State of the Certificate Transparency Ecosystem

Nikita Blagov, Max Helm*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: n.blagov@tum.de, helm@net.in.tum.de

***Abstract*—Certificate Transparency (CT) can improve end user's security advancing the already established HTTPS; it is an emerging, though not yet fully developed technology. Its imperfections bring about new potential threats such as Partitioning Attacks, leaking of domain names, or overloaded logs. In this paper, we study how CT has been deployed so far and which browsers already support it. We take a closer look at logs and their requirements, as well as available HTTP headers assisting CT. We further study the aforementioned threats and demonstrate currently available countermeasures.**

***Index Terms*—certificate transparency, log, browser, deployment, expect-ct, threats, equivocation, gossiping, leaking domain names, wildcard certificates, label redaction**

1. Introduction

HTTPS ensures confidentiality on networks by encrypting communication. In a TCP connection, a client's browser performs a handshake with a webserver during which an SSL certificate is submitted, proving authenticity of the domain (represented as 'B' in Figure 1). Website owners request this certificate from a Certificate Authority (CA) ('A' in Figure 1). [1]

In 2011, however, hackers compromised the Dutch CA DigiNotar and issued a fraudulent certificate targeting Google users in Iran [2]. Subsequently, "the certificate was revoked and the offending CA was removed from client trust stores" [1], causing many Dutch websites using a certificate from DigniNotar becoming inaccessible. On numerous occasions, similar incidents were not detected for several weeks causing a great deal of damage [3].

To better cope with such incidents, Certificate Transparency (CT) has been developed [1]. It aims to provide "an open auditing and monitoring system that lets any domain owner or CA determine whether certificates have been mistakenly or maliciously issued" [3], ultimately protecting end users.

This paper provides an overview of background information, CT's current deployment, contemporary threats and their countermeasures. The rest of this paper is structured as follows. Chapter 2 presents important background information and introduces the key concepts. Chapter 3 focuses on CT's current deployment. How successfully has CT been developed so far? Is it already supported by every browser? How does this support look like? Which requirements must logs meet to become trusted? Can HTTP headers assist if CT is not supported by default? Chapter 4 focuses on contemporary threats and unintended uses of CT. What are Partitioning Attacks and is there any available countermeasure today? Why is the leaking of domain names treacherous and can it be prevented? What if logs become overloaded? Has it ever happened at all and how could it be averted? Related work can be found in Chapter 5.

2. Background

CT extends the existing TLS system by three new actors: logs, monitors, and auditors. Figure 1 illustrates the existing components in blue and supplementary ones in green.

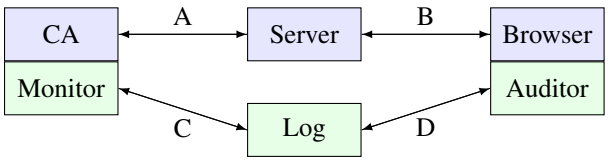

Figure 1: Components of CT [4]

Logs are independently operated, publicly auditable records of certificates. They are append-only: once a certificate has been added to a log, it cannot be deleted, modified, or retroactively inserted.

Logs are formed by a binary Merkle Hash Tree (with leaves and nodes). Figure 2 exemplifies such a tree. Each leave represents a value of the corresponding appended hashed certificate. Leaves are paired with other leaves forming nodes, which can be paired too forming further nodes. Nodes also represent hash-values. The root hash, comprising all nodes and leaves, is called the Merkle Tree Hash (MTH). It assures logs are append-only since any change of log's entries leads to a different MTH. Logs regularly sign it together with other information such as the Merkle Hash Tree's size and version. A signed MTH is called Signed Tree Head (STH). [4]–[7]

"A certificate is considered included in a log when it is covered by an STH" [5]. "When a log receives a certificate, it replies with a Signed Certificate Timestamp (SCT)" [1].

