C[n])` of all entries – the root of the binary tree – is named *tree head*. Periodically, the merkle tree will be re-created based on all current and in the meantime newly added certificate entries. The new tree head will be signed by the private key of the CT log and published to a new CT log version. This is the so-called *Signed Tree Head (STH)*.

Merkle Audit Proof

The *merkle audit proof* provides evidence for a given certificate to be included into a merkle tree as a leaf hash. The CT log answers an auditor for an inclusion proof request (question *certificate published?* in figure 2.2) with such an merkle audit proof.

Technically, the inclusion proof `PATH(m, C[n])` is provided as a list of nodes of the merkle tree for a given certificate entry `m` from all certificate entries `C[n]` of a CT log which are required to calculate the *tree hash* which is published as the STH by the CT log.

The *inclusion proof* is defined as: If there is exactly one entry (`C[1] = {c0}`), then the *inclusion proof* for a binary tree which contains only one single leaf is empty:

```
PATH(0, {c0}) = ()
```

In case of more than one entry (`n > 1`) the *inclusion proof* is defined recursively: Let `k` be the biggest power of 2 which is smaller than `n` (`k < n <= 2k`). The *inclusion proof* for the `m+1` element `cm` of a list of `n` elements (`m < n`) is:

```
m < k:

    PATH(m, C[n]) := PATH(m, C[0:k]) : MTH(C[k:n])

m => k:

    PATH(m, C[n]) := PATH(m - k, C[k:n]) : MTH(C[0:k])
```

The colon-opreator `:` means the concatenation of the `element` after the `list`: `appended_list = list : element`.

For example, the *inclusion proof* of the merkle tree in figure 2.3 of the certificate entry `c2` is calculated as:

```
PATH(2, C[8])
= PATH(2, C[0:4]) : MTH(C[4:8])
= PATH(0, C[2:4]) : MTH(C[0:2]) : MTH(C[4:8])
= PATH(0, C[2:3]) : MTH(C[3:4]) : MTH(C[0:2]) : MTH(C[4:8])
= () : MTH(C[3:4]) : MTH(C[0:2]) : MTH(C[4:8])
= [MTH(C[0:2]), MTH(C[3:4]), MTH(C[4:8])]
= [i, d, n]
```

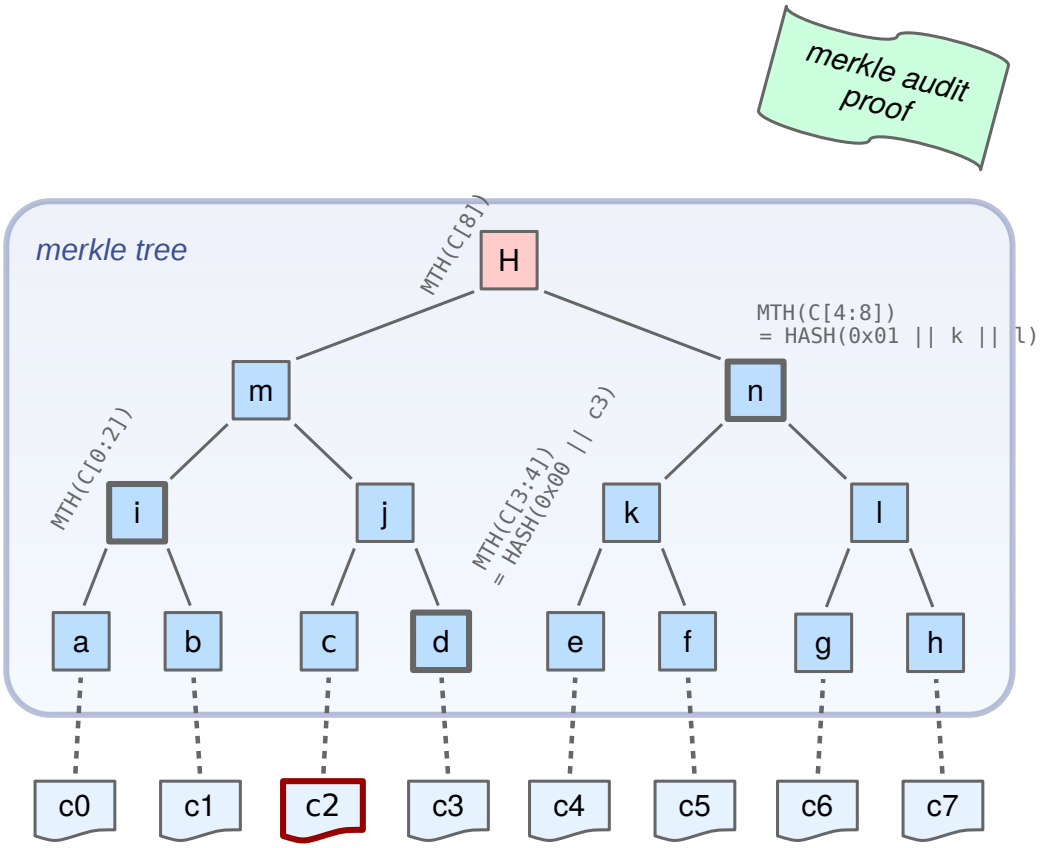

Figure 2.3: *merkle audit proof*: `PATH(c2, C[8]) = [i, d, n]`. With the nodes `i`, `d`, and `n` together with `c2` it is possible to calculate the tree head H. The result can be compared with the published STH of the CT log.

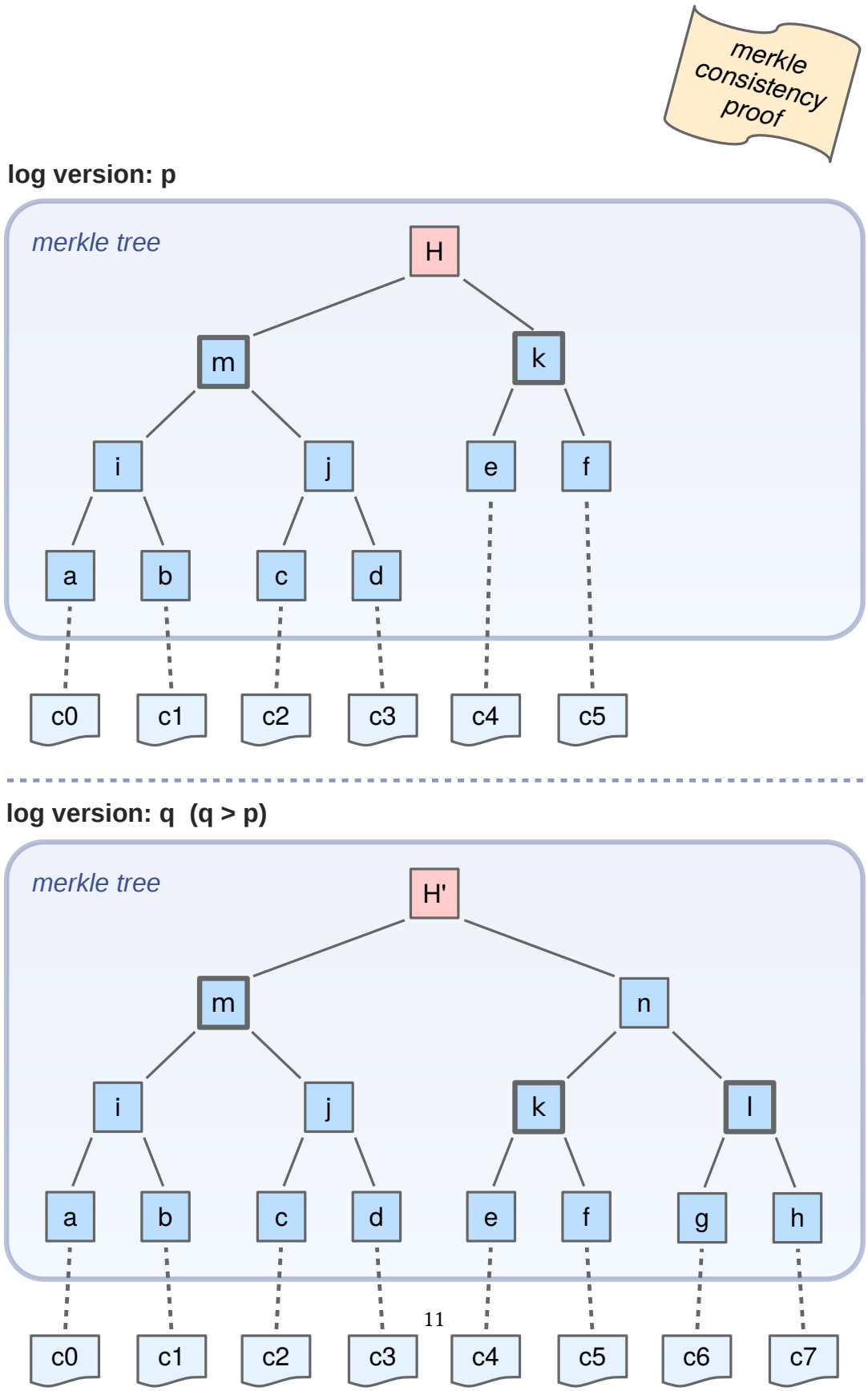

Figure 2.4: *merkle consistency proof*: `PROOF(6, C[8]) = [m, k, l]`. With the nodes `m`, `k` and `l` it is possible to calculate the tree head H of CT log version `p`. Also, it is possible to calculate the tree head H of the newer CT log version `q`.

