2018-10-01

Certificate Revocation Table: Leveraging Locality of Reference in Web Requests to Improve TLS Certificate Revocation

Luke Austin Dickinson
Brigham Young University

Certificate Revocation Table: Leveraging Locality of Reference in Web Requests to Improve TLS Certificate Revocation

Luke Austin Dickinson

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Kent Seamons, Chair
Daniel Zappala
Casey Deccio

Department of Computer Science

Brigham Young University

ABSTRACT

Certificate Revocation Table: Leveraging Locality of Reference in Web Requests to Improve TLS Certificate Revocation

Luke Austin Dickinson
Department of Computer Science, BYU
Master of Science

X.509 certificate revocation defends against man-in-the-middle attacks involving a compromised certificate. Certificate revocation strategies face scalability, effectiveness, and deployment challenges as HTTPS adoption rates have soared. We propose Certificate Revocation Table (CRT), a new revocation strategy that is competitive with or exceeds alternative state-of-the-art solutions in effectiveness, efficiency, certificate growth scalability, mass revocation event scalability, revocation timeliness, privacy, and deployment requirements. The CRT periodically checks the revocation status of X.509 certificates recently used by an organization, such as clients on a university's private network. By prechecking the revocation status of each certificate the client is likely to use, the client can avoid the security problems of on-demand certificate revocation checking.

To validate both the effectiveness and efficiency of using a CRT, we used 60 days of TLS traffic logs from Brigham Young University to measure the effects of actively refreshing certificates for various certificate working set window lengths. Using a certificate working set window size of 45 days, an average of 99.86% of the TLS handshakes from BYU would have revocation information cached in advance using our approach. Revocation status information can be initially downloaded by clients with a 6.7 MB file and then subsequently updated using only 205.1 KB of bandwidth daily. Updates to this CRT that only include revoked certificates require just 215 bytes of bandwidth per day.

Keywords: certificate revocation, X.509 certificate, caching, working sets, locality of reference

ACKNOWLEDGMENTS

I would like to acknowledge my wife for her support during graduate work and research. I would like to thank my advisor, Kent Seamons, for the numerous discussions we had exploring and simplifying the design used this thesis research. Additionally, I would like to thank the network administrators at BYU who supported our research with real world data, as our research could not have been done without them.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet depends on X.509 certificates for encrypted HTTPS traffic. While the TLS protocol natively gives message confidentiality and integrity, message authentication relies on the web's Public Key Infrastructure (PKI) and correct use of X.509 certificates. X.509 certificates include fields that a client should expect in the TLS handshake, such as the public key, and these certificates are usually signed by a trusted Certificate Authority (CA). Browsers use these fields to validate they are communicating with the correct host.

When a certificate's private key is stolen or leaked, guarantees of information protection given by HTTPS are lost. Certificate revocation is used to address this situation by blacklisting a certificate. A website administrator can request the issuing CA to revoke, or invalidate, their certificate. Notice of the revocation needs to be disseminated to all clients who may interact with this certificate, otherwise connections to compromised websites are vulnerable to a man-in-the-middle attack until the certificate has expired. The lifetime of a certificate typically ranges from 3 months to 3 years. Without a viable certificate revocation strategy, malicious parties can impersonate a compromised website undetected and glean private information such as personally identifying information, passwords, and personal emails.

Problems with current revocation strategies have limited the use of certificate revocation checking, even though checking the revocation status of a certificate is required to mitigate attacks using a compromised, but otherwise valid certificate. Most TLS certificate revocation strategies consume a relatively large amount of client bandwidth

[12], expose client traffic patterns [29], protect a small percentage of certificates [1, 3], or are vulnerable to downgrade attacks when access to revocation information is critical [12, 16, 29]. The remaining strategies have other debilitating weaknesses slowing and halting adoption such as requiring significant infrastructure changes [34, 36], requiring participation and additional costs by CAs and/or third parties [25, 36], or exposing new attack surfaces [2]. Because there is no revocation strategy without a major cost or flaw, certificate revocation is largely being ignored by most clients, leaving many computers and smartphones vulnerable to man-the-middle attacks [26].

In this work, we propose Certificate Revocation Table (CRT) as a new certificate revocation status dissemination strategy to regularly check the revocation status of X.509 certificates recently used by an organization, such as clients on a university's private network. Network administrators are able to deploy this strategy at scale to protect their own clients. Clients access this revocation information through either a downloadable file or an on-demand API. By taking advantage of an organization's and clients' behavior to regularly return to many of the same websites, our solution is able to provide the needed revocation status of a high percentage of TLS handshakes while minimizing bandwidth and vulnerability concerns.

Evidence has been shown [10] that a vast majority of TLS handshakes made by a group of similar clients will reuse recently seen certificates over the space of an hour. We find that this idea continues to hold true even when a client's cache includes certificates originally used many hours previously, if the cache includes multiple weeks of data. Because effectiveness is maintained even in the presence of this delay, we are able to design a flexible revocation strategy in which administrators are able to adjust the rate clients receive updates from a supporting server based on their organization's needs, while still protecting clients. Because even a large private network, such as a university network [8, 19], will only use a small fraction of the globally available X.509 certificate space, the bandwidth consumption of our design is competitive with alternative solutions with higher data density.

The CRT's design addresses seven concerns facing certificate revocation strategies today:

- **Effectiveness: Vulnerable to Downgrade Attacks**— When a revocation status cannot be determined because the authoritative server is not accessible, modern browsers will "soft-fail" by assuming that the certificate in question has not been revoked. Soft-failing values accessibility over protecting against a potentially revoked certificate. This is dangerous because an attacker who is in a man-in-the-middle position to serve a revoked certificate to end clients can trivially block these clients from downloading the revocation status needed to detect the malicious certificate. Langley [24] stated, "soft-fail revocation checks are like a seat-belt that snaps when you crash."

  CRT allows clients to avoid soft-failing their revocation checks without affecting accessibility by maintaining a cache containing the revocation status of certificates likely to be used again. Predownloading revocation information in advance [1, 3, 25] reduces the number of vulnerable on-demand checks required. To successfully use a revoked certificate whose status is provided by this kind of strategy, malicious actors need to block clients from access to these endpoints for an extended period of time in contrast to only requiring to block access to a revocation endpoint at the time of the malicious connection. We expect this will make it more difficult for attackers to remain undetected.

