Server-side Adoption of Certificate Transparency

Carl Nykvist, Linus Sjöström, Josef Gustafsson, and Niklas Carlsson

Linköping University, Sweden

Abstract. Certificate Transparency (CT) was developed to mitigate shortcomings in the TLS/SSL landscape and to assess the trustworthiness of Certificate Authorities (CAs) and the certificates they create. With CT, certificates should be logged in public, audible, append-only CT logs and servers should provide clients (browsers) evidence, in the form of Signed Certificate Timestamps (SCTs), that the certificates that they present have been logged in credible CT logs. These SCTs can be delivered using three different methods: (i) X.509v3 extension, (ii) TLS extension, and (iii) OSCP stapling. In this paper, we develop a client-side measurement tool that implements all three methods and use the tool to analyze the SCT adoption among the one-million most popular web domains. Using two snapshots (from May and Oct. 2017), we answer a wide range of questions related to the delivery choices made by different domains, identify differences in the certificates used by these domains, the CT logs they use, and characterize the overheads and potential performance impact of the SCT delivery methods. By highlighting some of the tradeoffs between the methods and differences in the websites selecting them, we provide insights into the current SCT adoption status and differences in how domains have gone upon adopting this new technology.

1 Introduction

The majority of the internet traffic is delivered using HTTPS and encrypted using Transport Layer Security (TLS). While most of these connections use relatively strong security algorithms [20], one of the major weaknesses in securing the end-to-end communication is instead the amount of trust that is placed in the Certificate Authorities (CAs) that generate the X.509 certificates (connecting public keys to servers/domains) needed for us (and our browsers) to trust that the servers/domains that we communicate with are who they claim to be [8,15].

Browsers typically trust that the private key associated with the public key inside a certificate belongs to a given server/domain as long as (i) the certificate is signed by a CA (or an organization that a CA has delegated trust to, using chained certificates), and (ii) the CA's corresponding root certificate, or a root certificate that the certificate chains back to, is stored in the browser's root store. Unfortunately, not all CAs are equally trustworthy, CAs sometimes make mistakes (e.g., due to human errors, intentional fraud, etc. [16]), and there is no current PKI mechanism informing domain owners of issued certificates.

The high reliance on CAs combined with some high-profile (but hard-to-detect) incidents have prompted various efforts to address the shortcomings of

This is the authors' version of the paper that will appear in *Proc. Passive and Active Measurement Conference (PAM)*, Berlin, Germany, Mar. 2018. The final publication is available at Springer via http://dx.doi.org/[doi].

the TLS/SSL landscape [6,13,14,17,18,23]. One of the most successfully deployed such systems is Certification Transparency (CT) [12, 18, 19]. To address some of the flaws with the current TLS landscape, CT requires that certificates are published in public append-only logs and that servers provide clients (browsers) proof, in the form of Signed Certificate Timestamps (SCTs), that the certificates have been logged in credible CT logs.

CT is already used by Google's Chrome browser for validation of certificates and Mozilla is drafting their own CT policies for Firefox. There also exist many public well-maintained logs that have proven valuable in identifying rogue certificates. In prior work, we analyzed the content of these CT logs [12] and VanderSloot et al. [22] have analyzed their certificate coverage. Here, we instead study the server-side adoption. In particular, we characterize the SCT usage among the one million most popular domains according to `alexa.com`, which due to high popularity skew are responsible for most of the web traffic [11].

SCTs can be delivered to a client in three different ways [18]: (i) using the X.509v3 extension, (ii) using the TLS extension, and (iii) using Online Certificate Status Protocol (OSCP) stapling. Each of these methods comes with their own advantages and disadvantages. In this paper, we first highlight some of these tradeoffs (Section 2) and then analyze the delivery choices made by different domains, including what domains select which delivery method and whether there are differences in the certificates associated with the different methods, the logs used to store the corresponding certificates, and other factors. We also use our measurements to look closer at overheads and the potential performance impact that SCT delivery may have when using the different methods.

For this work, we developed a measurement tool (Section 3) that extracts rich meta information about the handshake process, the SCTs, the SCT delivery, and the associated certificates. Using the tool, we collect and analyze two snapshots of the server-side SCT adoption as seen on May 30, 2017 and Oct. 6, 2017. These datasets allow us to capture the current status and comment on the impact that potential trends may have on the results. For our analysis, we first characterize the SCT usage (Section 4) as seen across popularities and how the number of SCTs and the log selection differ between domains that use different SCT methods. We then present a certificate-based analysis (Section 5) that looks closer at biases between the SCT delivery methods used and the type of certificates, signatures, and public keys, for example, providing us with some initial insight into the characteristics of the domains that use each delivery method. Finally, we present a performance and overhead analysis (Section 6) in which we analyze the handshake times, time until the clients obtain the SCTs, and quantify the potential delay and byte overheads associated with delivering the SCTs.