Consistency Proof

The *merkle consistency proof* provides evidence for the append-only property of a CT log. Append-only means that new entries could only be appended to a log. In the reverse conclusion all entries of a former log version must be the first entries in a later log version. Now, both log versions will be matched to each other and all entries of the former log version must be contained in the later log version in the same order as the first entries before any newly added entries.

Let version p be a former, already proofed log version about m entries, i.e. the tree head is `MTH(C[0:m])`. Let version q be the new log version version about n entries, i.e. with a tree head `MTH(C[n])`. Now, we have to prove if log version q is consistent against log version p. Both log versions need to be identical in its m first entries, which has to be checked now. Therefore, we need a (best minimal) list of nodes in the new log version to be able to calculate both the tree head of the log version q and the tree head of the log version p. If the calculated tree heads match each with the published STHs of both versions, then the new log version q is consistent to the former log version p.

The *consistency proof* for a list of n entries
`C[n] = {c0, c1, ... , cn-1}` of a log version q to a former log version p with m entries and with a tree head `MTH(C[0:m])` is defined as:

```
PROOF(m, C[n]) := SUBPROOF(m, C[n], true)
```

For SUBPROOF is:

```
m = n,  m is the argument value of PROOF:

    SUBPROOF(m, D[m], true) := ()

m = n, else:

    SUBPROOF(m, C[m], false) := {MTH(C[m])}
```

For `m < n` is SUBPROOF recursively defined:
Let k be the biggest power of 2 which is smaller than n (i.e. `k < n <= 2k`).

```
m < n  and  m <= k:

   SUBPROOF(m, C[n], b) := SUBPROOF(m, C[0:k], b) : MTH(C[k:n])
```

```
m < n  and  m > k:

   SUBPROOF(m, C[n], b) := SUBPROOF(m - k, C[k:n], false) : MTH(C[0:k])
```

For example, the *consistency proof* of the merkle tree in figure 2.4 with the log versions p and q can be calculated as:

```
PROOF(6, C[8]) = SUBPROOF(6, C[0:8], True)
= SUBPROOF(2, C[4:8], False) : MTH(C[0:4])
= SUBPROOF(2, C[4:6], False) : MTH(C[6:8]) : MTH(C[0:4])
= {MTH(C[4:6])} : MTH(C[6:8]) : MTH(C[0:4])
= [MTH(C[0:4]) : MTH(C[4:6]) : MTH(C[6:8])]
= [m, k, l]
```

2.2.2 Publishing Certificates into CT Logs

When a certificate is accepted by a CT log it returns as a receipe the *signed certificate timestamp* (SCT). The is a link to the CT log entry of the certificate and is handed over by the webserver to the client browser at the TLS handshake [7]. While the SCT itself is evidence for the client that the certificate is published, the client (theoretically) could verify the log entry as an auditor.

The design of CT respects practical aspects of the deployment and there are different kinds to publish a certificate. The publication could be applied by the CA itself which issues the certificate, the domain owner, or by any other party. The steps for the CA and the domain owner for a website differ for each kind of publication. This also concerns the way how an SCT will be delivered to a client during the TLS handshake.

Issue Certificate without Publication into CT Log

Figure 2.5 shows the issuance of a certificate in the context of the web-PKI. The CA creates a certificate which will be handed over to the domain owner of the webserver. With this certificate (including the corresponding certificate chain), the webserver authenticates himself to a client during the TLS session.

There is no CT log and certificates will not be published. The TLS server does not offer an SCT and no TLS client checks an SCT.

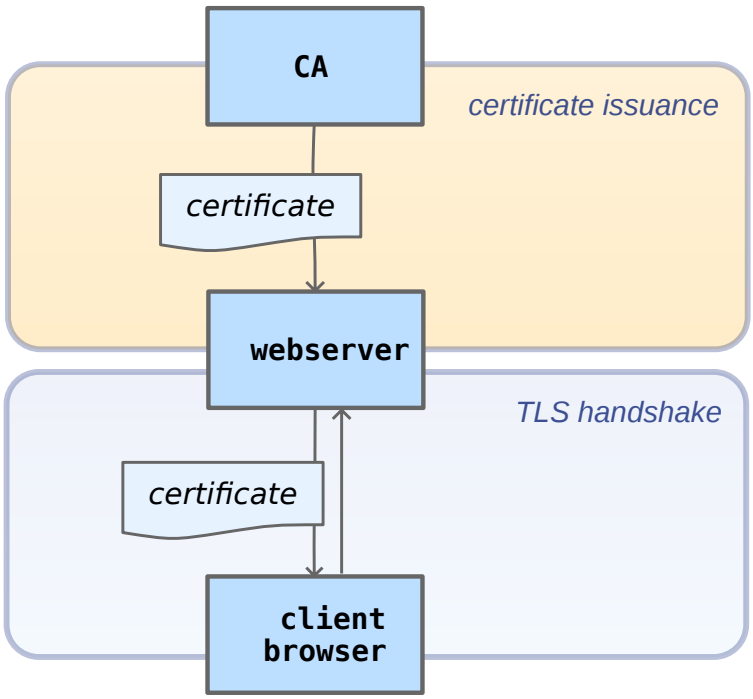

Figure 2.5: Issuance of a webserver certificate and its usage during a TLS handshake; without Certificate Transparency support

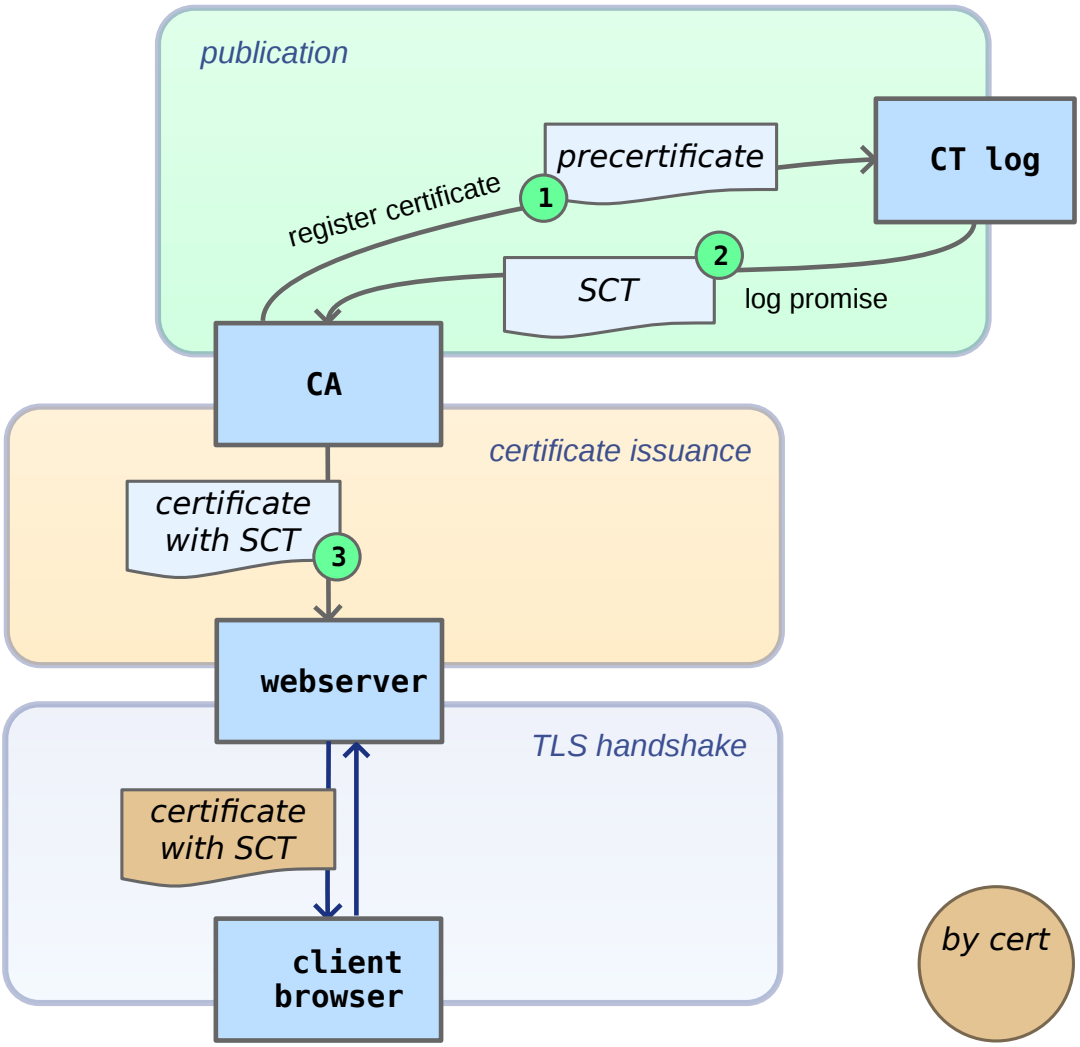

Figure 2.6: Issuance of a webserver certificate and its usage during a TLS handshake; pre-certificate is published instead of the certificate itself and SCT is integreated in the certificate (by X.509v3 extension)

Issue Precertificate

In figure 2.6 the kind of publication is shown where the SCTs are embedded into the certificate itself. The CA needs to adopt its processes on certificate issuance in order to publish a so-called *Precertificate* before of the creation of the final certificate. The *Precertificate* contains all parts of the final certificate except of the SCTs. Also, a *Precertificate* contains a special poison extension (OID 1.3.6.1.4.1.11129.2.4.3) which is critical and prevents the usage of such a certificate on TLS handshakes for authentication. When all SCTs are available the final cert will be created by replacing the poison extension by another X.509v3 extension (OID 1.3.6.1.4.1.11129.2.4.2), steps 1 and 2 in the figure 2.6.