- **Efficiency: Bandwidth Consumption**— The bandwidth consumption to check the revocation status for a certificate or group of certificates can be from one kilobyte to many megabytes depending on the revocation strategy used. To maintain a sufficiently up-to-date certificate revocation status, strategies repeat these downloads or requests from once a week to many times per day. This cost can limit participation of clients, especially those with a monthly bandwidth limit. Liu et al. [26] found that no mobile browser evaluates any certificate revocation status and suggests bandwidth concerns are to blame.

A CRT provides the revocation status certificates that clients will likely use in the near future, i.e. an organization's certificate working set. While this size scales to the organization a CRT is protecting and parameters set by the network administrators, our experiments found that revocation status information can be initially downloaded by clients with a 6.7 MB file and then subsequently updated using only 205.1 KB of bandwidth daily. If the CRT only sends notifications of revoked certificates to clients, the daily bandwidth is only 215 bytes.

- **Certificate Growth Scalability**— Over the last few years HTTPS usage has grown tremendously. One reason for this growth is the emergence of a new CA, Let's Encrypt, which allows certificates to be issued for free through an automated system. In just over a year, from August 2017 to September 2018, the number of active trusted certificates signed by Let's Encrypt has increased by sixfold, rising from 34 million [6] to 213 million [4]. Because of the growth of Let's Encrypt and other certificate authorities, the live CA-trusted certificates on the Internet more than doubled in one year from January 2017 (30 million [25]) to December 2017 (78.5 million [4]), and then quadrupled again in just over half of a year from December 2017 (78.5 million [4]) to September 2018 (317 million [4]).

  The growth of TLS activity places more demands on the scalability of revocation strategies than ever before. The daily bandwidth requirement of some certificate revocation strategies, such as CRLite, increases with the number of global CA-trusted TLS certificates [25]. Other strategies avoid the growth in size by opting to limit the number of certificates they protect as with CRLSets [1].

The daily bandwidth required by a CRT and its clients only grow based on the number of certificates used by the clients. Global certificate adoption only indirectly affects a CRTs bandwidth requirements because websites previously visited by clients may begin to use TLS.

- **Mass Revocation Event Scalability**— While revocation strategies should maintain acceptable bandwidth requirements during normal conditions, it is also imperative that the revocation strategy can gracefully handle mass revocation events such as the period following the announcement of the Heartbleed vulnerability[1]. In 2014, Liu et al. [26] found that before the announcement of Heartbleed, approximately 1% of fresh certificates (non-expired certificates signed by a trusted CA) were revoked. After the announcement, the revocation percentage rose to over 8%. During this time, Cloudflare estimated they would incur an additional $400,000 per month to publish their enlarged CRL due to the increased requisite bandwidth [32]. After measuring the effects of Heartbleed, Durumeric et al. stated, "The community needs to develop methods for scalable revocation that can gracefully accommodate mass revocation events, as seen in the aftermath of Heartbleed" [14].

  During a mass revocation event, the bandwidth consumption of a CRT will not change because CRT already provides the revocation status of each certificate in the working set regardless of whether it has been revoked or not. Delta updates downloaded by clients will be more expensive while a large number of certificates are being revoked, but will return to their expected size soon after. This is not true for other revocation strategies such as CRLs [12] or CRLite's full Bloom filter cascade [25] whose bandwidth requirements are based on the number of globally valid, but revoked certificates. These strategies will maintain this significant increase in bandwidth until the revoked certificates begin to expire.

[1] http://heartbleed.com

- **Revocation Timeliness**— Revocation timeliness is the elapsed time from when a CA revokes a certificate to when end clients learn of the revocation. Some strategies allow several days before updating clients. The median CRL has a lifetime, and therefore suggested update rate, of 7 days [25] and the median OCSP response for the Alexa top 1 Million has a lifetime, and therefore suggested update rate, of 4 days [25]. A push-based revocation strategy that updates clients once a day [1, 25] actually has a worst-case revocation timeliness of almost 48 hours (see Section 4.2).

  CRT allows administrators to balance between revocation timeliness and server bandwidth consumption as needed by the organization through design parameters.

- **Privacy: Exposing Client Traffic Patterns**—

  Some revocation strategies such as OCSP [29] share detailed client traffic patterns to revocation endpoints.

  A CRT does not not require clients to expose their browsing history to third parties to receive revocation information.

- **Deployment Requirements and Incentives**—

  Difficult deployment requirements and weak incentives can slow or halt adoption of a revocation strategy. For example, while initially CRLite had acceptable deployment requirements in January of 2017, the global certificate growth has increased the difficulty to deploy this solution. As of September 2018, we estimate CRLite would require at least 2,465 OCSP requests per second to be made to Let's Encrypt's OCSP endpoints alone (see Section 6.1). This would require coordination between the Let's Encrypt and the CRLite system in order to avoid overwhelming Let's Encrypt's OCSP responders. OCSP Must-Staple faces even stronger deployment hurdles as it requires certificate owners to commit to regularly obtain OCSP status from historically unreliable OCSP responders. If mistakes are made, websites become unavailable to clients which disincentivizes website administrators to adopt OCSP Must-Staple.

CRT has strong deployment incentives as network administrators who would deploy a CRT already have an invested interest in protecting the clients in their organization. Additionally, CRT is able to be deployed at scale based on the number of certificates used in the organization. From an experiment where we daily rechecked the revocation status of all X.509 certificates used on the BYU campus network within 45 days (see Section 4.5.5), we found the server bandwidth consumption and required OCSP requests of CRT to be 3 orders of magnitude lower than CRLite.

1.1 Contributions and Outline

In this work, we make two important contributions to better understand and take advantage of certificate revocation systems that leverage reference locality:

- We provide a novel certificate revocation strategy which is competitive with alternative state-of-the-art strategies. This strategy enables system and network administrators to protect their own clients without relying on third parties. We describe how the privacy of clients' browsing patterns can be preserved through alternative deployment options.
- We are the first study to examine how X.509 certificate working sets change over time for all clients in an organization and for individual clients. We examine passively collected TLS traffic logs from a university network over a 2-month period and measure the effectiveness of our revocation strategy. Our results shed light on the effectiveness of our certificate revocation strategy as well as other certificate revocation and validation strategies [10, 21].