An SCT is a promise to add the certificate within a time known as Maximum Merge Delay (MDD). There are three ways to deliver an SCT either embedded as an X.509v3 Extension, as an TLS Extension, or via OCSP Stapling. The first method is most widely used and oftentimes the only supported. The CA logs the certificate

doi: 10.2313/NET-2020-11-1_09

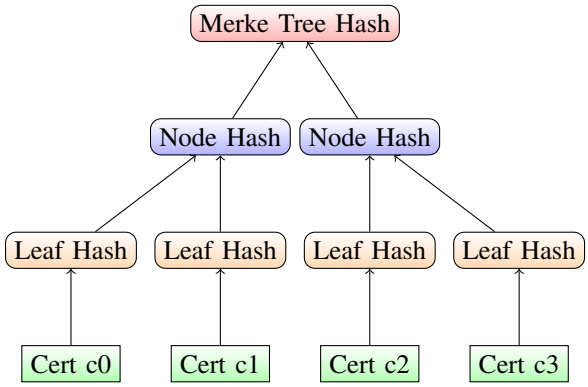

Figure 2: Merkle Hash Tree: Log's Structure [6], [8]

and embeds the SCT, thus it does not require any server modification. [4]

In TLS Extension, domain's server itself submits the certificate to the log and obtains the SCT itself [4]. It is considered the fastest method but requires modifying the server, thus less supported [9]. OCSP Stapling is more complex, very little supported and disabled in Chrome [5].

Both certificates and logs can lose their trust. Misissued or bogus certificates are revoked. Misbehaving logs can be fully distrusted, disqualified, or frozen. A frozen log remains in read-only mode but cannot add any new certificates or sign any new STHs, though existing STCs remain being accepted. Logs face similar consequences if they are disqualified, except that non-embedded SCTs must be replaced. In case a log is fully distrusted, none of its issued SCTs will be accepted anymore. [1]

Monitors detect suspicious certificates in logs and verify the visibility of all certificates ('C' in Figure 1). They are typically run by CAs and since they maintain a log's complete copy, they can "act as a backup read-only log". [4]

Auditors are typically included in client's browsers and verify SCT's validity [6]. They also assure that a "certificate represented by a given SCT has in fact been logged" by requesting an inclusion proof [1]. In addition, they can request a consistency proof to verify that log's entries have neither been modified nor retroactively deleted and logs only present consistent views ('D' in Figure 1) [1].

3. Deployment

CT was first standardized in June 2013 with RFC 6962 [7]. Today, CT is already supported in more than 60% of HTTPS traffic [1]; still not every browser is capable of effectively handling it. Overall, CT is an emerging, yet not fully developed technology.

In the following, we show which of the major four browsers provide support. We take a more detailed look at Chrome for two reasons. First, Google has been the pioneer in CT deployment and its enforcement. Secondly, Chrome was by far the most widely used browser worldwide with over 65% in May 2020. Safari browser was on the second place with 18%, followed by Firefox and Edge both with around 3% [10]. We examine Chrome's and Safari's policies for CT and trusted logs and finally introduce two HTTP headers: Require-CT and Expect-CT.

3.1. Browser Inclusion

Fearing to break too many websites, browsers enforced CT gradually. At first, Google required all Extended Validation (EV) certificates issued following January 2015 to be CT-logged. In September 2017, Chrome also began to provide support for the Expect-CT HTTP header. In July 2018, they enforced CT for all certificates issued since May to comply with Chromium's CT policy. If during a TLS handshake the certificate is missing, expired or is not logged, the connection fails and users are shown a security warning. [1], [11]

Each certificate must be accompanied by at least two SCTs from diverse qualified logs, one from a Google log and one from a non-Google Log. This forces a potential attacker to compromise multiple different logs simultaneously [12]. Currently qualified logs are listed in [13].

If at least one of the SCTs is delivered using TLS extension or OCSP Stapling, the aforementioned criteria is enough; otherwise, the number of required logs depends on certificate's validity period, as listed in Table 1. Should it be shorter than 15 months, 2 distinct (qualified) logs are sufficient. One more additional log is required in case the certificate's lifetime is between 15 and 27 months. The policy argues against exceeding 27 months; however, it does not prohibit it. Supposing the validity period is longer than 27 months but maximum 39, 4 different logs are needed; all longer valid certificates must be logged in 5 diverse logs. [12]

TABLE 1: Number of Embedded SCTs in Chrome [12]

Lifetime of Certificate	Number of SCTs from distinct logs
< 15 months	2
>= 15, <= 27 months	3
> 27, <= 39 months	4
> 39 months	5

Apple quickly followed with its Safari enforcing all certificates issued subsequent to October 15th, 2018 to be CT-logged. Its CT policy comprises the same criteria as Chrome's [14]. Microsoft announced their aspiration for CT checking in Edge [15], though there is no current support yet [16]. Neither is any available for Mozilla Firefox [17], their experimental implementation for telemetry caused performance issues [18]. The middle column of Table 2 displays the aforementioned four browsers and whether they enforce CT.