Using this kind of publication the webserver supports CT inherently. The SCTs required by an TLS handshake which supports for CT is inseparable embedded into the certificate and will be automatically handed over to the client.

TLS-Extension

Figure 2.7 shows a kind of issuance, where the domain owner can publish a webserver certificate if the CA does not adopt its certification process for CT publication.

After the certificate issuance the domain owner sends the certificate to an CT log and gets an SCT as a reply. Now the webserver will be configured to hand over the SCT at the TLS handshake via a TLS-Extension next to the certificate.

This is useful for the transitional period, when there are CAs which does not adopt its certificate issuance processes for CT publication. Also, already issued certificates could be published into CT logs afterwards.

Notably is that the publication of (valid) certificates into CT log is applied without any authentication. In order to be accessed by an CT log it is only required that the certificate issued by a CA which has a chain reaching to a root CA accepted by the web-PKI. There is not check who sends the certificate to the CT log.

OCSP-Extension

Also, in the kind of CT publication in figure 2.8 the SCT will be handed over to the client on the TLS handshake next to the certificate. Both, the CA and the webserver needs to adapt its processes. The CA issues the certificate to the domain owner of the webserver and also publishes it into a CT log. The gathered SCT then will be handed over to the webserver within of a OCSP status request. Then, the webserver sends the SCT to the TLS client during a stapled OCSP response of a OCSP request by the client (steps 2 and 3 [orange]).

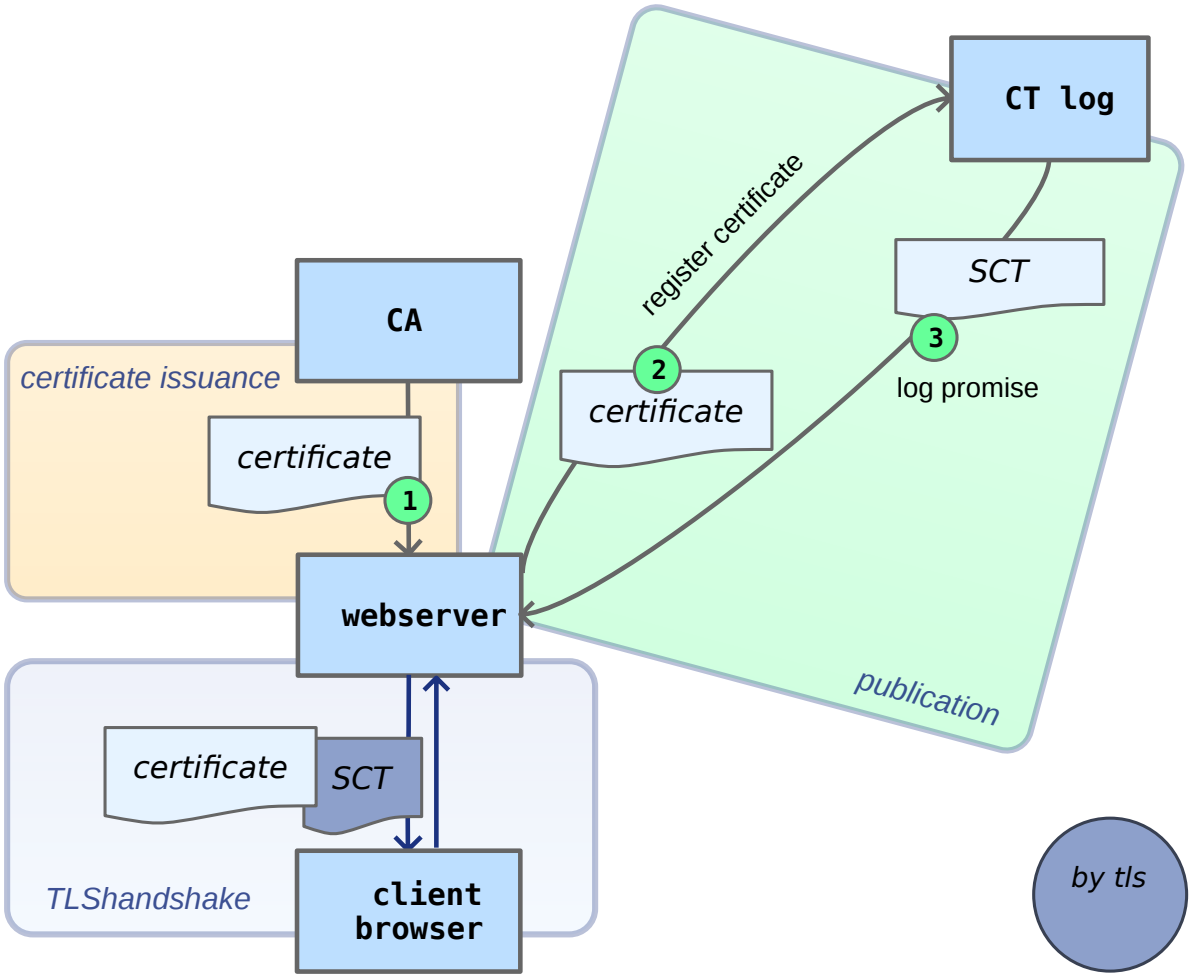

Figure 2.7: Issuance of a webserver certificate and its usage during a TLS handshake; SCT is sent to browser by TLS extension next to the certificate

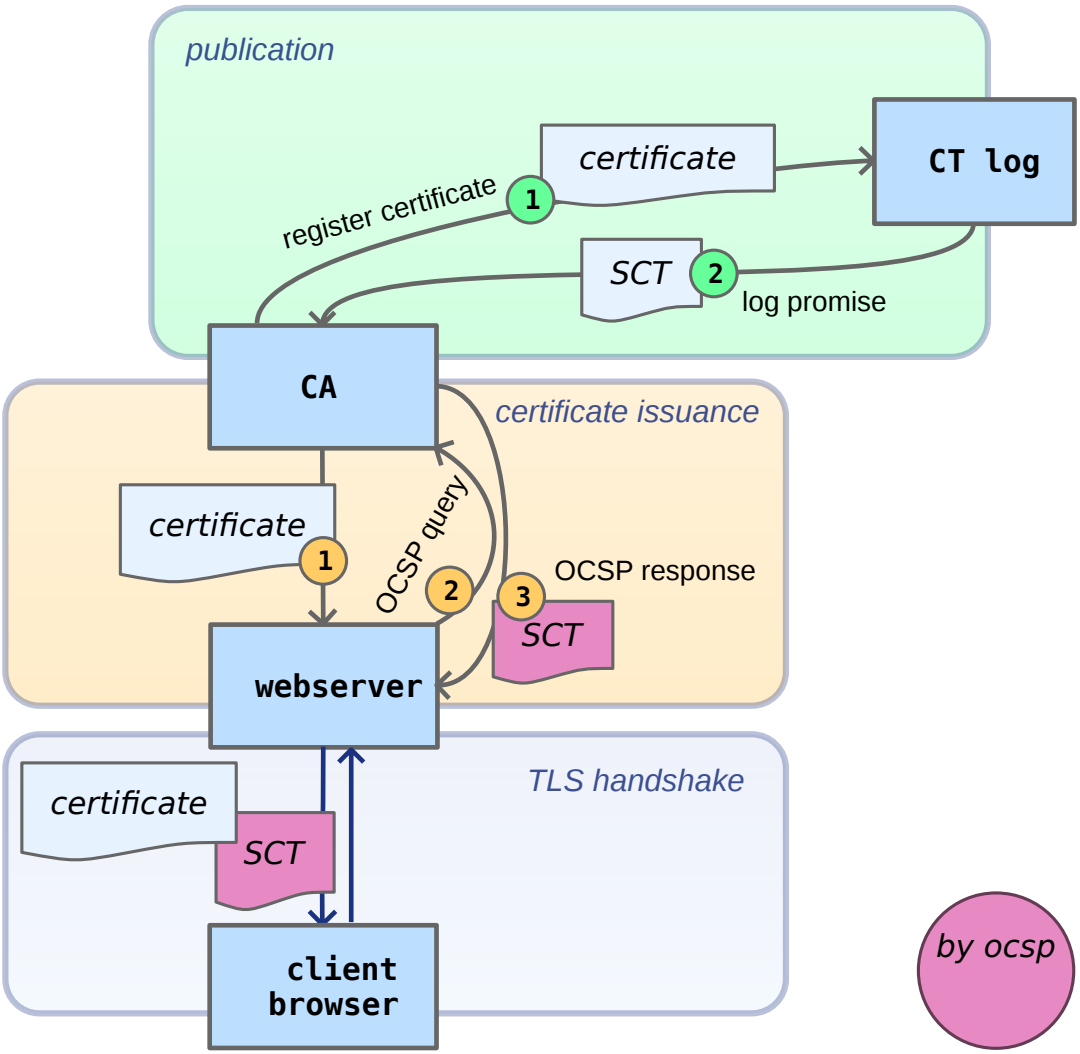

Figure 2.8: Issuance of a webserver certificate and its usage during a TLS handshake; SCT is sent to browser by stapled OCSP response

This kind of publication is suitable if the full certificate issuance and publication into CT logs has to be applied automatically. OCSP was choosen for practical reasons. CAs already support OCSP for revocation status requests.