In the following chapters, we summarize previous work analyzing certificate working sets and certificate revocation. Next, we present the system design of a certificate revocation table, including the operations and data items it contains. Using 60 days of traffic logs from our university, we then measure and report the effectiveness and efficiency of a reference implementation of a CRT. In these experiments we explore the effects of various design parameters, including using aggregated TLS traffic from many clients in an organization and using TLS traffic from only a single client. We discuss these results and suggest implementation options. Using the data from our measurements, we next evaluate our reference implementation against each of the seven concerns described in introduction. We then evaluate alternative state-of-the-art solutions and discuss the strengths and weaknesses of these strategies compared to our implementations. Finally, from the results of our research, we provide the security community with future research avenues to improve X.509 revocation systems that leverage reference locality.

Chapter 2

Related Work

In this chapter, we discuss background information related to certificate working sets along with various approaches to certificate revocation, including pull-based, push-based, and network assisted.

2.1 X.509 Certificate Working Sets

A working set is a collection of all data (or in our case certificates) referenced over some time interval [13]. In 1968, Denning [13] argued that elements in a working set will be used in the near future again with a high probability, at least compared to elements outside of this working set. Because of this intuitive result, caches storing recently used items are effective in CPUs, file systems, and on the web. We aim to understand how quickly X.509 certificate working sets change and to measure their effectiveness at predicting future certificates.

To understand how certificates are actually being used in the wild, previous studies have passively examined real traffic. While various studies have passively measured certificate use [7, 8, 10, 19, 21], only two studies reported any information referring to a certificate working set's effectiveness to predict future traffic. Bates et al. [10] found that 99.3% of HTTPS requests originating from a college campus were to repeated websites over a period of an hour. While examining the effects of caching OCSP responses, Hu et al. [21] found that 95% of their certificates at their university were repeated over 30 minutes.

While the results from these studies have motivated our research, many questions have been unanswered about certificate working sets. In this work, we expand these previous studies' results by examining the effects of increasing the working set window length to multiple days, introducing up-to a 24-hour delay to deliver processed revocation information to clients, and examining the working set of individual clients.

2.2 Certificate Revocation

There have been many certificate revocation proposals the past two decades. Strategies for certificate revocation generally fall into three groups: Pull, Push, and Network-assisted Revocation.

2.2.1 Pull-Based Certificate Revocation

Clients using a pull-based certificate revocation strategy request revocation information only at the time a revocation status needs to be confirmed. Among pull-based certificate revocation strategies, Certificate Revocation Lists (CRLs) [12] and Online Certificate Status Protocol (OCSP) [29] are the most commonly used today.

CRLs [12] are lists of all revoked certificates that the issuing CA assembles, signs, and distributes. Scalability is the main criticism against CRLs, because they can grow quite large. For example, Apple published a 76 MB CRL [26]. Due to the potential bandwidth and latency concerns, Mozilla Firefox and Google Chrome have disabled revocation checking using CRLs.

OCSP [29] responders provide a signed certificate revocation status for individual certificates. OCSP requires clients to make this request for every web session initiated through HTTPS and must wait for the OCSP response to return before the page can be fully loaded. In addition to page load delays, OCSP presents a significant privacy concern as each client divulges its browsing history to a third party. While some revocation strategies only share coarse-grained traffic patterns [12], OCSP divulges detailed client traffic patterns to the CA who signed the certificate and all the nodes along the unencrypted OCSP request's path through the Internet. Finally, OCSP requests naturally soft-fail because clients make these requests at the same time the TLS handshake needs to be validated. Despite these concerns, most modern desktop browsers use OCSP.

In order to remove the page loading delays created by OCSP and TLS in general, Stark el al. [35] use content-based prefetching to gather parameters required for TLS Snap Start [23], which reduces prepared TLS handshakes to zero round trips. Clients perform OCSP requests to check the revocation status of the certificates predicted to be used next. While this can drastically reduce latency compared to no prefetching efforts, this revocation strategy suffers the same soft-failure problems as traditional OCSP because revocation status checking only occurs seconds before it is needed. Our research expands on this work by refreshing the revocation status of many certificates, which are predicted to be used again within several days, at least once every 24 hours. This is done to increase the difficulty of a successful man-in-the-middle attack.

Depending on the implementation, clients will cache these pull-based revocation requests to improve access times and availability. If a request is not cached or is expired, the client request introduces page delays and resorts to a soft-fail revocation check. Any issue in availability, malicious [27] or not, will force the client to soft-fail. Because uncached soft-fail revocation checks allow a man-in-the-middle attack to go unnoticed, this type of certificate revocation offers little protection.

2.2.2 Push-Based Certificate Revocation

Clients using a push-based strategy regularly receive revocation information at periodic intervals, such as once a day. In contrast to pull-based strategies, this does not reveal client traffic patterns as all clients receive the same payloads. Since clients using pull-based strategies download and cache data ahead of time, there is a higher probability that a client will be able to rely on the cached status information at the time of a TLS handshake.

Compared to pull-based revocation strategies, the status of more certificates will be collected and updated than a client will use.

Some revocation strategies minimize this additional bandwidth consumption by only including a small, hand-picked set of important revocations. Both Google's CRLSets and Mozilla's OneCRL fit into this category of selective, push-based revocation strategies. Google pushes periodic updates to Chrome with a small list of revoked certificates called a CRLSet. Millions of cliensts trust and use CRLSets via the Chrome browser. CRLSets have a maximum size of 250 KB [1], which equates to a capacity of about 40,000 revoked certificates. The intent of a CRLSet is to only include revocations where the risk of a compromised certificate is suspected[1]. Mozilla produces a similar revocation list called OneCRL [3]. Instead of filtering through leaf certificates, OneCRL includes only revoked intermediate certificates, which would have a much more significant impact if abused.

Another method to minimize bandwidth consumption is to use a more efficient data structure, such as a Bloom filter. A Bloom filter is an append only data structure in which false negatives are not possible but false positives are. Rabieh et al. [33] showed how using two Bloom filters in tandem, one stating if a certificate is not revoked and one stating if a certificate is revoked, drastically reduced false positive rates. In rare cases where the Bloom filters label a certificate both revoked and not revoked, an on-demand request is made.

Larisch et al. [25] presented CRLite, a revocation strategy that uses a Bloom filter cascade that allows a server to push the status of all live certificates across the Internet in a compressed deterministic data structure to clients. A Bloom Filter cascade is created by feeding the possibly false positive queries from one Bloom filter into another testing for the opposite value. The algorithm repeats this process until a Bloom filter is found with no false positive entries. The server requires substantial network resources to build

[1] Many certificates are revoked despite no suspicions of compromise exist. We found 88% of revoked certificates with a revocation reason reported "Cessation Of Operations" which is to indicate a certificate should not be used but it was not compromised (See Table A.4).

the filter as the entire set of both non-revoked and revoked certificates are used to create the data structure. In January 2017, the Bloom filter cascade was only 10 MB, with daily updates averaging 580 KB. Using the data we collected (see Appendix A) in March of 2018, we found the data structure had grown to 18.0 MB[2].