Our observations has implications on organizations running web services and our basic quantifications highlight SCTs minimal overheads. While most domains using SCTs opt to use the simplest delivery method (X.509v3 extension), which does not require any server-side changes, the fastest delivery method (TLS extension), which delivers the SCTs earlier within the handshake and only to clients requesting SCTs, is most frequently used by organizations (e.g., Google)

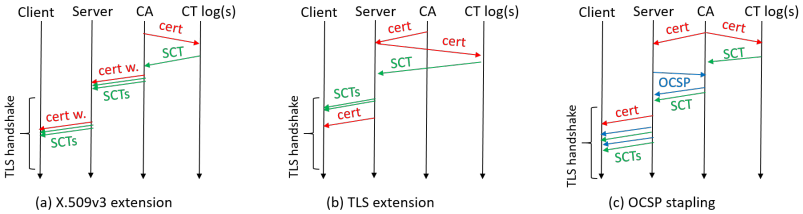

Fig. 1. High-level overview of the three SCT delivery methods.

that we (based on our measurement observations) conjecture are more performance oriented. It is also very encouraging that certificates that are accompanied with SCTs are much less likely to use weak signatures or public keys. Overall, the CT adoption, and use of the TLS extension in particular, is highest among the top domains, hopefully pushing others to follow.

2 Background

Browsers are increasingly requiring certificates to be included in CT logs. For example, since 2015, Google's Chrome browser has required that all Extended Validation (EV) certificates are accompanied by multiple SCTs before displaying visual cues to the user that normally come with these certificates. Today, they also require all certificates created by some (less trusted) CAs that have been caught misbehaving (e.g., Symantec) to be logged, and during the 39th CA/Browser Forum (Nov. 2016), the Chrome team announced plans that all certificates issued in Oct. 2017 or later will be expected to comply with Chrome's CT policy. Recently, Mozilla has also announced that CT is coming to Firefox.[1]

There are three methods for a server/domain to obtain and deliver SCTs to the clients. These methods use (i) the X.509v3 extension, (ii) the TLS extension, and (iii) OCSP stapling. The methods differ *both* by how the server obtains the SCTs and how the SCTs are delivered to the client. Figure 1 summarizes the main differences. From a service provider's perspective, the X.509v3 extension is by far the simplest method and does not require any server changes. Instead, the CA submits the certificate to the logs, obtains the corresponding SCTs, and bundles them together with the certificate (as part of the X.509v3 extension), allowing the server to deliver the STCs as a bundle together with the certificate (during a regular handshake). In contrast, with the TLS extension, the server submits the certificate (obtained from the CA) directly to the desired logs, obtains the corresponding SCTs, and then delivers the STCs to the client using the TLS extension option. While this method requires some changes to the server, we note that the TLS extension option comes earlier in the handshake and therefore typically allows faster delivery of SCTs. (This observation is analyzed further in Section 6.) Also OCSP stapling requires additional modifications on the server side; in this case to obtain an OCSP stapled SCT bundle that the CA creates after obtaining the SCTs. Compared with the other two methods, OCSP stapling results in later SCT delivery, since it takes place at the end of the handshake.

[1] https://www.thesslstore.com/blog/firefox-certificate-transparency/

3 Methodology

Using a collection of Java APIs available via Bouncy Castle[2], we implemented a special purpose program that we use as a tool for data collection and management of measurement campaigns.[3] Given a list of domains (in our case the top one million websites according to `alexa.com`), our program tries to establish a TLS/SSL connection with servers representing each domain. Using the SSLSocket in the Bouncy Castle library, during the TLS handshake, the program extracts and records detailed statistics about byte overheads, the SCT bundles, the certificates, the algorithms used during the handshake, timing information (e.g., time of handshakes, and time to obtain SCT bundles), and general information regarding the handshake process (e.g., why some connections fail).

The program implements all three SCT delivery methods and records information up-to the time of the first HTTP request, when connections are fully established and all potential SCTs have been obtained. After download of SCTs, the program decodes the SCTs and collects information about the logs used and the SCTs themselves. Public lists of known object identifiers (OIDs) and issuer information are used to determine the validation method of certificates.