2.3 Measurement of Certificate Transparency

In [28] was researched for the combination of different perspectives for the web-PKI with the goal to get a complete view of the certificate ecosystem. This research was applied before CT had become mandatory.

Also [3] was conducted before CT was mandatory. Among other new security features to mitigate the risk of certificate misissuance they describe basic properties of CT logs and certificates in the context of CT.

The performance impact at the TLS handshake which comes with CT was analyzed in [14].

3 Webserver Deployment of Certificate Transparency

3.1 Methodology

It will be examined the support or CT on TLS encrypted website requests.

On the examination three Steps will be applied: (1) Selection of the domain names of the websites; (2) Saving of the certificates gathered on the TLS handshakes with the webservers; (3) Determination and saving of the SCTs accompanied by the TLS handshakes.

(1) Selection of the domain names of the websites In order to measure the deployment of CT on HTTPS websites a set of domain names is required. Therefore, the Alexa Global Top 1M list [2] will be taken. This list contains 1 million domains names ordered by its popularity starting with the most popular domain as the first entry. This makes it possible to map the deployment of CT to the popularity of domain names. Other web surveys also use the Alexa Global Top 1M list [22], for example [30].

(2) Saving of the certificate gathered on the TLS handshake with the webservers The domain names will be mapped to the TLS certificates. In an intermediate step the domain names will be mapped to IP addresses by DNS resolver requests. It is kept unconsidered the possibility to get different IP addresses a domain address when asking different DNS resolver. Also, the possibility that in Content Delivery Networks hosted websites could use different webserver certificates will not be considered. This keeps open for further analyses.

For every entry of the Alexa Global Top 1M list as a domain name the entry itself will be taken and the entry applied by a *'www.'* prefix. In order to gather the certificates of the webserver, the webpages will be requested via HTTPS. If the webserver supports HTTPS, the certificate will be transmitted during the TLS handshake.

(3) Determination and saving of the SCTs accompanied by the TLS handshakes SCTs could be passed during the TLS handshake via three mechanisms, by-cert, by-TLS-extension, and by-OCSP-response. This mechanisms are discussed in 2.2.2. For every TLS handshake all three mechanisms will be tried to use in order to fetch all SCTs which are provided together with a webserver certificate.

3.2 Implementation

For the TLS handshake, in order to fetch the SCTs, and to read out the data from the SCTs, the author developed a software component named ctutlz [27] which is hosted as an open source project at github. A second component uses the library functions provided by ctutlz and runs the examination process and creates the analysis of the gathered data.

Both components are written in Python. To accomplish for the task to support CT a good maintained OpenSSL wrapper library can be used [19]. In order to draw figures the python software matplotlib [25] will be utilized. The author is familiar in coding with python which was not the last reason to chose for this programming language.

The implementation of the code to fetch the SCTs was a complex task burden with many false attempts. This approaches did not work:

- The module *ssl* of the standard python library [26] does not support OCSP-requests nor CT.
- m2crypto, a comprehensive python crypto library [12], does not support CT.
- pyOpenSSL [19], the most complete OpenSSL wrapper and crypto library in python does not support all CT functionalities required to gather the SCTs during the TLS handshakes. The same is for the module *cryptography* [18] which is used by pyOpenSSL to call OpenSSL callback functions. Also, there is lack in the functionality for the verification of SCTs.
- OpenSSL [17] supports CT and the functionality to gather the SCTs in all three possible ways (by-cert, by-TLS, by-OCSP). But an implementation which uses OpenSSL as a command in an own process out of the python code was too slow. And it would be necessary to apply several command calls of `openssl s_client` for a domain to be able to gather the SCTs in all three possible kinds.

3.3 Results

The SCTs of the domains of the Alexa Global Top 1M list [2] have been gathered two times, on 2017-09-28, and one year later on 2018-10-08. Certificates issued after 2018-04-30 must be ct-logged in order to comply for CT Chrome Policy [15, 4]. So an increase of CT support in general in the results of 2018 is to be expected.

3.3.1 TLS Handshake Tries

Table 3.1: TLS Handshake Tries (w/wo www. prefix)

	2017-09-28		2018-10-08	
	count	percent	percent	count
all	2,000,000	100.00	100.00	2,000,000
timeout	223,868	11.19	8.19	163,870
no certificate	446,313	22.32	20.48	409,551
certificate (no EV)	1,283,926	64.20	69.78	1,395,582
EV certificate	45,893	2.29	1.55	30,997

Table 3.1 lists the statistical results of the TLS handshakes tries on 2017-09-28 and on 2018-10-08 of the domains with and without 'www' prefix of the Alexa Global Top 1M list. The number of timeouts have been declined by 3% to about 8.2%. The proportion of tries where no TLS connection could be established declined by a fifth to about 20%. The number of extended validation certificates also have been declined by a fifth to about 1.6%, but the number of non-ev certificates increased by nearly 112,000 to about 70%.

Table 3.2: TLS Handshake Tries on 2018-10-08

	without 'www' prefix		with 'www' prefix	
	count	percent	percent	count
all	1,000,000	100.00	100.00	1,000,000
timeout	85,769	8.58	7.81	78,101
no certificate	203,619	20.36	20.59	205,932
certificate (no EV)	696,273	69.63	69.93	699,309
EV certificate	14,339	1.43	1.67	16,658

Table 3.2 compares the statistical results of the handshakes tries on 2018-10-08 of the domains of the Alexa Global Top 1M list with 'www' prefix and without 'www' prefix. Notably is that the proportion of timeouts of domains without 'www' prefix with about 8.6% is nearly 10% higher than the proportion of timeouts of domains with 'www' prefix. In both cases, the proportion of timeouts is nearly constant in the complete range of the alexa ranking. The proportions where the server answered but sent no TLS server certificate ('no certificate', i.e. HTTP only / HTTPS not supported) are nearly the same. Even the proportions of TLS-handshakes with

EV or non-EV certificates do not differ significantly. The following graphs and diagrams only for on type of domain, with or without (w/wo) 'www' prefix would not show meaningful differences. Therefore, the following results are about both types of domains.

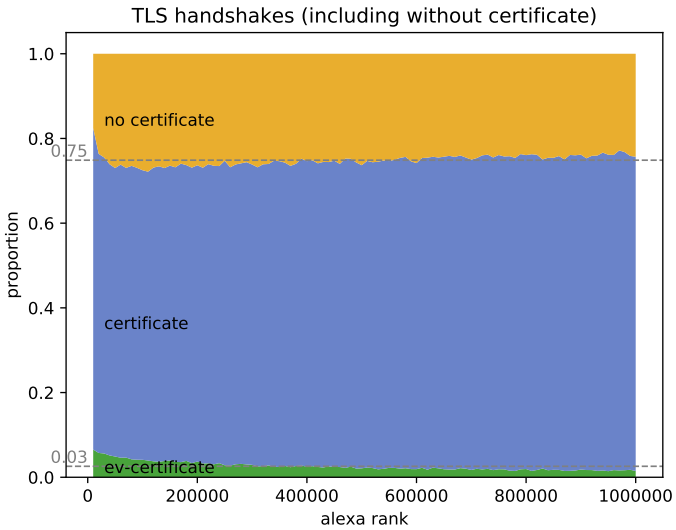

Figure 3.1: TLS handshake tries (alexa ranked) on 2017-09-28

The distribution of the results of the TLS handshake tries by decreasing popularitiy i.e. rising alexa rank shows figure 3.1 for 2017-09-28 and figure 3.2 for 2018-10-08. Both plots are alike but the number of outliers in the results have increased in 2018. The distribution is relatively stable in the complete alexa range. Both, EV and non-EV certificates are more popular for the most popular domains. While starting with an alexa rank of about 20,000 in 2017 a continually light increase with growing alexa rank could be registered. This increase has nearly vanished in 2018 and lays behind the amount of outliers. In 2017 and in 2018 the proportion of EV certificates continually decreases for less popular domains.

3.3.2 Signed Certficate Timestamps

Table 3.3: SCTs by deliver way (w/wo www prefix)

	2017-09-28		2018-10-08	
	count	percent	percent	count
all	1,038,227	100.00	1,906,975	100.00

Table 3.3: SCTs by deliver way (w/wo www prefix)

	2017-09-28		2018-10-08	
	count	percent	percent	count
by-cert	865,852	83.40	1,791,759	93.96
by-tls-extension	172,099	16.58	114,650	6.01
by-ocsp-response	276	0.03	566	0.03

Table 3.3 shows that in the last 12 month the sum of SCTs was nearly doubled. The Proportion of SCTs delivered by-cert now dominates with nearly 94%. The number of SCTs by-tls-extension decreased not only in proportion but also absolute by nearly 60,000 SCTs. While the number of SCTs delivered by stapled OCSP response have more than doubled they still are insignificant with a proportion of 0.03%.