TABLE 2: Browsers' CT-Support [12], [14], [16], [17], [19], [20]

Browser	CT Enforcement	ExpectCT Support
Chrome	yes	yes
Safari	yes	no
Firefox	no	no
Edge	no	yes

3.2. Log Requirements

In the following, we explain which criteria need to be met for a log to become and remain trusted in Chrome and

doi: 10.2313/NET-2020-11-1_09

Safari. It is important to note that both Google and Apple can always withdraw their trust in a previously accepted log again for any reason. Overall, their log policies are almost identical. [21], [22]

3.2.1. Inclusion Request Requirements. When applying for log inclusion, its operators must provide a log's description, policies for accepting and rejecting certificates, as well as log's MDD. They must further list all accepted root certificates, log's public key and its URL for the public HTTP endpoint. They must finally provide contact information for people both operating and representing, Apple requires two contacts respectively. [21], [22]

An example (Nimbus 2018) for an inclusion request can be found in [23].

3.2.2. Monitoring: Ongoing Requirements. Both Google and Apple monitor the log after accepting its inclusion request to verify it conforms to their CT policies. They expect the following criteria to be met at any time. [21], [22]

Logs must comply with RFC 6962 and implement its CT correspondingly, e.g., being append-only and publicly auditable. They must always provide consistent views to different parties, i.e., avoiding equivocation (see Section 4.1). Their MDD must not exceed 24 hours. Both Google and Apple further require an uptime of 99%, though both measure it individually. [21], [22]

Apple instructs to "trust all root CA certificates included in Apple's trust store" and to "accept certificates that are issued by Apple's compliance root CA to monitor the log's compliance with these policies" [22]. Google does similarly and also issues own certificates for monitoring purposes that must be accepted when requested [21].

3.3. HTTP Headers

A site operator can utilize an HTTP header to necessitate all certificates for its domain being logged even if CT is not enforced by default in the browser. This has been particularly important for certificates issued prior to May 2018 as these certificates are not required to be CT-logged to be shown as valid in Chrome [19]. A common header is Expect-CT, an alternative is Require-CT. The latter one, however, is unofficial and relatively unpopular; while 7,300 domains were using the Expect-CT header, and only 8 were using Require CT as shown by Gasser et al. in [5]. Expect-CT may become outdated by June 2021 since certificates issued before May 2018 will all expire until then [19].

Expect-CT is formed by three directives: a compulsory max-age, as well as an optional enforce and report-uri [24]. "The following example specifies enforcement of Certificate Transparency for 24 hours and reports violations to foo.example" [19].

```
Expect-CT:
max-age=86400, enforce,
report-uri="https://foo.example/report"
```

Most commonly, however, only the reporting feature is used by setting max-age to zero and leaving out the enforce directive [5]. To the best of our knowledge, Expect-CT support is currently only implemented in Chrome and Edge, not, however, in Safari or Firefox, as displayed in the right column of Table 2 [19], [20].

4. Threats and Countermeasures

New technology often has weaknesses at the beginning that offer attackers new possibilities. Next, we introduce three threats arising with CT: equivocation, leaking of domain names, and overloaded logs. In addition, we demonstrate potential countermeasures.

4.1. Equivocation and Gossiping

Ideally, even if an attacker uses a bogus certificate, as long as the end user pays attention to warnings and uses a browser supporting CT, a conventional man-in-the-middle (MITM) attack is useless. The connection fails prior to causing any harm. Since the certificate is logged, CT monitors can quickly detect it and prompt its revocation. If the attacker is capable, however, to prevent monitors and auditors to detect the rogue certificate by hiding its entry in the log, they would impede any security measures. Monitors simply cannot ascertain the rogue certificate and the corresponding isolated (browser-) auditor is provided a fraudulent, though in itself consistent view. Such an attack relies on a second Merkle Hash tree version within a log with a different STH. [6]

It is called 'Split-View', 'Partitioning Attack', or 'Equivocation' [5], [6], [25].