2.2.3 Network Assisted Certificate Revocation

Network-assisted revocation strategies eliminate the need for a client to request a revocation status, instead modifying the TLS ecosystem to address revocation.

One approach is through a middlebox. Revocation in the Middle (RITM) [36] is a solution that distributes revocation information to middleboxes throughout the Internet via a CDN. These middleboxes maintain a cache of the revocation status of all certificates, As the middlebox intercepts traffic, it checks the revocation status and appends it to the handshake as part of a TLS extension. Use of one of these middleboxes eliminates the latency of a separate revocation status check since the middlebox is along the route of the connection. However, it requires installing middleboxes throughout the Internet and requires potentially costly CDN access to update these systems. This approach also requires clients and servers to adopt a new TLS extension.

As an alternative middlebox strategy, Hu et al. [21] proposed Certificate Revocation Guard (CRG), which uses a middlebox to intercept all TLS traffic for an entity such as an organizational gateway. This middlebox uses its cache, or on a cache miss performs an OCSP request, to check a certificate revocation status. If the middlebox detects a revoked certificate, then a malformed certificate is returned to the client, effectively blocking the connection. This strategy does not require clients to make any modifications to participate. Alternatively, this strategy could be run as middleware on an individual client. While the paper suggests 95% of TLS handshakes will be able to use a cached

[2] We used source code as provided by the authors. CRLite's full Bloom filter cascade size is strongly dependent on the number of revoked certificates (19.1 M) and the number of total certificates (86.2 M). If a CRLite Bloom filter cascade only includes revocations from publicly used certificates (1.1 M), the size is reduced to 1.93 MB.

revocation status, the paper only reports the results of a short 30-minute experiment involving around 2,000 connections. Because this strategy also takes advantage of the certificate working sets of clients on a private network, the results from our experiments can be used to better evaluate the effectiveness of this strategy. However, by nature of using a middlebox, mobile clients such as laptops and smartphones will lose protection if they leave the network.

OCSP Stapling [16], proposed in 2001, requires each website administrator, instead of end clients, to make an OCSP request for their certificates. The web server transmits the revocation status to each client during TLS handshakes. While this eliminates the page load delay and the privacy concerns of traditional OCSP, OCSP Stapling is still vulnerable to man-in-the-middle attacks since an attacker can simply choose to not include the OCSP Staple in their handshake with the client.

OCSP Must-Staple [17] was proposed in 2015 to remedy the concerns of OCSP Stapling. An X.509 certificate extension signals the browser to break the handshake if the OCSP Staple is missing. The certificate extension removes the man-in-the-middle vulnerability at the cost of requiring server administrators to commit to always providing an updated OCSP Staple. Failure to do so will result in their website becoming inaccessible to clients.

Full adoption of OCSP Must-Staple also opens the Internet to a dangerous situation if proper infrastructure is not in place ahead of time. An OCSP responder hit with an extended Denial of Service (DoS) attack could block access to a large portion of the Internet since server administrators would not be able to acquire the requisite OCSP Staple. Due to these concerns, Google Chrome currently does not support the certificate extension for OCSP Must-Staple [2, 38]. Others [26] have argued that the potential for a DoS attack is not a fundamental problem as a CDN could distribute static revocation information. OCSP Must-Staple has also suffered other problems, including CA inconsistencies and bugs in server implementations, that have slowed adoption [11]. While currently only

0.032% of live certificates use OCSP Must-Staple [5], reporting mechanisms such as OCSP Expect-Staple [18] may accelerate adoption of this revocation strategy.

Lastly, using short-lived certificates is a strategy that eschews revocation checking. Instead of checking a revocation status, certificates are set to expire shortly after issuance, generally ranging from a matter of hours [20] to just a few days [37]. This strategy requires the server to regularly renew its certificate. While it was previously not practical to change public keys on renewal [37], the emergence of new technology such as the ACME protocol [9] and the EFF's CertBot[3] enables automatic public key rotation on renewal. If a private key compromise occurs, the server administrator simply does not renew the certificate. This strategy is similar to OCSP Must-Staple except that the certificate is regularly renewed instead of the OCSP Staple.

[3] https://certbot.eff.org/about/

Chapter 3

Certificate Revocation Table

Our approach to certificate revocation is inspired by memory management in operating systems. A working set W(t, τ) of a process is the collection of all data referenced by the process over the period of time t - τ to t [13]. The principle of locality states that data in a process's working set W(t, τ) will frequently be reused by a future working set W(t + α, α), as long as α is small. As α grows, the frequency of reused entries is reduced. Operating systems leverage this reuse to reduce Translation Lookaside Buffer (TLB) misses and page faults, for improved CPU throughput.

We hypothesize that the X.509 certificate working set of an organization, and to a lesser degree a single client, will follow this same principle of locality. A certificate working set W(t , τ) will contain a majority of the certificates in the working set W(t + α, α).

A certificate revocation table (CRT) (see Figure 3.1) collects and manages an organization's certificate working set. It periodically rechecks the revocation status of each certificate in this working set by querying a revocation endpoint, such as an OCSP responder or a CRL endpoint. The CRT uses the parameters β, τ, and α to set the rate at which the CRT's periodic functions are invoked (see 4.2 for more details). Clients access the revocation information of the organization's certificate working set through either a downloadable file or an on-demand API. By periodically downloading this information, clients reduce the possibility of failing their revocation check due to an occasionally inaccessible revocation endpoint or due to malicious activity.