Figure 3.3 showes that in 2017 the proportion of SCTs by-tls has its highest highest value on most popular domains and continually decreases till an alexa-rank of about 400,000. Then constant it remains relatively constant for the higher alexa ranks.

In 2018 this curve has completely diminished as shown in figure 3.4. The proportion of SCTs by-certificate were completely higher than in 2017-09-28 among a fewer spikes. The lowest proportion of SCT by-certificate in 2018 could be seen on the least popular domains.

For the spikes the author has no explanation; maybe they were related to temporary loss of internet connectivity during the measurement.

Table 3.4: Certificates with or without SCTs (w/wo www prefix)

	2017-09-28		2018-10-08	
	count	percent	percent	count
all	1,329,819	100.00	100.00	1,426,577
no SCTs	990,221	74.46	42.49	606,086
1 or more SCTs	339,598	25.54	57.51	820,491
1 SCT	53	0.00	0.00	9
2 SCTs	111,717	8.40	43.10	614,866
3 SCTs	130,358	9.80	10.88	155,205
4 SCTs	63,970	4.81	2.90	41,399
5 or more SCTs	33,500	2.52	0.63	9,012

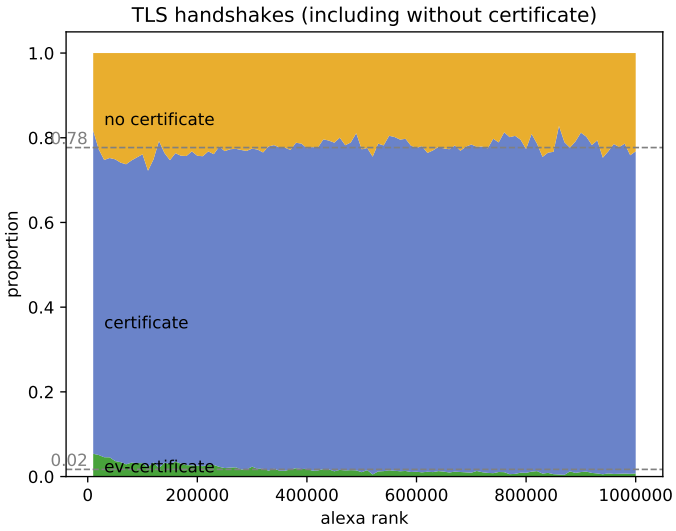

Figure 3.2: TLS handshake tries (alexa ranked) on 2018-10-08

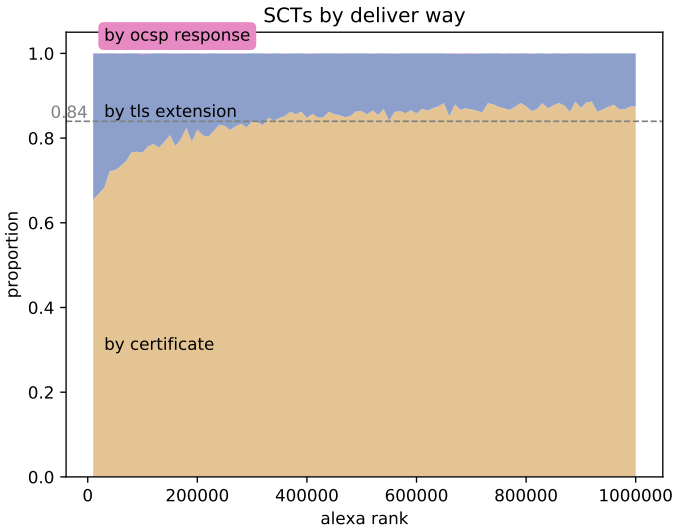

Figure 3.3: SCTs alexa ranked by Deliver Way on 2017-09-28

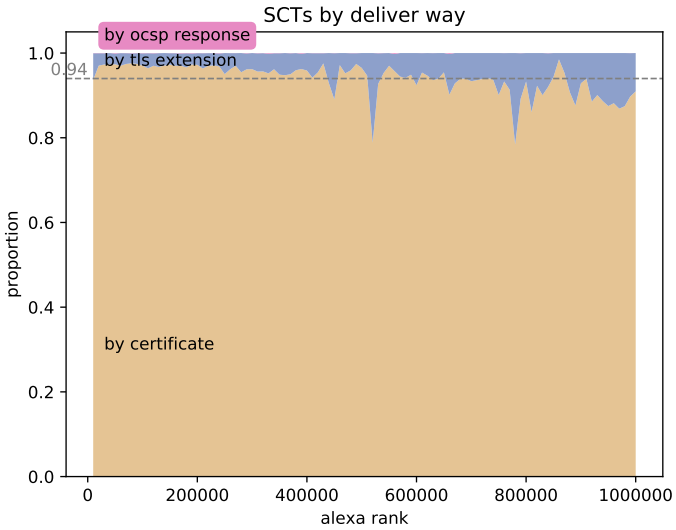

Figure 3.4: SCTs alexa ranked by Deliver Way on 2018-10-08

Table 3.4 showes that in 2018 more than 57% of certificates are delivered with SCTs As in the last year three was the most often occurrence of SCTs accompanied by a certificate at the TLS handshake the most certificate (43%) now come with two SCTs. Nearly 11% come with three SCTs, nearly 3% with four SCTs, five or more SCTs are negligible. Two SCTs seems to become the usable case of CT-supported certificates.

Figure 3.5 showes the proportions of the number of SCTs accompanied by a certificate in 2017, figure 3.6 in 2018. The comparison of the both figures show the increase of webserver which support CT. While the proportions are relatively stable for the less popular domains. The most popular domains have an increased proportion for CT support. But till now the top most domains still have a proportion of about 30% which do not support CT now. This is a relative high value.

3.3.3 CT Logs

The proportions of SCTs by CT log are shown in figure 3.7 for 2017 and in figure 3.8 for 2018. While in 2017 about 70% of SCTs came from only three CT logs, now the SCTs are distributed more even on a bigger number of CT logs.

Also – apart from Google – the distribution of the total number of SCTs by CT log operators is more even distributed in 2018 as shown in figure 3.10 than in 2017 which is shown in figure 3.9.