To successfully counteract it, there is a technology called 'Gossiping' whereby auditors gossip about a log. They exchange STHs or SCTs to verify their view is consistent with others' view [26]. Unfortunately, it has "next to no deployment in the wild" yet [5]. Gossiping faces two major challenges. First, a defined and scalable mechanism must be standardized among all clients; secondly it entails a high risk of uniquely identifying (fingerprinting) clients by means of gossiped STHs or SCTs, and thus violating their privacy [27].

There are various experimental gossiping-mechanims; however, it is important to note that they "are drafts and are subject to rapid and substantial change" [6]. We demonstrate two of them, STH Pollination and Google's current experimental Minimal Gossip, an extension and refinement of the prior.

4.1.1. STH Pollination. In STH Pollination, both clients and auditors submit STHs to and receive them from the servers representing STH pools. Clients and auditors do not communicate directly. [26]

Since "an STH uniquely represents an entire tree version", it suffices for a consistency verification. "STHs are normally not considered privacy sensitive, as long as they are shared by a large set of clients". Therefore, there are two restrictive measures. First, only fresh, maximum 14 days old STHs are gossiped; older ones are discarded. Secondly, participating (gossiped) logs are prohibited to issue (sign) new STHs more frequently than once per hour; otherwise, they will be ignored. In effect, the two measures guarantee having at most 336 pollinated STHs for each log at any time. These are finally verified by auditors and monitors for consistency. [6]

doi: 10.2313/NET-2020-11-1_09

4.1.2. Minimal Gossip. Google's Minimal Gossip approach implements an altered version of STH Pollination. It focuses on minimal modification of already existing CT and introduces a new synthetic certificate chain conjoining a root and a leaf certificate. The root certificate identifies the gossiper (usually an auditor) and "indicates minimal gossip use". Pollinated (destination) logs add it to their acceptable roots, thus trusting the gossiper. Each STH received from a source log is embedded in a separate leaf which is then added to the destination log. The leaf certificate is valid for only 24 hours and marked as critical to avoid being mistakenly recognized as a valid conventional certificate. The verification can then be performed using goshawk. [27]

Goshawk "scans a destination log for gossiped STH values and checks consistency against the source logs" [28].

4.2. Domain Names Leaking and Label Redaction

Logs can easily be misused by attackers to discover new domain names that would otherwise remain secret. The new information aids targeting victims precisely, thus, exposing the whole CT ecosystem to risk. [29]

Likewise, logs complicate keeping new products confidential prior to their public release. Let us assume, the company Guenther GmbH ('guenther.eu') is developing a new product 'Moepi' and dedicates an individual domain, 'moepi.guenther.eu', to it. Competitors could discover and ascertain Moepi's development by simply scanning a log since logged domain labels, in this case 'moepi', are publicly readable. [30]

We look at three countermeasures: starting with the least applicable wildcard certificates, following with name-constrained intermediate CA certificate, and finishing with the most applicable mechanism called label redaction. The more applicable a mechanism is, the higher its complexity. [31]

4.2.1. Wildcard Certificates. In wildcard certificates, a wildcard "*" label effectively hides the private domain. For instance, the domain 'secret.example.com' can be replaced by '*.example.com'; 'moepi.guenther.eu' by '*.guenther.eu'. [31]

Wildcard certificates, however, face two limitations. First, they cannot be used when dealing with EV certificates. Secondly, they can only cover one level of subdomain, conforming to RFC 2818. Consequently, '*.example.com' could not replace 'top.secret.example.com' since it involves 2 levels of subdomains. [31], [32]

A special type are partial-wildcard certificates, e.g., '*p.example.com', which covers both 'top.example.com' and 'flop.example.com'. Partial-wildcard certificates, however, are disabled in major browsers and should, therefore, not be used. [32]

4.2.2. Name-Constrained Intermediate CA Certificate. Another way is logging a name-constrained intermediate CA certificate instead of the "end-entity certificate issued by that intermediate CA" [31].