To allow efficient processing even when a significant number of domains timeout and reduce time-of-day effects, the program runs 600 parallel client threads. At each point in time a thread is responsible for collecting statistics for one domain. To avoid startup and end effects (e.g., unfair CPU availability for the first opened threads), at the start and end of an experimental run, a set of additional HTTPS "dummy" websites are processed but not included in the results. We have run experiments with other number of parallel client threads, but have found that 600 provides a nice tradeoff between the speed of the measurement campaign and representative (and stable) performance values when a client visits these domains. A measurement campaign takes on average four hours.

Limitations: Perhaps the largest limitations of our setup is that we only run experiments from a single machine, and that we needed to run parallel threads to obtain timely results. Naturally, the network connectivity and location of the measurement device impacts the absolute handshake and SCT delivery times reported. However, we believe that the relative timing values still provide nice insights into differences observed between SCT delivery methods and that byte overheads will likely be much less impacted by location. We therefore focus on relative differences observed between the SCT delivery methods, not absolute delivery times. Focusing on these aspects also minimize the impact that the use of parallel threads may have on conclusions and insights.

4 Dataset and SCT Usage

Due to the current changes in the CT landscape, we present results based on two datasets collected roughly four months apart: May 30, 2017, and Oct. 6,

[2] Bouncy Castle, `https://www.bouncycastle.org`

[3] Code+datasets available: `http://www.ida.liu.se/~nikca89/papers/pam18.html`.

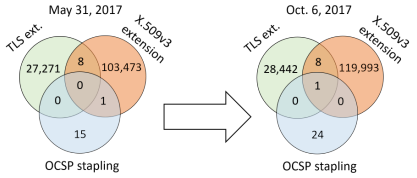

Fig. 2. Overview of dataset.

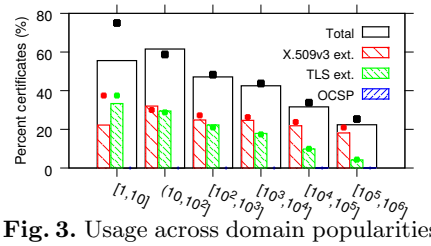

Fig. 3. Usage across domain popularities.

2017. Throughout this paper we use $x_1 \rightarrow x_2$ to indicate the values x_1 and x_2, for the same metric x, observed on these dates, respectively. The relative change provides an estimate of current change in the metric x.

Overview: For these two datasets, out of the top one million domains (according to `alexa.com`), 8.70→8.68% did not respond, 10.88→8.72% did not provide a certificate (and was deemed not to use HTTPS), and 23.52→26.20% resulted in the tool flagging a TLS error (typically indicating that the certificate is not valid). This left us with 557,485→563,866 sites that delivered valid X.509 certificates. While this suggests a small relative increase in the number of domains that uses HTTPS over this period, we were encouraged to see that the subset of domains that deliver SCTs with their certificates have increased more; from 130,768 (23.46%) to 148,468 (26.33%).

Figure 2 provides a breakdown of the delivery methods used by the servers to deliver these SCT bundles and how the use of the methods have changed. We note that the majority (103,482→120,002) of the SCT bundles are delivered using the X.509v3 extension. This is perhaps not surprising since this method does not require any changes to the servers. However, we also observe many SCT bundles (27,279→28,451) that are delivered using the TLS extension and a few (16→25) that are delivered using OCSP stapling. Again, both these later methods require server-side modifications. This may also explain why the X.509v3 extension is responsible for most of the increase in SCT usage.

Popularity-based usage breakdown: Figure 3 shows the SCT usage for domains with different popularity rank. Here, and throughout all other bar graphs in the paper (except Figure 5), we use bars to show the May 2017 values and large dots (with same colors) to indicate the corresponding Oct. 2017 values. The SCT usage is highest among the most popular domains (e.g., above 60% in among the top-100 domains and 49% among the top-1000 domains across both datasets). The top domains are also relatively equally likely to use the TLS extension and the X.509v3 extension for the delivery of the SCTs, whereas the (simpler) X.509v3 extension by far is the most popular choice among the less popular domains (e.g., X.509v3 is used to deliver 69.0→70.5% of the SCT bundles in the range $(10^4, 10^5]$ and 81.1→82.7% of the bundles in the range $(10^5, 10^6]$). The reason that OCSP stapling is not visible in the figure is that all 16→25 cases are for less popular domains, in the $(10^4, 10^6]$ range.