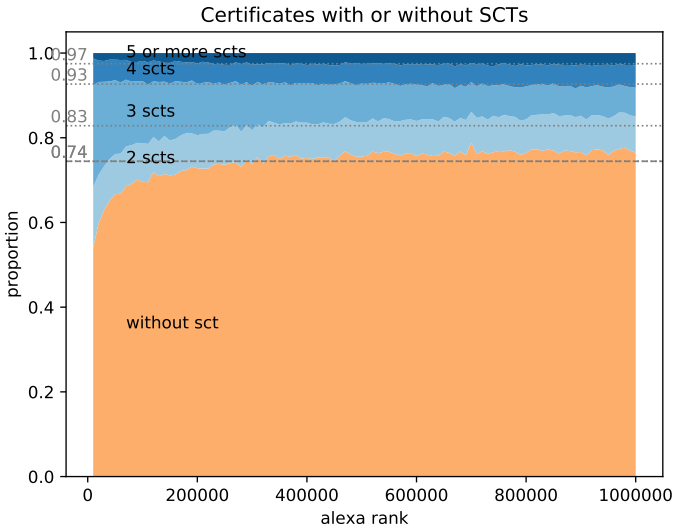

Figure 3.5: Certificates with or without SCTs (alexa ranked) 2017-09-28

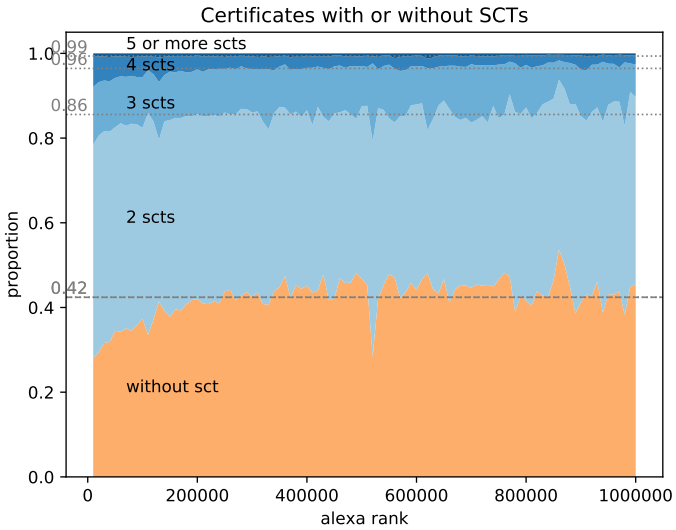

Figure 3.6: Certificates with or without SCTs (alexa ranked) 2018-10-08

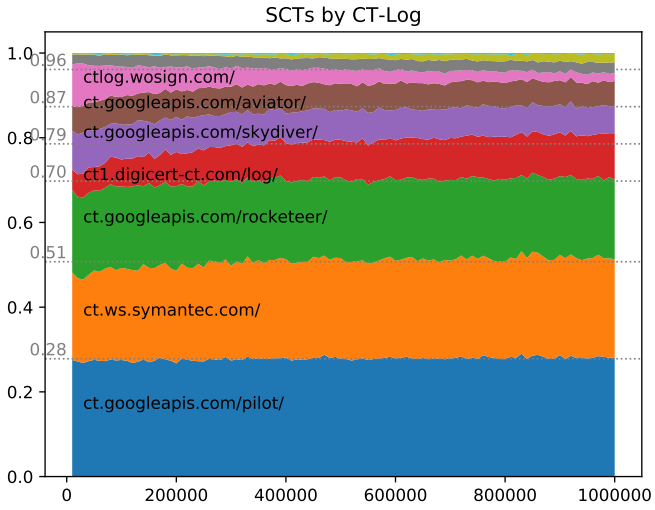

Figure 3.7: SCTs by CT log (alexa ranked) 2017-09-28

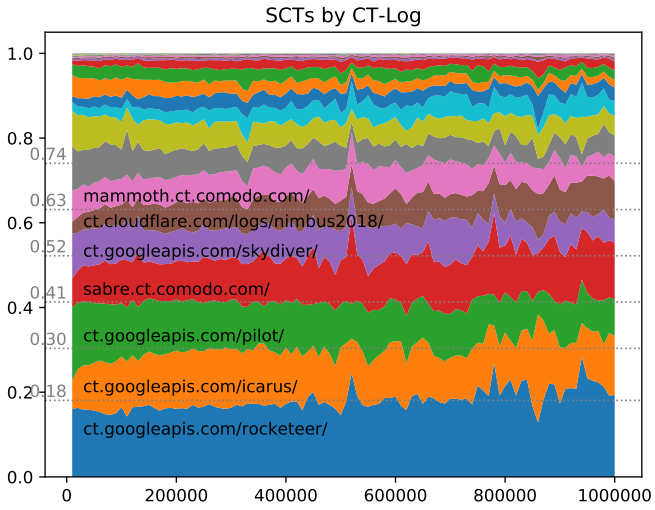

Figure 3.8: SCTs by CT log (alexa ranked) 2018-10-08

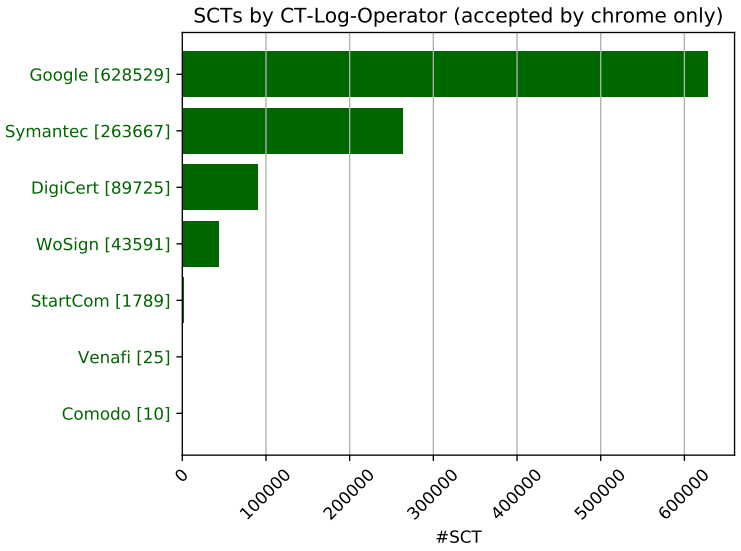

Figure 3.9: SCTs by CT log Operator 2017-09-28

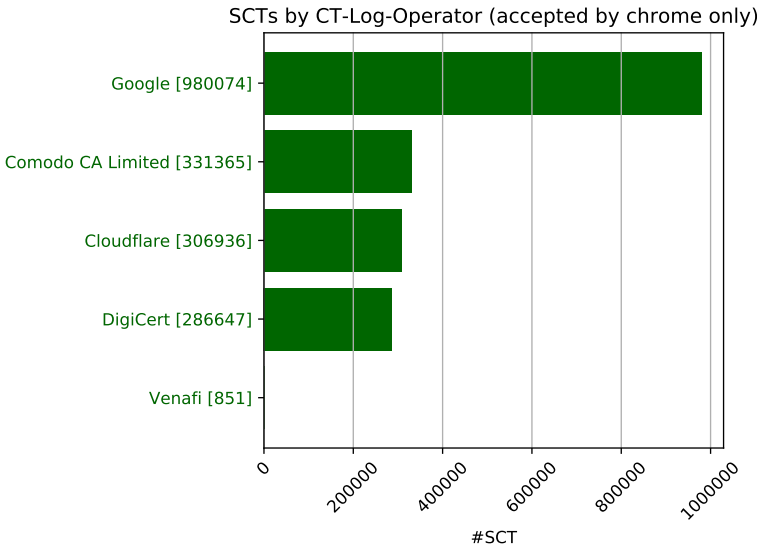

Figure 3.10: SCTs by CT log Operator 2018-10-08

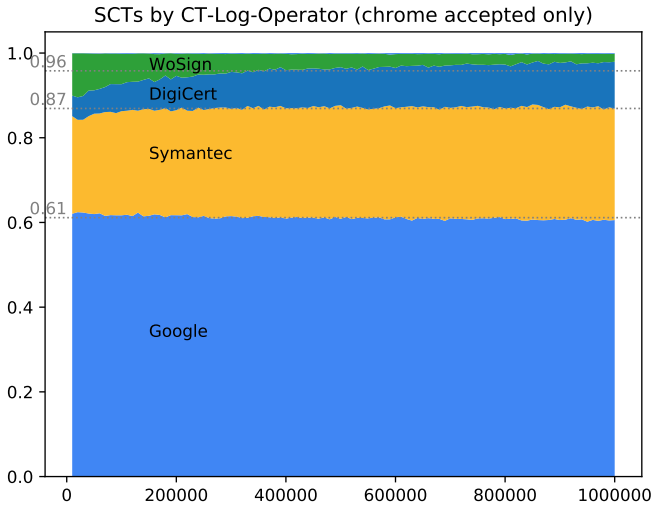

Figure 3.11: SCTs by CT log (alexa ranked) 2017-09-28

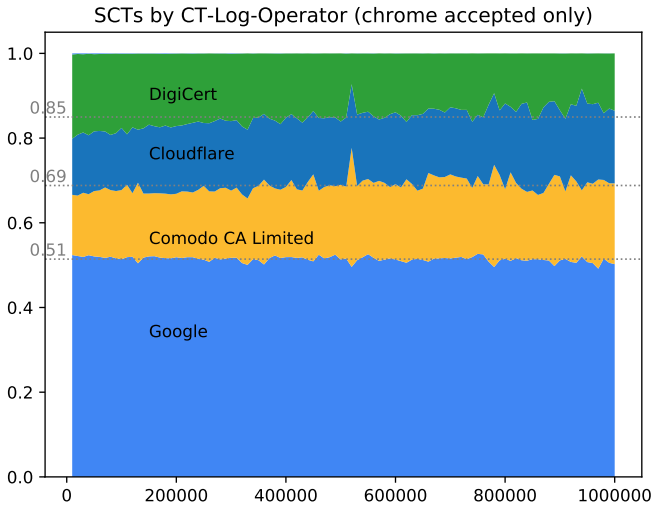

Figure 3.12: SCTs by CT log (alexa ranked) 2018-10-08

The reason why Google dominates here is due to the demand of the Chrome Policy [5] that at least two SCTs must be accompanied by a certificate and at least one SCT must come from a Google CT log (and also at least one SCT must belong to a CT log not from Google). This correlates well with the proportions in figure 3.11 for 2017 and figure 3.12 for 2018. The falling of Google from 61% to 50% of proportion correlates with the 'trend' to use only two SCTs instead of three or more.

4 CT Log Evolution

The investigation of the CT log evolution was applied in the context of the paper *The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem* [21]. This paper was accepted by the ACM Internet Measurement Conference 2018 in Boston (https://conferences.sigcomm.org/imc/2018/).

4.1 Methodology

It will be examined the evolution of the CT logs. Therefore, the rate of increase for all Chrome accepted CT logs will be analyzed for Precertificate entries only. CT log entries of final certificates could be added to a CT log by any party. Precertificate entries can be added by the CAs itself. The examination includes the steps *(1) Gather data sets of CT log entries*; *(2) Save Precertificate entries only*; *(3) Apply statistical analysis on the Precertificate entries.*

(1) Gather data sets of CT log entries It is only possible to download certificate entries from CT logs by index. The API does not provide a mechanism to filter for attributes such as entry type or domain name. So, for the analysis of Precertificate entries all entries for a CT log needs to be downloaded.

(2) Save Precertificate entries only Only the CT log entries will be stored into a database for further analyses.

(3) Apply statistical analysis on the Precertificate entries In order to investigate the CT log evolution several statistical evaluations will be applied on the saved Precertificate entries. An update rate will be calculated which is the average of the number of logged Precertificates for a CA for the days only when logging was applied by the CA.

4.2 Implementation

The dataset of all CT log entries accepted by the Google Chrome browser was created by the working group Lehrstuhl fuer Netzarchitekturen und Netzdienste at the Technische Universitaet Muenchen (TUM) using Bro [29]. It already contained of Postgres Database exports sorted by Percertificate entries and final certificate entries for each CT log.

The original data set of all CT log entries as uncompressed JSON files till April 2018 which was also gathered by the author is of more than 6TB of data.

For the statistical analysis the author extended the software for the statistical analysis of the webserver deployment of CT.