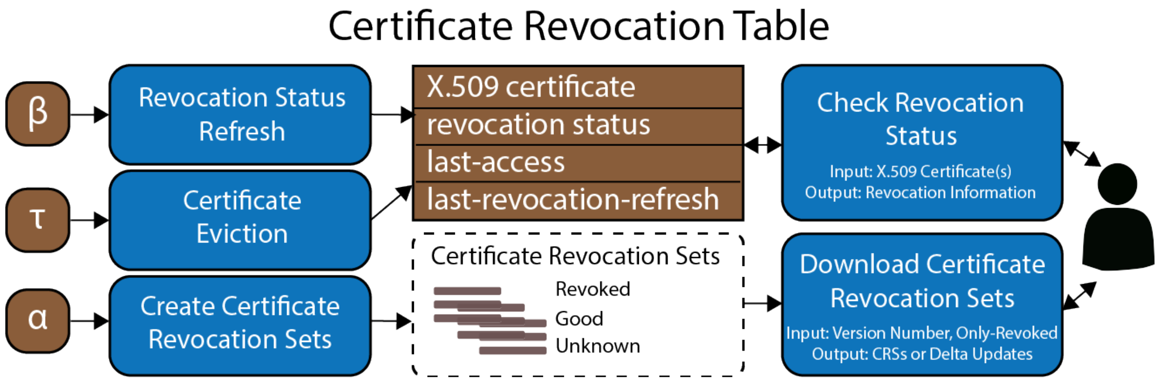
Certificate Revocation Table
β
τ
α
Revocation Status Refresh
Certificate Eviction
Create Certificate Revocation Sets
X.509 certificate
revocation status
last-access
last-revocation-refresh
Certificate Revocation Sets
Revoked
Good
Unknown
Check Revocation Status
Input: X.509 Certificate(s)
Output: Revocation Information
Download Certificate Revocation Sets
Input: Version Number, Only-Revoked
Output: CRSs or Delta Updates

Figure 3.1: The design of a certificate revocation table. Periodically, the CRT runs the Revocation Status Refresh, Certificate Eviction, and Create Certificate Revocation Sets functions. Clients access revocation information through the Download Certificate Revocation Sets and Check Revocation Status APIs.

In the rest of this chapter, we first detail the purpose of a certificate revocation table (CRT). Next, we address the periodic tasks used to maintain the CRT. Then, we discuss how clients interact with the system to provide their certificate usage and retrieve revocation information. Finally, we analyze the security of the system in relation to our threat model.

3.1 Certificate Revocation Table Design

The purpose of a certificate revocation table (CRT) is to connect clients together to create a single, shared certificate working set. At regular intervals, a CRT updates or removes existing table entries. Incoming certificates from clients are inserted into the CRT and later each revocation status is packaged for end clients.

Each row in a CRT contains an X.509 certificate, its revocation status, and two timestamps: last-access and last-revocation-refresh. The certificate field does not need to be the entire X.509 certificate, only enough to uniquely identify it and any information required to check the revocation status[1]. The revocation status reports the last received status which can be Good, Revoked, Unknown, or New. Certificates are marked as "New"

[1] If OCSP is used for revocation, this includes information from an issuer certificate.

if the CRT has not yet attempted a revocation status request and "Unknown" if the CRT has never successfully retrieved a revocation status. Each time a certificate is inserted into the CRT, the last-access timestamp is updated to the current time. Last-revocation-refresh is updated to the current time anytime the revocation status is accessible. When sending information to the client, this timestamp accompanies an out-of-date status (inaccessible for longer than β; see 4.3.1) to inform clients how long the status has been inaccessible.

3.2 Periodic System Tasks

Each of the following tasks are periodically run to update a CRT and to prepare the revocation information that will be used by the clients via the APIs.

3.2.1 Revocation Status Refresh

The CRT rechecks the revocation status of each certificate it contains every β hours[2]. Decreasing β will increase revocation status freshness and a CRT's bandwidth consumption.

The last-revocation-refresh field is used to manage revocation status rechecking. The revocation status of the certificate is checked when a certificate is first added to the CRT, or when the row in the CRT has a last-revocation-refresh timestamp that is greater than or equal to β hours from the current time. Sometimes a revocation endpoint, such as an OCSP responder, will be inaccessible. In this case, each subsequent retry should introduce a delay before attempting the request again to avoiding an accidental DoS attack on a revocation endpoint. Some form of exponential backoff is recommended.

3.2.2 Certificate Eviction

Evicting unused certificates from a CRT increases the efficiency of the revocation model. First, revocation status requests occur regularly for every certificate in the CRT. Reducing

[2] In practice, most revocations last for the lifetime of certificate. However, there are times a revocation can be reverted [12]. For this reason, the revocation status of a revoked certificate should be rechecked occasionally.

the number of requests reduces the bandwidth consumption of a CRT. Second, the amount of storage needed by a CRT and end-client CRSs can be reduced. Both of these values can grow without bound if certificates are never evicted.

Periodically, a CRT evicts certificates that are no longer in the organization's certificate working set[3]. To measure what is in the organization's certificate working set, we recall that a working set at time t is defined as W(t, τ), which includes all active certificates from the period of time t - τ to t. Applying this to our model, we evict all certificates with a last-access timestamp older than t - τ.

The window length of the working set, τ, affects both the effectiveness to maintain each up-to-date revocation status and cost to store them. As τ increases, we expect the effectiveness of using a CRT to grow while the efficiency is reduced.

Additional certificates can be evicted from the CRT if they are unlikely to be used again. The simplest example is removing certificates that have expired from the certificate working set. Clients do not trust expired certificates, therefore, they do not need to know the revocation status to avoid a man-in-the-middle threat. We believe analyzing when certificates in the certificate working set become inactive may reveal other early eviction strategies to increase efficiency. We leave the exploration of these eviction strategies to future work.

3.2.3 Create Certificate Revocation Sets

Once every α hours, the CRT projects its revocation information into multiple certificate revocation sets (CRSs), one for each revocation status option i.e. Good, Revoked, and Unknown. End-clients regularly download these CRSs. As stated by the principle of locality, we expect a working set W(t, τ), a group of CRSs, will frequently be reused by a future working set W(t + α, α), the certificates that will be seen by clients before the next group of CRSs, as long as α is small. While a smaller α will likely increase the

[3] In practice, extra care should be considered before evicting a known revoked certificate.

overlap between working sets, the α value also defines how often clients should download updated CRSs. In our experiments, we use an α value of 24 hours.

Each CRS includes certificates with a revocation status of Good, Revoked, or Unknown. To include an out-of-date 'Good' revocation status (inaccessible for longer than β; see 4.3.1), the CRS includes the last-revocation-refresh timestamp to give clients necessary data to decide to avoid a suspicious connection. By including each of these CRSs, clients can be certain which certificates they can trust, which ones are revoked, which ones are currently inaccessible, which ones have never been accessible to the CRT, and which certificates are completely new. Finally, this data structure contains a small header including a timestamp, a version number, and a digital signature.