Bundle sizes: Certificates are expected to be accompanied by multiple SCTs. For example, with Chrome's EV policy a certificate should be logged in at least one Google operated log and one other (typically CA operated) log.

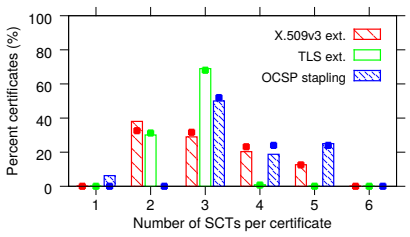

Fig. 4. Number of SCTs per certificate for each type of SCT distribution mechanism.

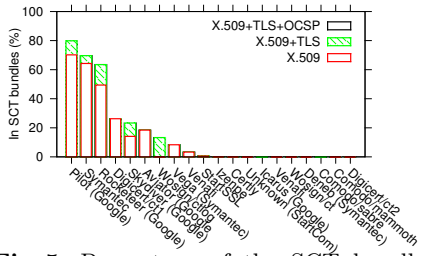

Fig. 5. Percentage of the SCT bundles that the observed CT logs covers.

The mean and median number of SCTs per bundle are relatively similar across the methods and we have not seen any major changes in the numbers. For example, in May the averages were 3.08 (X.509v3), 2.71 (TLS), and 3.56 (OCSP), respectively, and in Oct. the averages were 3.16, 2.70, and 3.72. Similarly, the median has remained equal to 3 for all three classes. However, there are substantial distribution differences between the methods. This is illustrated in Figure 4. With the TLS extension, almost all bundles include two (30.1→31.2%) or three (69.0→68.0%) SCTs. In contrast, the size of the bundles delivered using the X.509v3 extension are much more diverse. Although, the most common cases again is that two (38.0→32.6%) or three (29.0→31.7%) SCTs are included, with the X.509v3 extension, there is also a substantial number of bundles with four (20.3→23.2%) and five (12.7→12.6%) SCTs per bundle. Also with OCSP stapling we see relatively more SCTs per bundle. For example, 44→48% of these bundles have four or five SCTs. The smaller and more homogeneous bundle sizes observed with the TLS extensions are likely due to these sites being more performance conscious. We discuss this further in Section 6, when looking at the overheads and delivery times of the SCT bundles.

Logs used: Figure 5 shows the percentage of times each log is observed in an SCT bundle observed in the Oct. 2017 dataset. In general, the log usage is heavily skewed towards a small subset of logs, dominated by Google logs and logs operated by three major CAs (Digicert, Symantec, and Wosign). The main differences between the two datasets (May 2017 omitted this time) are that the Oct. 2017 dataset contains four extra logs (16 vs. 20) and that Aviator (operated by Google) has seen a drop in rank (from 4 to 6) and number (percent) of SCT bundles; from 39,889 (30.5%) to 27,336 (18.4%). This drop is explained by Aviator being frozen on Nov 29, 2016.[4]

Referring to the Chrome policy, we only found 21 SCT bundles in the Oct. dataset that did not have at least one SCT from a Google operated log. All these contained a single SCT; 4 were logged in Deneb (by Symantec) and 17 came from an "unknown" log for which we could not find a public log with the listed logID.[5] However, since all certificates with SCTs from this log (the same set of 17 single-log bundles) were issued by StartCom, we conjecture that it is operated by StartCom. Another interesting observation is that the main Digicert

[4] Chrome bug report: `https://crbug.com/389514`.

[5] `https://www.certificate-transparency.org/known-logs`

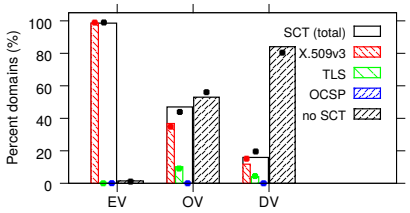

Fig. 6. Delivery method used by domains to deliver certificates of each type.