Such an intermediate CA certificate comprises a name constraint extension [see RFC 5280] with two core components, permitted and excluded subtrees. A permitted subtree explicitly defines an allowed namespace, whereas an excluded subtree disallows certain namespaces. Assuming we want to allow 'example.com' and its subdomains (e.g., 'top.secret.example.com'), we would enter 'example.com' (for the domain itself) and '.example.com' (for all subdomains) under permitted subtrees. Likewise, to disallow 'bad.example.com' and its subdomains, we would add 'bad.example.com' and '.bad.example.com' under excluded subtrees. [31], [33], [34]

The following criteria must be met to log a name-constrained intermediate CA certificate. First, it must contain a non-critical extension indicating the acceptability of not logging certificates issued by the according intermediate CA. Secondly, there must be at least one DNS-name defined in permitted trees; and finally, excluded subtrees must disallow the full range of IPv4 and IPv6 addresses. [31]

This method is included by default in CT 2 [see RFC6962-bis], a currently developed improved version of CT [35], [36]. Its latest version is 34 [37].

4.2.3. Label Redaction. Label redaction can effectively hide non-wildcard domain-names by replacing them with redacted labels. In case of Guenther GmbH, 'moepi.guenther.eu' would be redacted to '(redacted).guenther.eu'. [30]

Each redacted label is determined, inter alia, using a hash function and a Base 32 Encoding function (see RFC 4648). Using hashing, legitimate domain owners can verify that each redacted label corresponds with the original label. A redactedSubjectAltName extension helps reconstructing the certificate by imparting which labels have been redacted. [31]

Nevertheless, label redaction is not absolute, as it may be forbidden by client's policies [31].

Furthermore, it requires domains owners to include new domains (new products) as subdomains of an existing domain. Thus, label redaction would not be sufficient provided Guenther GmbH wants to allocate an independent domain for Moepi, such as 'moepi.com'. It would require redacting the domain to '(redacted).com' which will be rejected by logs. Likewise, wildcard certificates ('*.com') also fail in this particular situation. [30]

4.3. Overloaded Logs and Log Splitting

Large CAs submit certificates to only a small number of CT logs, leaving them with a massive number of certificates [29]. This can lead a log to performance issues due to overload, which may result in disqualification or being frozen [29]. E.g., Google's log Aviator, which was frozen on November 30th, 2016 due to failing to add a certificate within the MDD [38].

Today, log operators often split their logs into several, one for each year, as can be seen in [39]. It reduces the number of certificates per log significantly. A potential disqualification affects only the logs expiring in the same year and not all together, thus, reducing the overall damage. The year represents the year of certificate's expiration, e.g., Nimbus 2018 promising to log only certificates expiring in 2018 and being frozen straight afterwards. [23]

Nevertheless, one may criticize these measures as not thoroughgoing enough. Despite log splitting, one Nimbus

doi: 10.2313/NET-2020-11-1_09

CT log encountered performance issues in 2018 during an update, risking being disqualified [40]. Nimbus and Google Argon are among the largest CT logs. The final (fixed) STH of 'Nimbus 2019' has had over 493 million entries, while the one of Google Argon 2019 has had over 857 million entries [41], [42]. One potential solution could be CAs distributing "their logging load more evenly among logs and log operators" [29].

5. Related Work

Scheitle et al. analyze the deployment of CT until the first half of 2018 and define new threats of certificates; they do not, however, study possible countermeasures [29]. Gasser et al. observe CT logs and search for violations of its requirements. They also examine HTTP headers and gossiping [5]. Stark et al. investigate challenges facing in deployment and its adoption on the web until 2019 [1].

6. Conclusion

In this paper, we provided an overview of CT's current deployment state; how Google and Apple have progressively implemented it so far, while others fall behind. We studied how CT can be unintendedly used by attackers or competitors to discover new domain names and how the use of wildcard or intermediate CA certificates, as well as label redaction can hinder them. We illustrated equivocation as a threat to the CT ecosystem and presented Google's experimental Minimal Gossip, an extension of STH Pollination, as a potential solution. Further research could be done examining gossiping's impact once it has been officially deployed and standardized.

References

[1] E. Stark, R. Sleevi, R. Muminovic, D. O'Brien, E. Messeri, A. P. Felt, B. McMillion, and P. Tabriz, "Does Certificate Transparency Break the Web? Measuring Adoption and Error Rate," in *2019 IEEE Symposium on Security and Privacy*. Los Alamitos, CA: IEEE Computer Society, 2019.

[2] H. Adkins, "Google Online Security Blog: An update on attempted man-in-the-middle attacks," https://security.googleblog.com/2011/08/update-on-attempted-man-in-middle.html, 2011, [Online; accessed: 7/06/2020].

[3] "What is Certificate Transparency? - Certificate Transparency," https://www.certificate-transparency.org/what-is-ct, [Online; accessed: 5/06/2020].