Lastly, as with other strategies such as CRLite [25] and delta CRLs [12], a delta update can be calculated to reduce the amount clients need to download per α hours. The delta includes any additions or subtractions from the previous data structure. The deltas also include header fields with a timestamp, a version number, and a digital signature. A client downloads all deltas they have missed or the entire data structure, whichever is smaller.

3.3 System APIs

A CRT provides two APIs to allow clients to insert certificates into a CRT and/or to provide clients with access to relevant revocation information.

3.3.1 Download Certificate Revocation Sets

End clients are able to download differing versions of CRSs depending on their needs[4]. In this API, there are two optional parameters, the client's current version number, and an only-revoked boolean value. If neither parameter is set, CRSs are sent to the client for revoked, non-revoked, and unknown certificates. If the version number is set, either all

[4] In practice, this API could be replaced with a push-based model.

deltas from the client's version to the most up-to date version or the group of CRSs are given to the client, whichever is smaller. If the only-revoked boolean is set to true, only the CRS or group of deltas including revoked certificates is downloaded. For the majority of clients today who soft-fail on an unknown revocation status[26], only downloading the CRS for revoked certificates is as effective as downloading all three CRSs (revoked, non-revoked, and unknown).

Locally checking a certificate's revocation status with a CRS provides revocation information even if an active attacker is manipulating a client's Internet connections. Notably, these CRSs will not always contain the revocation status of future traffic because new certificates are created in between updates and because clients may visit new websites. For certificates that are not included in these CRSs, clients must fall back to alternative validation methods.

3.3.2 Check Revocation Status

Check Revocation Status is a two-purpose API that clients use to query a CRT for the revocation status of one or more certificates[5]. The first purpose is to collect traffic information. All certificates the CRT receives through this API are either inserted into a new row or each certificate's existing row's last-access timestamp is updated. This API is the only way a CRT receives clients' certificate usage information. The second purpose is to provide clients an on-demand revocation check. When clients call this API, the CRT immediately returns the revocation status of each certificate requested as recorded in the CRT. An out-of-date revocation status is accompanied by the last-revocation-refresh timestamp. These timestamps inform clients how long a revocation status has been inaccessible. As the revocation status of newly added certificates will soon be checked and recorded (see 3.3.1), a client unwilling to trust an unknown status on first use should

[5] In practice, this API could be implicitly called by a middlebox for each intercepted TLS handshake. A middlebox adaptation of this API could append revocation information to each TLS connection [36] or automatically block revoked connections for the user [21, 30]

repeat their request to this API after a short delay. If the revocation endpoint is accessible to a CRT, a good or revoked response will be returned from the following request.

There are a few concerns that must be addressed when using an API that accepts client certificate usage. First, clients using this API are subject to the privacy concern of directly sharing their certificate usage with a CRT. This privacy concern can be addressed in various ways depending on how it is deployed (see 5.1). Second, care should be taken to enforce who can access this endpoint because all certificates sent to this API will be included in the organization's working set. If a client from outside the targeted organization uses the endpoint, the working set will possibly become polluted with certificates not used by the targeted organization.

3.4 Threat Model

The threat model includes active network attackers that can (1) modify or replay revocation information as it is transmitted over the network, and (2) block access to revocation status information. We assume that the CRT and local clients using the CRT are trusted. Attacks against either of them are out of scope.

3.5 Security Analysis

In this section, we analyze the security of the system against the two primary threats in our threat model.

3.5.1 Revocation Status Integrity

A CRT collects only signed and trusted revocation information. Assuming a benign CRT will not present false information, CRSs and on-demand responses created by a CRT will contain valid revocation information. As all information generated by the CRT includes a cryptographic signature and this signature can be verified by end clients, malicious actors cannot generate or modify revocation information undetected. In addition, because all information generated by a CRT includes a timestamp, replay attacks can be detected.

3.5.2 Accessibility to Revocation Information

An active network attacker may block access to a revocation endpoint from a CRT. Similarly, an active attacker could block a client from accessing the CRT. In either case, it is difficult to distinguish whether an active attacker is present or the system is temporarily unavailable due to a technical problem. The last-revocation-refresh data in the CRT and CRS can be used to detect an excessive refresh delay and prompt a warning for system administrators.

Chapter 4

Analysis

In this chapter, we analyze the impact of various settings for the CRT design parameters. The results illustrate the benefits of a CRT compared to alternative revocation approaches. They also inform implementers and system administrators about the effects of different values for the system parameters.

4.1 Deployment Alternatives

In our experiments, we explored two deployment alternatives for a CRT. The first utilizes an organization's shared certificate working set by sending certificate usage information from all clients in an organization to a shared CRT. This allows clients to reduce the number of possible downgrade attacks by regularly downloading CRSs. The second deployment option simplifies deployment requirements of the system by placing a CRT on the end-client device. This design does not use CRSs because the CRT is locally accessible to the end client. Because the CRT is always locally accessible in single-client systems, the end client has immediate access to changes to the CRT instead of waiting for subsequent CRS updates as in an organization-based system.

4.2 Design Parameters

There are three design parameters that influence the behavior of the CRT. These values affect various properties of the system including the effectiveness to reduce soft-fail prone TLS handshakes, bandwidth consumption, storage requirements, and revocation timeliness.

- The certificate working set window length, τ, determines when certificates will be removed from a CRT. Lengthening the window will increase the number of certificates in a CRT that will have their revocation status periodically rechecked, reducing the percentage of soft-fail prone TLS handshakes at the expense of more resources devoted to revocation status checking. However, these revocation checks are done asynchronously without requiring end clients to block while waiting for a response. We report the effects of several τ values in our experiments.
- The rate at which CRSs are created, every α hours, defines the frequency clients can retrieve updates from a remote CRT[1]. Increasing this rate improves both the revocation timeliness and effectiveness of the model, reducing the delay between the time a CRT learns of a new revocation and when a CRT delivers it to a client. However, increasing this rate requires clients to interact with a CRT more often. In our experiments α is set to 24 hours.
- The rate at which a certificate revocation status is rechecked, every β hours, in a CRT affects both the revocation timeliness and bandwidth consumption.

  The worst-case revocation timeliness[2] for an organization-based design is $\alpha + \beta$. To achieve a revocation timeliness equivalent to competing systems [1, 25] (48 hours), we set β to 24 hours in our organization-based implementation.