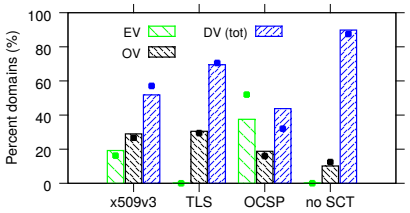

Fig. 7. Certificate types used by the domains using each delivery methods.

log (rank 4) and Aviator (old Google log with rank 6) almost entirely contains certificates for which the SCTs are delivered with the X.509v3 extension (12 of 38,964 and 6 of 27,336 non-X.509v3 extension SCTs, respectively) and that the Wosign log (rank 7) contains almost only certificates for which the SCTs are delivered using the TLS extension (only 198 of the 19,691 domains logging their certificates in this log do not use the TLS extension). This suggests that some CAs may have strong biases in how they help their customers deliver SCTs.

5 Certificate-based Analysis

Certificate type: We have found very large differences in how different types of certificates are delivered. This is highlighted in Figure 6, which breaks down the SCT delivery methods used for each type of certificate. We note that the SCT usage is by far the highest among domains that use EV certificates and the lowest among domains that use DV certificates. For example, 98.6→99.0% of the domains that use EV certificates use SCTs, but only 15.3→15.1% of the DV domains uses SCTs. The large SCT adoption for domains using EV certificates is expected since SCT compliance has long been required for Chrome (and soon other browsers). Perhaps more surprising is that we still observed 289→203 domains in the top-million websites that did not yet appear to deliver SCTs with their EV certificates. Despite being a decreasing fraction (1.4→1.0%), this is still a non-negligible number of domains. For OV certificates the absolute number of SCT domains is larger and increasing, although unfortunately the ratio of OV domains that use SCTs is decreasing (47.0→44.0%).

The X.509v3 extension is by far the dominating (98.5→98.9%) delivery method for EV domains. This may be an effect of many domains having had to scramble to deliver SCTs for their EV certificates and therefore opted to use the simplest-to-deploy method, not requiring any changes to their servers. However, it may also be due to rumors that Chrome would stop supporting the TLS extension and OCSP stapling (rumors that Google have strongly dismissed![6]), domains using the least resistance path (not requiring any server changes), and biases in the methods promoted by some CAs (e.g., Figure 5). For domains using OV certificates, 36.8→34.8% of the domains use X.509v3 and 10.2→9.1% use the TLS extension. Figure 7 breaks down the same data on a per-delivery method

[6] CT FAQ: `https://www.certificate-transparency.org/faq`

Table 1. Top-five issuers for domains using each SCT delivery method (Oct. 2017) and the number of domains using their certificates with that delivery method (in brackets).

Rank	X.509v3 ext.	TLS ext.	OCSP stapling	No SCT used
1	RapidSSL (25,130→34,006)	Comodo (18,392→19,525)	SwissSign (11→20)	Comodo (95,940→95,956)
2	GeoTrust (16,434→17,464)	Google (7,888→7,858)	DigiCert (5→5)	Let's Encrypt (52,891→65,635)
3	Thawte (12,349→13,545)	Go Daddy (358→366)	-	Go Daddy (33,000→32,474)
4	Symantec (12,649→13,260)	DigiCert (158→219)	-	cPanel (31,629→32,118)
5	AlphaSSL (8,676→10,880)	CloudFlare (121→122)	-	DigiCert (21,053→21,378)

basis. Again, differences between the methods are visible. For example, the domains using the X.509v3 extension typically deliver a much larger fraction of EV certificates than those that use the TLS extension.

Top issuers: Table 1 summarizes the top issuers in each category. Except Let's Encrypt, which targets the low-budget market, most top CAs appear to have increased their SCT usage. RapidSSL has seen the largest increase in SCTs delivered with the X.509v3 extension (25,130→34,006), simultaneously as dropping of the top-five list for non-SCT certificates (25,087→8,766). The other main exception is DigiCert, who now delivers less certificates with the X.509v3 extension (10,576→9,403) and slightly more non-SCT certificates (21,053→21,378).

For certificates delivered with the TLS extension, we found that 7,888→7,858 of the 8,314→8,374 OV domains used certificates from Google (typically clear Google owned domains such as google.com, google.se, or some-name.blogspot.com, for example) and Comodo was responsible for 18,335→19,458 of the 18,960→20,074 DV domains (and 57→67 OV domains). In the complete dataset, we only observed 149→193 other domains using Google issued certificates. Also these where OV certificates, but no corresponding SCTs where delivered during the handshake. These domains typically were associated with companies that have Google as parent company (e.g., `nest.com`). Clearly, Google has decided to use the TLS extension to deliver SCTs for their domains. One reason for this is perhaps that SCTs delivered using the TLS extension are delivered earlier in the handshake than when using the X.509v3 extension; therefore, allowing more time to process the SCTs. In Section 6 we look closer at this and other performance aspect.