4.3 Results

Table 4.1: Precert Entries by CA; CAs with the 10 most logged certificates

	date of first entry	update rate	day maximum	day max date
Let's Encrypt	2018-03-29	2,239,669.93	2,948,832	2018-04-17
DigiCert	2016-04-30	19,763.89	244,816	2017-12-04
Comodo	2016-05-03	12,836.65	442,491	2018-04-03
GlobalSign	2016-06-30	2,917.11	17,612	2018-02-04
StartCom	2017-01-10	1,823.19	5,757	2016-09-09
Western Digital	2018-04-11	85,416.14	309,339	2018-04-19
Go Daddy	2014-12-23	425.96	57,392	2018-04-11
Entrust	2014-07-21	250.10	7,917	2018-04-17
Cybertrust JP	2015-01-07	225.08	7,020	2015-10-21
Hostpoint AG	2016-03-01	190.64	3,112	2016-07-04

When the experimental RFC 6962 [11] was released on June 2013 only a few number of CT log entries was made, mostly for testing purposes. But there were opposite views for a public register of webserver certificates, mainly for privacy and security reasons as of to protect business interests [21].

The first of the top-most logging CAs started to publish into CT logs in July 2014 as shown in table 4.1. Let's Encrypt was the least CA of the top 10 which started to log but immediately dominates the number of entries with an update rate of more than 2.2M certificates per day.

The domination of Let's Encrypt is also visible in figure 4.2 which shows the relative update rate of the top 10 CAs. Before CT had become mandatory in April 2018, the most CT log publications were made by DigiCert.

Table 4.2: Precert Entries by CA in April 2018 from 2018-04-01 till 2018-04-26

	average	update rate	sum	number of days logged
Let's Encrypt	2,291,499.04	2,291,499.04	59,578,975	26
DigiCert	96,008.38	99,848.72	2,496,218	25
Comodo	328,337.46	328,337.46	8,536,774	26
GlobalSign	6,072.81	6,315.72	157,893	25
StartCom	0.00	0.00	0	0
Western Digital	45,993.31	85,416.14	1,195,826	14
Go Daddy	13,914.12	14,470.68	361,767	25
Entrust	4,356.65	4,530.92	113,273	25
Cybertrust JP	235.62	291.71	6,126	21
Hostpoint AG	0.00	0.00	0	0

As we can see in figure 4.1 before of April 2018 the number of cumulated CT log entries grew relatively constantly and doubled about every one or two years. Then, when CT support was mandatory for webserver certificates in Chrome [24, 16] in April, all active CAs (i.e. not StartCom, nor Hostpoint AG) logged with an higher update rate. But Let's Encrypt which started in the end of March immediately outnumbers the rates of all other CAs added together. This is also visible in Table 4.2 by comparing the sum of publications made by Let's Encrypt in April 2018 with the sums of the other CAs.

The heatmap in figure 4.3 shows for each CA the CT logs where the most Precertificates are logged. Not only Let's Encrypt but also DigiCert and Comodo are logging mostly into a small set of CT logs.

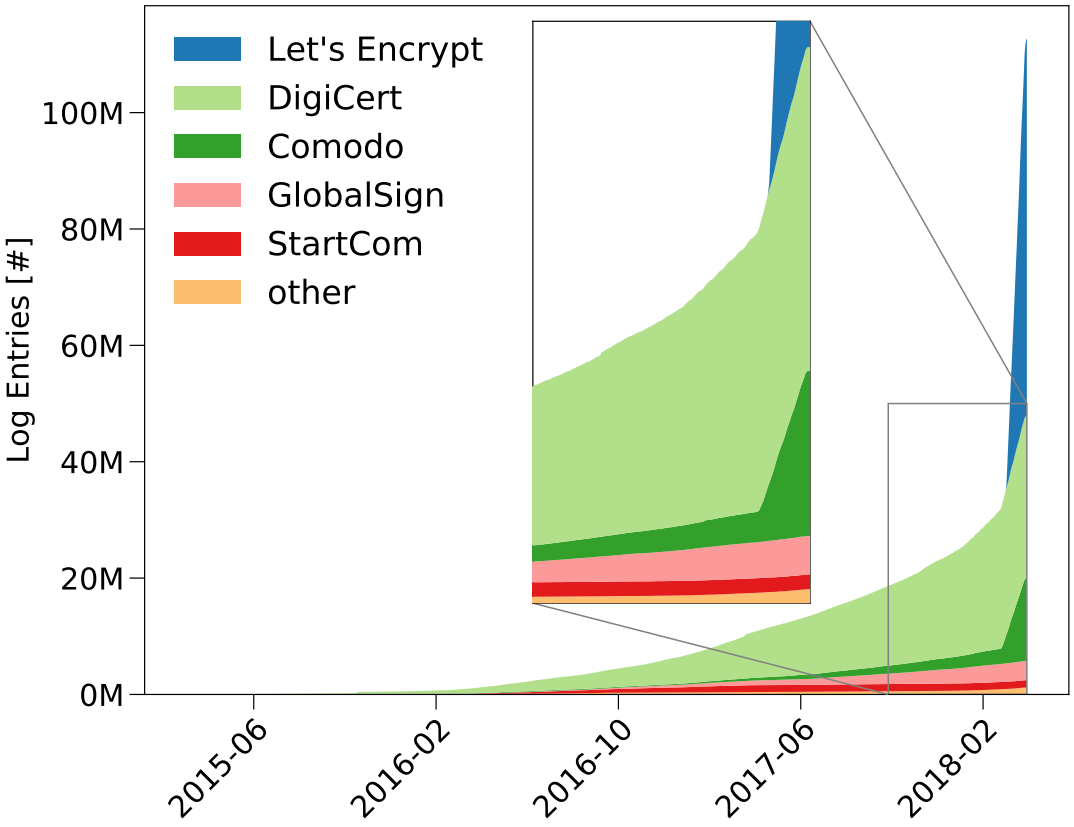

Figure 4.1: Cumulative growth of logged precertificates by Certification Authority (CA)

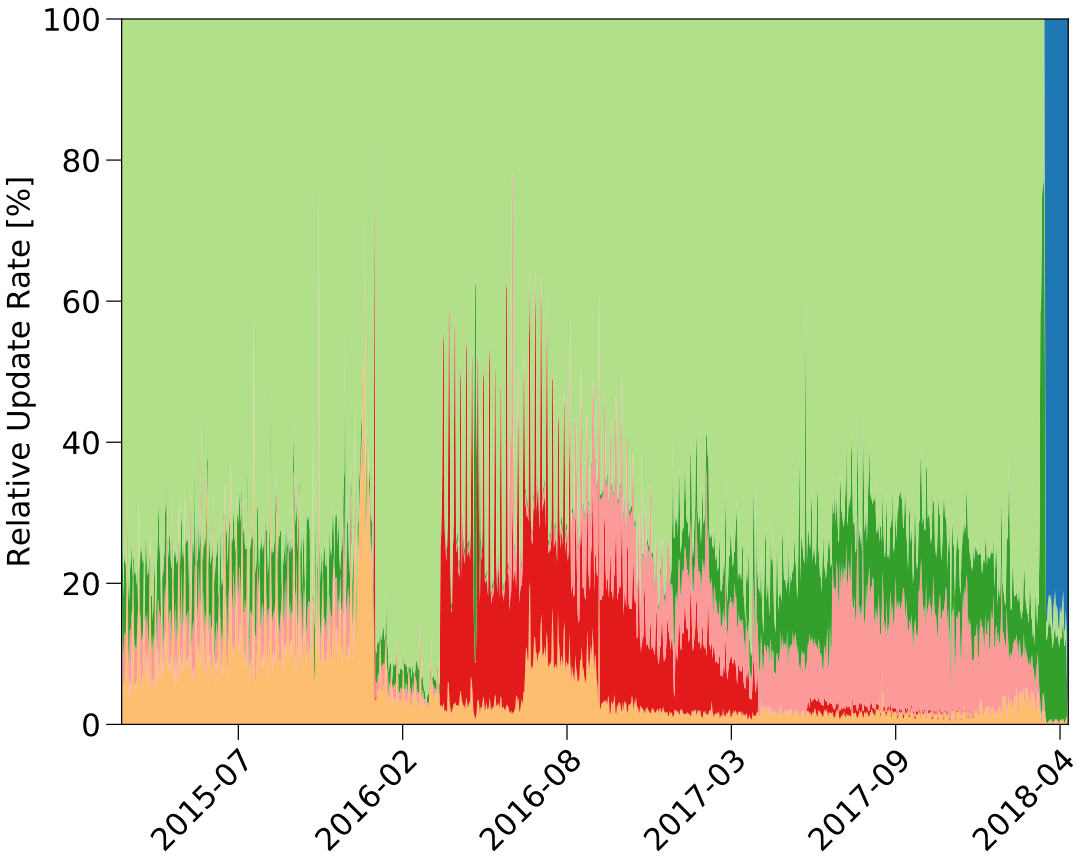

Figure 4.2: Relative update rate per CA and day. Let's Encrypt dominates after starting to log.