[4] "How Certificate Transparency Works - Certificate Transparency," https://www.certificate-transparency.org/how-ct-works, [Online; accessed: 5/06/2020].

[5] O. Gasser, B. Hof, M. Helm, M. Korczynski, R. Holz, and G. Carle, "In Log We Trust: Revealing Poor Security Practices with Certificate Transparency Logs and Internet Measurements," in *Passive and active measurement*, ser. LNCS sublibrary. SL 5, Computer communication networks and telecommunications, R. Beverly, G. Smaragdakis, and A. Feldmann, Eds. Cham, Switzerland: Springer, 2018, pp. 173–185.

[6] J. Gustafsson, "Certificate Transparency in Theory and Practice," Ph.D. dissertation, 2016, http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A909303&dswid=3471, [Online; accessed: 17/06/2020].

[7] B. Laurie, A. Langley, and E. Kasper, "RFC 6962 - Certificate Transparency," https://tools.ietf.org/html/rfc6962, 2013, [Online; accessed: 11/06/2020].

[8] "How Log Proofs Work - Certificate Transparency," https://www.certificate-transparency.org/log-proofs-work, [Online; accessed: 29/06/2020].

[9] C. Nykvist, L. Sjöström, J. Gustafsson, and N. Carlsson, "Server-Side Adoption of Certificate Transparency," in *Passive and active measurement*, ser. LNCS sublibrary. SL 5, Computer communication networks and telecommunications, R. Beverly, G. Smaragdakis, and A. Feldmann, Eds. Cham, Switzerland: Springer, 2018, pp. 186–199.

[10] "Browser Market Share - May 2020 - NetMarketShare," https://netmarketshare.com/?options={"filter":{},"dateLabel":"Custom","attributes":"share","group":"browser","sort":{"share":-1},"id":"browsersDesktop","dateInterval":"Monthly","dateStart":"2020-05","dateEnd":"2020-05","segments":"-1000"}, 2020, [Online; accessed: 10/06/2020].

[11] D. O'Brien, "Certificate Transparency Enforcement in Google Chrome–Google Groups," https://groups.google.com/a/chromium.org/forum/#!msg/ct-policy/wHILiYf31DE/iMFmpMEkAQAJ, 2018, [Online; accessed: 11/06/2020].

[12] "Certificate Transparency in Chrome," https://github.com/chromium/ct-policy/blob/master/ct_policy.md, 2019, [Online; accessed: 11/06/2020].

[13] "Qualified Logs Chromium," https://github.com/chromium/ct-policy#chromium-certificate-transparency-policy, 2020, [Online; accessed: 11/06/2020].

[14] "Apple's Certificate Transparency policy," https://support.apple.com/en-gb/HT205280, 2019, [Online; accessed: 11/06/2020].

[15] R. Jha, "Certificate Transparency," https://docs.microsoft.com/en-us/archive/blogs/azuresecurity/certificate-transparency, 2018, [Online; accessed: 12/06/2020].

[16] B. Li, J. Lin, F. Li, Q. Wang, Q. Li, J. Jing, and C. Wang, "Certificate Transparency in the Wild," in *CCS'19*, L. Cavallaro, J. Kinder, X. Wang, J. Katz, L. Cavallero, and X. Wang, Eds. New York, NY: Association for Computing Machinery, 2019, pp. 2505–2520.

[17] "Implement Certificate Transparency support (RFC 6962)," https://bugzilla.mozilla.org/show_bug.cgi?id=1281469, [Online; accessed: 11/06/2020].

[18] "Certificate Transparency Signature Verifications Negatively Impact TLS Handshake Performance," https://bugzilla.mozilla.org/show_bug.cgi?id=1353216, [Online; accessed: 11/06/2020].

[19] "Expect-CT," https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expect-CT, 2020, [Online; accessed: 11/06/2020].

[20] "when will Mozilla support Expect-CT, a new security header," https://support.mozilla.org/de/questions/1268433, [Online; accessed: 30/06/2020].

[21] "Certificate Transparency Log Policy," https://github.com/chromium/ct-policy/blob/master/log_policy.md, 2017, [Online; accessed: 11/06/2020].

[22] "Apple's Certificate Transparency log program," https://support.apple.com/en-om/HT209255, 2019, [Online; accessed: 15/06/2020].