In contrast to Google, Comodo also had issued many certificates for domains that used the X.509v3 extension (5,374→5,355) and domains that did not use SCTs (102,092→96,629), including 21→20 EV certificates without SCTs. Overall, Google and Comodo appears to be early adopters of the TLS extension. For domains using OCSP stapling, eleven used SwissSign and five used DigiCert. The set of top issuers using the X.509v3 extension was much more diverse.

Signatures: Figure 8 shows the fraction of domains that use different signature algorithms together with each type of SCT delivery method. We note that 99.9→99.8% of the certificates associated with X509.v3 SCTs are signed with RSA. This is very similar to what we observe for the certificates delivered using OCSP stapling and those that we did not associate with any SCTs. In sharp contrast, 65.0% of the certificates associated with SCTs delivered using the TLS extension are signed with ECDSA (all using SHA256).

We have also found that domains using SCTs are less likely to use weak signature algorithms than the non-SCT domains. For example, among the SCT domains, we only found 318 (0.24%)→ 1,017 (0.68%) domains that used SHA1 (with RSA). The corresponding numbers for non-SCT domains are 49,607 (11.6%)

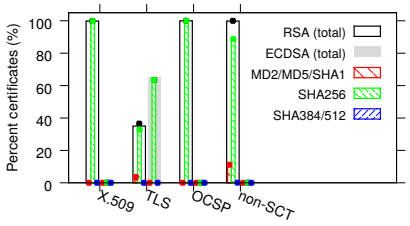

Fig. 8. Signatures used for certificates.

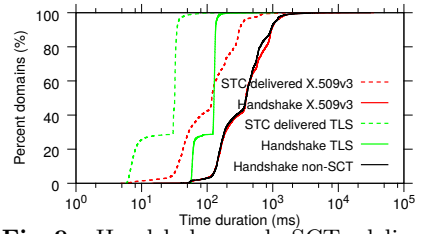

Fig. 9. Handshake and SCT delivery times of domains using different methods.

→ 44,398 (10.7%). There were even 2,048 (0.48%) → 1,813 (0.44%) non-SCT domains that used MD5. The significant use of SHA1 and MD5 are concerning since they long have been known to be susceptible to attacks. While the SCT domains clearly use weak signatures algorithms much more seldom, we were surprised by the relative raise in use SHA1 among these domains. A closer look revealed that except one GeoTrust certificate, all the other 1,016 SHA1 certificates were DV certificates issued by Comodo (DV legacy server).

Public keys: Also when looking at the public keys included in the certificates, the certificates with SCTs delivered using the TLS extensions sticks out. In particular, among these 27,279→28,451 certificates, a total of 17,724 (65.0%) → 18,071 (63.5%) are using Elliptic Curve (EC) keys. In contrast, among the 103,482→120.002 domains associated with SCT bundles delivered with the X.509v3 extension only 164 (0.16%) → 230 (0.19%) and none of the 16→25 OCSP stapled certificates use EC. For SCT related certificates, all remaining public keys use RSA. When interpreting these results, it should be noted that RSA (99.5→99.6%) also is dominating among the public keys seen among non-SCT domains. Again, a significant reason for the above differences are due to Comodo, who is responsible for 17,940 of the 18,071 domains using EC with the TLS extension. The other EC users in the TLS category (although using EC less frequently) are CloudFare (122), Let's Encrypt (5), DigiCert (2), and AlphaSSL (2). While omitted, we have also found that public keys not associated with SCTs are more likely to use shorter RSA key lengths.

6 Performance and Overhead Analysis

Handshake and SCT delivery times: We have not observed any significant differences in the handshake times when using our SCT enabled client with the SCT capability turned on or off, regardless if it communicates with domains that use SCTs or not. Instead, the handshake time distributions for these client variations are almost identical, regardless of the subset of domains considered. In the following, we therefore only show results for a client using all three methods.

When comparing the delivery methods, on the other hand, there are significant differences in the handshake times, and (perhaps most importantly) in the times until the SCTs are delivered to the clients. Figure 9 highlights these differences. Here, we have plotted the total handshake times (solid lines) and the SCT delivery times (dotted lines) for the different delivery methods.