  To illustrate this worst-case scenario in revocation timeliness for an organization-shared CRT with an α and a β value equal to 24 hours, assume that immediately after a CRT compiles revocation information into the first day's CRS, the CRT begins rechecking revocation information in preparation for the second day's CRS.

[1] The parameter α is only relevant in organization-based designs.

[2] Any measurement of revocation timeliness assumes there are no active attackers preventing access to revocation information.

Suppose the CRT checks the status of the first certificate in its queue and finds that the certificate status is still good. The CRT marks this certificate as good for the second day's CRS. Moments later, suppose the certificate owner revokes this certificate. Clients will download the second day's CRS almost 24 hours later and will not know that this certificate is already revoked. On the third day, the CRT again checks the revocation status of each certificate, finds that this certificate has been revoked, and creates the third day's CRS. Clients will continue to trust this revoked certificate until they download the third day's CRS, which is almost 48 hours after the certificate was revoked. This worst-case scenario applies to similar revocation strategies [1, 25] that periodically provide revocation updates to clients.

Revocation timeliness in a single-client design is simply equal to the value of β. This is because an end-client directly rechecks revocation information from a revocation endpoint (i.e. a CRL endpoint or an OCSP responder) once every β hours. We set β to 24 hours in our single-client implementation.

4.3 Experiment Methodology

To measure the effectiveness and efficiency of the CRT design parameters, we obtained traffic logs from Brigham Young University's network administrators. In our experiments, we used the certificates from these logs to simulate inputs in a working CRT implementation.

4.3.1 Passive Dataset Collection

Coordinating with the network administrators at Brigham Young University, we obtained logs generated by the Bro Network Security Monitor [31] for SSL/TLS traffic, X.509 certificate information, and DHCP requests and responses from 2018-04-17 to 2018-06-17. The logs included outbound TLS traffic for the university campus network. We used DHCP logs to identify traffic generated by 1,000 anonymized individual clients.

Several precautions were taken to protect the privacy of collected data. First, non-disclosure agreements were signed by all researchers who had access to the data. Second, all of the raw data processing was performed on a computer provided and secured by the network administrators. Third, the MAC addresses used in DHCP logs were deterministically anonymized before use in the research. Fourth, all results of our experiments were reported in aggregate form, never revealing any hostnames, IP addresses, or identifying certificate information. Finally, the university's network administration reviewed our results to verify that no private information had been disclosed.

4.3.2 Representing Client Activity

We ran our experiments using all certificate traffic in our dataset to represent an organization-based design. To represent a single-client design, we repeated our experiments 1,000 times, each time using the traffic of a single anonymized client. To correlate a client and its traffic, we associated anonymized MAC addresses from DHCP logs to assigned IP addresses. If a client made no TLS handshakes during the sampled time, we excluded the data point from our results.

For our single-client experiments, we randomly selected 1,000 clients from all the clients with some TLS activity for at least 40 of the 60 days in the dataset in order to measure the effects on a client with an active CRT. We do not know if the clients with fewer than 40 days of activity used rotating MAC addresses[3], used static IP addresses, connected less often to the campus network, or simply made fewer TLS handshakes.

4.3.3 Data Analysis

Using an implementation of a CRT (see 4.4) and TLS traffic logs from our university, we measured the following six metrics to evaluate the effectiveness and efficiency of our revocation strategy. Each of these metrics were measured every 24 hours.

[3] Many modern operating systems natively allow users to enable rotating MAC addresses including Android, iOS, Windows, and some Linux distributions.

- **Total TLS handshakes with known status**— The primary metric by which we evaluated the effectiveness of our revocation strategy is the percentage of TLS handshakes in which the requested revocation status was available in either a CRT or CRS. We reported the total across all certificates regardless of status and also across only the revoked certificates.
- **Total certificates with known status**— Another way to evaluate effectiveness is to measure the percentage of unique certificates used in TLS handshakes for which the requested revocation status was available. The inverse of this metric is equivalent to the percentage of certificates used by clients which could be exploited. Similar to "Total TLS handshakes with known status", we reported the total across all certificates regardless of status and also across only the revoked certificates.
- **CRT total certificates**— The total number of certificates in a CRT.
- **CRT idle certificates**— The total certificates in a CRT that will not be used again before eviction. We calculated the number of idle certificates by looking forward through traffic logs, checking which certificates would not be used again in the next τ days. If these certificates could be identified and evicted early, efficiency could increase while effectiveness would remain the same.
- **Daily network bandwidth**— The primary metric to evaluate the efficiency of our revocation strategy is the daily network bandwidth consumption to operate a CRT and to regularly download CRSs. In our experiments, we estimated the CRT's bandwidth cost by assuming OCSP was used to regularly check each stored certificate's revocation status. For our organization-based design, a client's bandwidth requirement is only the cost to regularly download CRS delta updates.
- **Storage required**— The storage required to operate a CRT and to store CRSs on end clients. A CRT stores the certificate information required for revocation

checking. In our experiments, we assumed OCSP was used to check each certificate, which required the leaf and issuer certificates to be stored.

Our experiments collected data points for several values of τ for the entire dataset and for each of the 1,000 clients. We recorded a value every 24 hours for each of the six metrics at each specified value of τ over the course the 60 days. For our organization-based experiments, we measured the effects of regularly using the periodically created CRSs to check each revocation status. Revocation information was only updated every β (24) hours. For our single-client experiments, clients checked each revocation status through the "Check Revocation Status" API in a locally run CRT. If a revocation status was unknown in the CRT, we assumed the revocation information was collected and available for future connections 1 minute after the initial request.

To better represent the long-term effectiveness of a deployed implementation, we removed data points collected during a CRT's initial learning period from our results. The initial learning period of a CRT is the first τ days during which it collects traffic. After this period, the number of certificates in a CRT will stabilize because the CRT may begin to evict certificates.

4.3.4 Limitations

Data from passively collected traffic logs is a reflection of the corresponding population. We collected data from only a single university, and our results may not generalize across all organizations.

Because clients were anonymized in our single-client experiments, we were unable to report demographics of the people using these clients or the type of device that was used (e.g. a mobile phone, server, or general-use PC). Additionally, it is possible that clients we measured in our experiments made additional unmeasured TLS handshakes through another network. While more intrusive, installing logging software on individual clients, especially for mobile devices such as laptops and smartphones, would have given a more thorough view of client certificate-usage patterns.