[23] "Issue 780654: Certificate Transparency - Cloudflare "nimbus2018" Log Server Inclusion Request," https://bugs.chromium.org/p/chromium/issues/detail?id=780654#c14., 2017, [Online; accessed: 11/06/2020].

[24] E. Stark, "Expect-CT Extension for HTTP," https://github.com/bifurcation/expect-ct/blob/master/draft-stark-expect-ct.md, 2017, [Online; accessed: 14/06/2020].

[25] R. Dahlberg, T. Pulls, J. Vestin, T. Høiland-Jørgensen, and A. Kassler, "Aggregation-based gossip for certificate transparency," 2019.

[26] L. Nordberg, D. Gillmor, and T. Ritter, "Gossiping in CT - draft-ietf-trans-gossip-05," https://tools.ietf.org/html/draft-ietf-trans-gossip-05, 2018, [Online; accessed: 16/06/2020].

[27] "Google/Certificate-Transparency-Go," https://github.com/google/certificate-transparency-go/blob/5690bed5b44db4a1b17fae99187a8bf4dc69830a/gossip/minimal/README.md, 2019, [Online; accessed: 15/06/2020].

[28] "Command goshawk," https://godoc.org/github.com/google/certificate-transparency-go/gossip/minimal/goshawk, [Online; accessed: 18/06/2020].

[29] Q. Scheitle, O. Gasser, T. Nolte, J. Amann, L. Brent, G. Carle, R. Holz, T. C. Schmidt, and M. Wählisch, "The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 343–349.

[30] "CA/CT Redaction," https://wiki.mozilla.org/CA/CT_Redaction, [Online; accessed: 01/07/2020].

[31] R. Stradling, "Certificate Transparency: Domain Label Redaction draft-strad-trans-redaction-01," https://tools.ietf.org/html/draft-strad-trans-redaction-01, 2017, [Online; accessed: 11/06/2020].

[32] "Wildcard certificate," https://en.wikipedia.org/wiki/Wildcard_certificate, [Online; accessed: 30/06/2020].

[33] T. Quinn, "Name Constraints in x509 Certificates," https://timothy-quinn.com/name-constraints-in-x509-certificates/, [Online; accessed: 30/06/2020].

[34] V. Podāns, "X.509 Name Constraints certificate extension – all you should know," https://www.sysadmins.lv/blog-en/x509-name-constraints-certificate-extension-all-you-should-know.aspx, [Online; accessed: 30/06/2020].

[35] E. Messeri, "Overview of changes between RFC6962 and RFC6962bis," https://www.certificate-transparency.org/ct-v2-rfc6962-bis, [Online; accessed: 30/06/2020].

[36] B. Laurie, A. Langley, E. Kasper, E. Messeri, and R. Stradling, "Certificate Transparency draft-ietf-trans-rfc6962-bis-14," https://tools.ietf.org/html/draft-ietf-trans-rfc6962-bis-14, [Online; accessed: 30/06/2020].

[37] ——, "Certificate Transparency Version 2.0 draft-ietf-trans-rfc6962-bis-34," https://tools.ietf.org/html/draft-ietf-trans-rfc6962-bis-34, [Online; accessed: 30/06/2020].

[38] "Issue 389514: Certificate Transparency: Inclusion of Google's "Aviator" log," https://bugs.chromium.org/p/chromium/issues/detail?id=389514, 2014, [Online; accessed: 20/06/2020].

[39] "Certificate Transparency Monitor - Graham Edgecombe," https://ct.grahamedgecombe.com/, 2020, [Online; accessed: 15/06/2020].

[40] Brendan McMillion, "Post-Mortem: Nimbus issuing bad SCTs - Google Groups," https://groups.google.com/a/chromium.org/forum/#!searchin/ct-policy/Nimbus$20issuing$20bad$20SCTs%7Csort:date/ct-policy/E88pjOZzkIM/-uphL2fqBQAJ, 2018, [Online; accessed: 06/06/2020].

[41] "Cloudflare Nimbus 2019 - Graham Edgecombe," https://ct.grahamedgecombe.com/logs/53, [Online; accessed: 20/06/2020].

[42] "Google Argon 2019 - Graham Edgecombe," https://ct.grahamedgecombe.com/logs/43, [Online; accessed: 20/06/2020].

doi: 10.2313/NET-2020-11-1_09