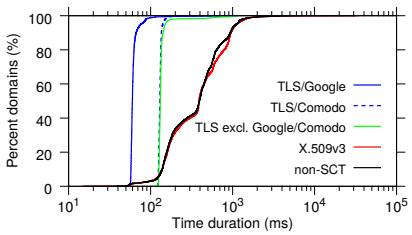

Fig. 10. Handshake times when breaking down domains using the TLS extension.

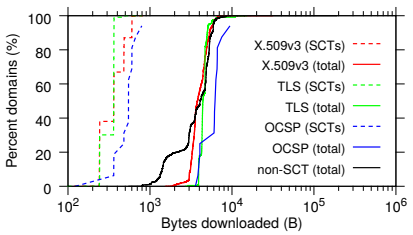

Fig. 11. Bytes delivered as part of the SCT bundles and total bytes received.

First, note that the handshake time distribution for domains using the X.509v3 extension is almost identical to that of non-SCT domains. In contrast, the handshake times with domains using the TLS extension are much shorter. This suggests that the domains using the TLS extension may leverage service replication (e.g., using third-party CDNs or their own distributed data centers) to a larger extent. This observation also matches our prior observation (e.g., Figure 3) that these domains are more likely to be popular domains that perhaps are more likely to be both performance aware and early adopters. As interesting and supportive evidence for the conjecture that these domains are more performance aware, we note that the bump with low-delay handshakes is almost entirely due to Google domains. This is highlighted in Figure 10. Here, we also separate Comodo and the "other" domains using the TLS extension; both of which have roughly the same handshake time distribution. Similar short-tailed, low-delay distributions as we observe for Google here, have also been observed when analyzing the RTTs (from many different locations) to Google infrastructure in the past [3]. It is also interesting that the other domains using the TLS extensions provide lower handshake times than both the non-SCT domains and X.509v3 domains and that those domains are responsible for the majority of the distribution. In addition, performance aware websites may select the TLS method since this method allows the SCTs to be delivered only to clients requesting SCTs.

Figure 9 highlights that the SCTs often are delivered much sooner (within the handshake) when using TLS than when using the X.509v3 extension. The reason for this was highlighted in Section 2 and is due to the TLS extension happening earlier in the handshake. Clearly, this would give a client (browser) more time to decode the SCTs and process the information associated with them. As a reference point, the simple/naive decoder that we used was able to decode 99% of the individual SCT bundles within 0.161 ms (during the data collection). Since this delay is small compared to the handshake itself, we can approximate also the distribution of the time until the clients have the decoded SCTs with the dotted lines, again highlighting that the TLS extension would be preferred from a performance standpoint.

Finally, Figure 11 summarizes the byte overheads associated with the SCTs. Here, we plot both the size of the SCT bundles and the total bytes received during the handshake. Overall, the byte overheads of the SCTs are very small (x-axis on log scale) and there are only small differences between the delivery methods (due to differences in the number of SCTs per bundle; see Figure 4).

7 Related Work

Being relatively new, there is limited research characterizing the CT landscape. In parallel work to ours, Amann et al. [2] use both active and passive measurements to evaluate the adoption of a number of improvements and additions proposed to strengthen the X.509 PKI, including CT. Compared to that work, we use only active measurements, but place particular focus on the comparison of the relative differences in the server-side use and client-side performance of the three alternative SCT delivery methods. Gustafsson et al. [12] have characterized the usage of public CT logs and the certificates observed in these, but do not consider the use of different SCT delivery methods. VanderSloot have evaluated the certification coverage of the CT logs [22]. Others have proposed optimizations or enhancements to CT [7, 21].

There are also a lot of measurement-based research that have characterized the TLS/SSL landscape [1,4,10,15,16]. This includes many works that have tried to capture the trust landscape [16, 20], identified weaknesses in the TLS/SSL connection establishment [5,9], or identified SSL error codes and their reasons [1]. These works typically excluding CT from the analysis, although a few have commented that CT may significantly change the landscape. We should also note that there have been various other attempts to address the limitations in the current TLS/SSL landscape [6,13,14,17,18,23], but thus far most other have seen limited adoption [2].