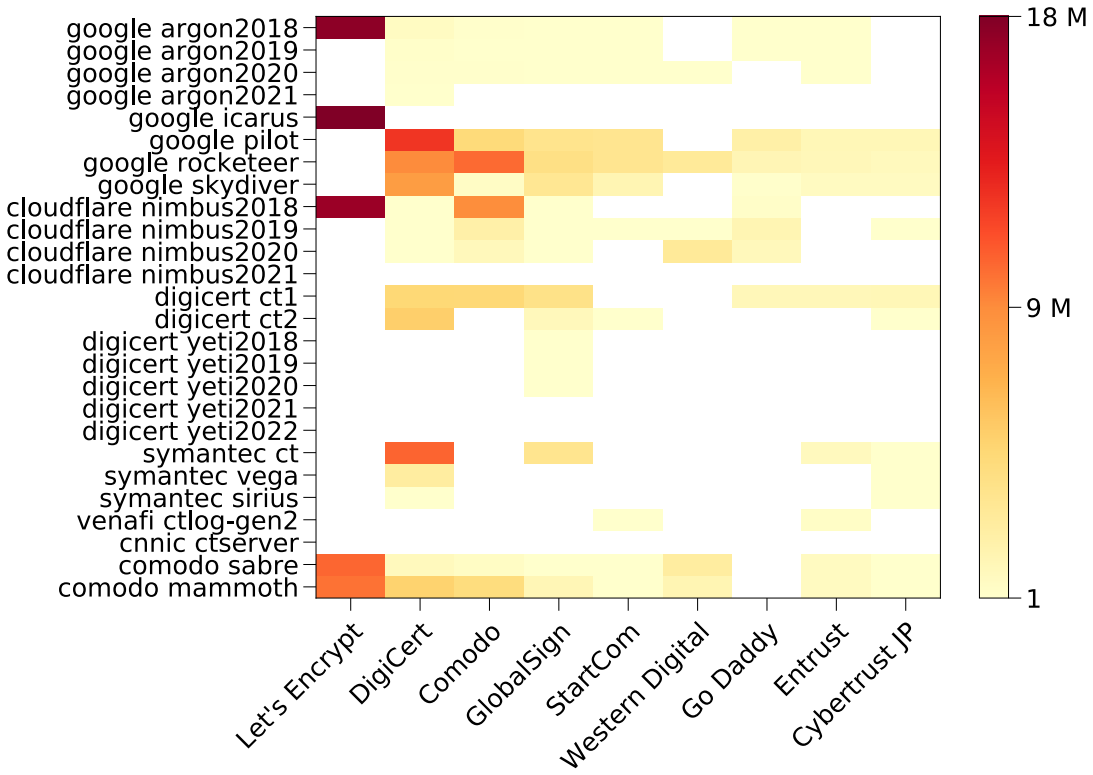

Figure 4.3: Distribution of precertificate logging by CAs over different CT logs for April 2018

5 Conclusion and Outlook

5.1 Conclusion

The TLS ecosystem is based on the trust of users in the CAs to correctly issue certificates. CT adds a control element by making webserver certificates used by HTTPS website requests auditable in a public manner.

During the last twelfe month the deployment of CT has achieved a relative prosper development. About 75% of the HTTPS enabled websites support CT. But there is still a relative big proportion of 30% of the top domain names which do not support CT till now.

Currently, nearly all CAs are publishing certificates on its issuance into CT logs. So more than 90% of the delivered SCTs during the TLS handshakes with the most popular domains are embedded in the certificates itself which means that corresponding Precertificates were published into CT logs. From this it follows that a nearly complete view of the web-PKI is now possible by analyzing Precertificate entries of CT logs.

The distribution of CT logs has been improved on the one side. While twelve month ago about 70% of the SCTs of the most popular domains came from only three CT logs, now they are distributed more diverse into seven CT logs. On the other side, due to the massive number of published Precertificates by Let's Encrypt into two CT logs only, the distribution of CT log entries over all CT logs currently is highly concentrated.

5.2 Outlook

For a domain owner CT makes it possible to detect misissuance for its domain. He has to become active himself to use a monitor in order to achieve this. While a big proportion of domain owner probably would not set up for a CT log monitoring it would be desirable if there could be a systematic approach to automatically detect such an misissuance. For example, CT log monitoring could be enhanced by matching the issuing CA with Certificate Authority Authorization (CAA) rules set in the Domain Name System.

After the exponential growth of the number of CT log entries in April 2018 a new measuring of the CT log evolution would show more interesting developments and is therefore desirable.

This is one of the next tasks to be achieved. The author already extended ctutlz for the download and parsing of CT log entries (sub-module `ctutlz/rfc6962.py`).

Bibliography

[1] D. Eastlake 3rd and T. Hansen. US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). RFC 6234, IETF, May 2011.

[2] Alexa. Top 1M sites. https://www.alexa.com/topsites. http://s3.dualstack.us-east-1.amazonaws.com/alexa-static/top-1m.csv.zip, accessed on 26. October 2018.

[3] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. Mission Accomplished? HTTPS Security after DigiNotar. In *IMC*, 2017.

[4] Chromium. Certificate Transparency in Chrome. https://github.com/chromium/ct-policy/blob/master/ct_policy.md, accessed on 28. October 2018.

[5] Chromium. Certificate Transparency in Chrome. https://github.com/chromium/ct-policy/blob/master/ct_policy.md, 2018.

[6] COMODO CA. crt.sh. https://crt.sh/, accessed on 31. October 2018.

[7] Michael Driscoll. The Illustrated TLS Connection. https://tls.ulfheim.net/, accessed on 30. October 2018.

[8] Google. Certificate Transparency. https://www.certificate-transparency.org/, accessed on 30. October 2018.

[9] Google. Certificate Transparency - Known Logs. https://www.certificate-transparency.org/known-logs, accessed on 30. October 2018.

[10] Hans Hoogstraaten and others, Fox-IT BV. Black Tulip: Report of the investigation into the DigiNotar Certificate Authority breach. http://heise.de/-1741726, accessed on 31. October 2018.

[11] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. RFC 6962, IETF, June 2013.

[12] Matej Cepl. M2Crypto. https://gitlab.com/m2crypto/m2crypto/blob/master/README.rst, accessed on 31. October 2018.

[13] Merkle, R. A digital signature based on a conventional encryption function. In Proceedings of CRYPTO, pages 369 - 378. Springer, 1988.

[14] Carl Nykvist, Linus Sjöström, Josef Gustafsson, and Niklas Carlsson. Server-Side Adoption of Certificate Transparency. In *PAM*. Springer, 2018.

[15] Devon O'Brien. Certificate Transparency Enforcement in Google Chrome. https://groups.google.com/a/chromium.org/forum/#!msg/ct-policy/Qqr59r6yn1A/2t0bWblZBgAJ, accessed on 28. October 2018.

[16] Devon O'Brien. Certificate Transparency Enforcement in Google Chrome. https://groups.google.com/a/chromium.org/forum/#!msg/ct-policy/wHILiYf31DE, 2018.

[17] OpenSSL Software Foundation. OpenSSL. https://www.openssl.org/, Abrufdatum: 12. November.

[18] Python Cryptographic Authority (pyca). pyca/cryptography. https://github.com/pyca/cryptography, accessed on 31. October 2018.

[19] Python Cryptographic Authority (pyca). pyOpenSSL - github-Repository. https://github.com/pyca/pyopenssl, accessed on 31. October 2018.

[20] Ronald Eikenberg, heise online. Protokoll eines Verbrechens: DigiNotar-Einbruch weitgehend aufgeklärt. http://heise.de/-1741726, accessed on 31. October 2018.

[21] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, and Matthias Wählisch. The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. In *Proc. of ACM Internet Measurement Conference (IMC 2018)*, New York, NY, USA, October 2018. ACM. accepted for publication.

[22] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D. Strowes, and Narseo Vallina-Rodriguez. A long way to the top: Significance, structure, and stability of internet top lists. *CoRR*, abs/1805.11506, 2018.

[23] Seth Schoen and Eva Galperin. Iranian Man-in-the-Middle Attack Against Google Demonstrates Dangerous Weakness of Certificate Authorities. https://www.eff.org/deeplinks/2011/08/iranian-man-middle-attack-against-google, accessed on 31. October 2018.

[24] Ryan Sleevi. Certificate Transparency in Chrome - Change to Enforcement Date. https://groups.google.com/a/chromium.org/forum/#!msg/ct-policy/sz_3W_xKBNY, 2017.

[25] The Matplotlib development team. https://matplotlib.org/, accessed on 31. October 2018.

[26] The Python Standard Library. ssl – TLS/SSL wrapper for socket objects. https://docs.python.org/3/library/ssl.html, accessed on 31. October 2018.

[27] Theodor Nolte. ctutlz. https://github.com/theno/ctutlz, accessed on 31. October 2018.

[28] Benjamin VanderSloot, Johanna Amann, Matthew Bernhard, Zakir Durumeric, Michael Bailey, and J Alex Halderman. Towards a Complete View of the Certificate Ecosystem. In *IMC*. ACM, 2016.

[29] Vern Paxson and Robin Sommer. The Bro Network Security Monitor. https://www.bro.org/, accessed on 31. October 2018.

[30] Matthias Wählisch, Robert Schmidt, Thomas C. Schmidt, Olaf Maennel, Steve Uhlig, and Gareth Tyson. RiPKI: The Tragic Story of RPKI Deployment in the Web Ecosystem. In *Proc. of 14th ACM Workshop on Hot Topics in Networks (HotNets)*, pages 11:1–11:7, New York, Nov. 2015. ACM.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 1. November 2018 Theodor Nolte