8 Conclusions

Our analysis of two snapshots (May and Oct. 2017) of the SCT usage among the one million most popular web domains provides insights into the current status of the SCT adoption and highlights key tradeoffs between the three different SCT delivery methods and the choices made by different domains. Whereas the majority of domains have opted for the simplest solution (using the X.509v3 extension) that does not require any server side changes, it is interesting to see that the method that provides the fastest delivery of SCTs (the TLS extension) is used by organizations (e.g., Google) that appear to provide much faster connection establishment, handshake times, and smaller SCT bundle sizes. We have also seen that SCT delivery has low overhead and that SCT usage is highest among the very top domains, hopefully pushing others to follow. By comparing the two snapshots we also observe some positive and encouraging trends in the adoption, including an overall increase in use of SCTs, how the use of SCTs goes hand-in-hand with a reduced use of weak signatures and public keys, and that big players such as Google is pushing the adoption. On the slightly negative side, it appears that some CAs may have a bias towards the (simpler) X.509v3 extension, although (performance-wise) many of their customers may benefit from implementing the TLS extension method (e.g., as used by Google).

Acknowledgements: The authors are thankful to our shepherd Niky Riga and the anonymous reviewers for their feedback. This work was funded in part by the Swedish Research Council (VR).

References

1. D. Akhawe, B. Amann, M. Vallentin, and R. Sommer. Here's my cert, so trust me, maybe?: understanding TLS errors on the web. In *Proc. WWW*, 2013.
2. J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz. Mission accomplished? HTTPS security after diginotar. In *Proc. IMC*, 2017.
3. M. Arlitt, N. Carlsson, C. Williamson, and J. Rolia. Passive crowd-based monitoring of world wide Web infrastructure and its performance. In *Proc. ICC*, 2012.
4. H. Asghari, M. J. G. van Eeten, A. M. Arnbak, and N. A. N. M. van Eijk. Security economics in the HTTPS value chain. In *Proc. WEIS*, 2013.
5. B. Beurdouche et al. A messy state of the union: Taming the composite state machines of TLS. In *Proc. IEEE S&P*, 2015.
6. D. Basin, C. Cremers, T. H.-J. Kim, A. Perrig, R. Sasse, and P. Szalachowski. Arpki: Attack resilient public-key infrastructure. In *Proc. ACM CCS*, 2014.
7. L. Chuat, P. Szalachowski, A. Perrig, B. Laurie, and E. Messeri. Efficient gossip protocols for verifying the consistency of certificate logs. In *Proc. IEEE CNS*, 2015.
8. J. Clark and P. C. van Oorschot. SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements. In *Proc. IEEE S&P*, 2013.
9. D. Adrian et al. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In *Proc. ACM CCS*, 2015.
10. T. Fadai, S. Schrittwieser, P. Kieseberg, and M. Mulazzani. Trust me, I'm a root CA! analyzing SSL root CAs in modern browsers and operating systems. In *Proc. ARES*, 2015.
11. P. Gill, M. Arlitt, N. Carlsson, A. Mahanti, and C. Williamson. Characterizing organizational use of Web-based services: Methodology, challenges, observations, and insights. *ACM Trans. on the Web*, Oct. 2011.
12. J. Gustafsson, G. Overier, M. Arlitt, and N. Carlsson. A first look at the ct landscape: Certificate transparency logs in practice. In *Proc. PAM*, 2017.
13. P. Hallam-Baker and R. Stradling. *RFC6844: DNS Certification Authority Authorization (CAA) Resource Record*. IETF, 2013.
14. P. Hoffman and J. Schlyter. *RFC6698: The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA*. IETF, 2012.
15. R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSL landscape: a thorough analysis of the X.509 PKI using active and passive measurements. In *Proc. IMC*, 2011.
16. L. Huang, A. Rice, E. Ellingsen, and C. Jackson. Analyzing forged SSL certificates in the wild. In *Proc. IEEE S&P*, 2014.
17. T. H.-J. Kim et al. Accountable key infrastructure (AKI): A proposal for a public-key validation infrastructure. In *Proc. WWW*, 2013.
18. B. Laurie, A. Langley, and E. Käsper. *RFC6962: Certificate Transparency*. IETF, 2013.
19. B. Laurie, A. Langley, E. Käsper, E. Messeri, , and R. Stradling. *RFC6962-bis: Certificate Transparency draft-ietf-trans-rfc6962-bis-10*. IETF, 2015.
20. G. Ouvrier, M. Laterman, M. Arlitt, and N. Carlsson. Characterizing the HTTPS trust landscape: A passive view from the edge. *IEEE Com. Mag.*, July 2017.
21. M. D. Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *Proc. NDSS*, 2014.
22. B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. Halderman. Towards a complete view of the certificate ecosystem. In *Proc.IMC*, 2016.
23. D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving SSH-style host authentication with multi-path probing. In *Proc. USENIX ATC*, 2008.