4.4 Reference Implementation

We created a partial reference implementation of a CRT in order to simulate a production system and generate measurements that we can use for evaluating the system performance.

4.4.1 Revocation Status Refresh

An important goal of our experiments was to collect the revocation status of each certificate in a CRT in order to create and measure the file size of the CRSs. As making OCSP requests for each of the numerous runs of the experiment would be infeasible for us and for OCSP endpoints, we checked the revocation status of each certificate in our dataset one time after the logs were collected (see Appendix A). From this information, we created a revocation database that included each certificate and its revocation status. If the certificate was revoked, the time at which it was revoked was also included. When a revocation status was checked for a certificate, a status was only returned 'revoked' if the query in question was made after the time of revocation.

4.4.2 Certificate Eviction

In our experiments, certificate eviction occurred under two conditions: the certificate had left the working set W(t, τ), or the certificate had expired. A certificate left the working set W(t, τ) if it had not been used by any client from time t - τ to t.

4.4.3 Creating Certificate Revocation Sets and Delta Updates

The implementation created CRSs and delta updates to these CRSs for every 24 hours of logged data. To create these CRSs, we chose to use a simple but efficient list-based approach. This list used certificate serial numbers as identifiers that were stored in 28 to 34 characters in hexadecimal format. This is a significant savings compared to using SHA-256 hashes (64 characters). Because certificate serial numbers are unique only for a given issuer, entries were categorized by the SHA-256 hash of the issuer certificate. A separate list was created for two certificate status options: good and revoked[4]. Each group of CRSs included the header with a SHA-256 signature, a 4-digit version number, and a Unix epoch timestamp. With proper organization, these lists could be searched on disk in O(log(n)) time with two binary searches. If the lists are loaded into an in-memory set implementation, they could be searched in O(1) time.

The LZMA compression algorithm was used on the group of CRSs. While compression does not reduce the file size while in use, it does reduce the amount of network bandwidth required to transmit the file. In our experiments, the compression rate was approximately 50%. While these savings would be similar to using the binary form of the serial numbers, we decided to use the hexadecimal form because it is human readable.

Delta updates were created every 24 hours. A delta update consists of only the certificates that were added or removed since the creation of the last group of CRSs. These delta updates utilize the same file structure as a CRS.

4.5 Results

This section contains the results for the six-evaluation metrics described in our methodology using the passive traffic logs and our reference implementation. In the dataset, there were 524,597 Internet-usable unique certificates[5]. After removing entries for resumed or failed connections, we measured 4,144,404,123 outgoing TLS connections. We identified a total of 112 revoked certificates that were transmitted within 228,427 TLS connections in our dataset. A minimum of 14 revoked certificates were identified each day.

[4] We did not include any certificates with an unknown status in our experiments.

[5] We removed 79,663 certificates from our dataset. Each of these certificate's subject field was unusable for Internet use such as 'LocalHost' or a large string of numbers.

4.5.1 TLS handshakes with known status

For our organization-based experiments, we found over 99% of the TLS handshakes had local access to the needed revocation status even using a working set window size, τ, of only 1 day (see Table 4.1 and Figure 4.1). When we only measured the TLS handshakes using a revoked certificate, we found a similar percentage of TLS handshakes had local access to the needed revocation status information when τ was at least 10 days[6]. This locally cached revocation information will be accessible even in the presence of Internet traffic manipulation at the time of the connection.

We found a lower percentage rate when we examined the single-client experiments. With a τ value of 1 day, on average we found the status of 64.97% of TLS handshakes had local access to the needed revocation status (see Table 4.2 and Figure 4.2). At the largest value τ we report, 45 days, the average accessibility rate across clients reached 91.75%.

4.5.2 Total certificates with known status

On average with a τ value of 1 day, the revocation status of 60.63% of unique certificates used by our organization were available to clients via bi-daily CRSs. This average increased to 90.91% when we extended the working set window length to 45 days. When examining only the revoked certificates, we found even stronger results for each specified value of τ (77.42% for τ equal to 1 day; 95.28% for τ equal to 45 days).

Similar to the average TLS handshakes with a known status, the average percentage of certificates with a known status in our single-client experiments was significantly lower than our organization-based experiments at a τ value of 1 day (42.26% versus 60.63%). However, when using a large value of τ such as 45 days, the difference was reduced (84.45% versus 90.91%).

[6] Unlike our experiments, in practice we advise that extra care should be taken before evicting revoked certificates from the CRT before they have expired.

τ: working set window length	TLS handshakes with known status		Certificates with known status		CRT total certificates	CRT idle certificates	Daily network bandwidth		Storage required	
	Of Handshakes with Any Certificate	Of Handshakes with Revoked Certificates	Any Certificate	Revoked Certificates			CRT	End client	CRT	End client
1 day	99.52%	96.55%	60.63%	77.42%	56,957.83	40.73%	72.31 MB	747.31 KB	220.27 MB	1.71 MB
5 days	99.71%	98.82%	80.01%	92.45%	127,702.09	42.87%	162.12 MB	401.45 KB	493.85 MB	3.83 MB
10 days	99.73%	99.59%	85.28%	94.84%	180,355.30	45.82%	228.97 MB	302.39 KB	697.47 MB	5.41 MB
15 days	99.73%	99.59%	87.34%	95.22%	223,133.91	48.95%	283.28 MB	265.04 KB	862.90 MB	6.70 MB
20 days	99.73%	99.55%	88.38%	95.20%	261,310.38	51.72%	331.74 MB	245.00 KB	1,010.54 MB	7.86 MB
25 days	99.76%	99.49%	89.34%	94.86%	297,767.51	54.15%	378.03 MB	229.07 KB	1,151.52 MB	8.96 MB
30 days	99.83%	99.65%	90.05%	95.90%	332,136.97	N/A	421.66 MB	216.17 KB	1,284.44 MB	10.00 MB
35 days	99.84%	99.67%	90.48%	96.16%	363,148.84	N/A	461.03 MB	209.08 KB	1,404.36 MB	10.94 MB
40 days	99.82%	99.67%	90.35%	95.96%	392,611.35	N/A	498.43 MB	208.71 KB	1,518.30 MB	11.83 MB
45 days	99.86%	99.61%	90.91%	95.28%	423,032.13	N/A	537.05 MB	205.09 KB	1,635.94 MB	12.75 MB

Table 4.1: The average values in our organization-based experiments for each of the six-evaluation metrics for several values of τ. Values from a CRT's initial learning period are excluded from the aggregation.

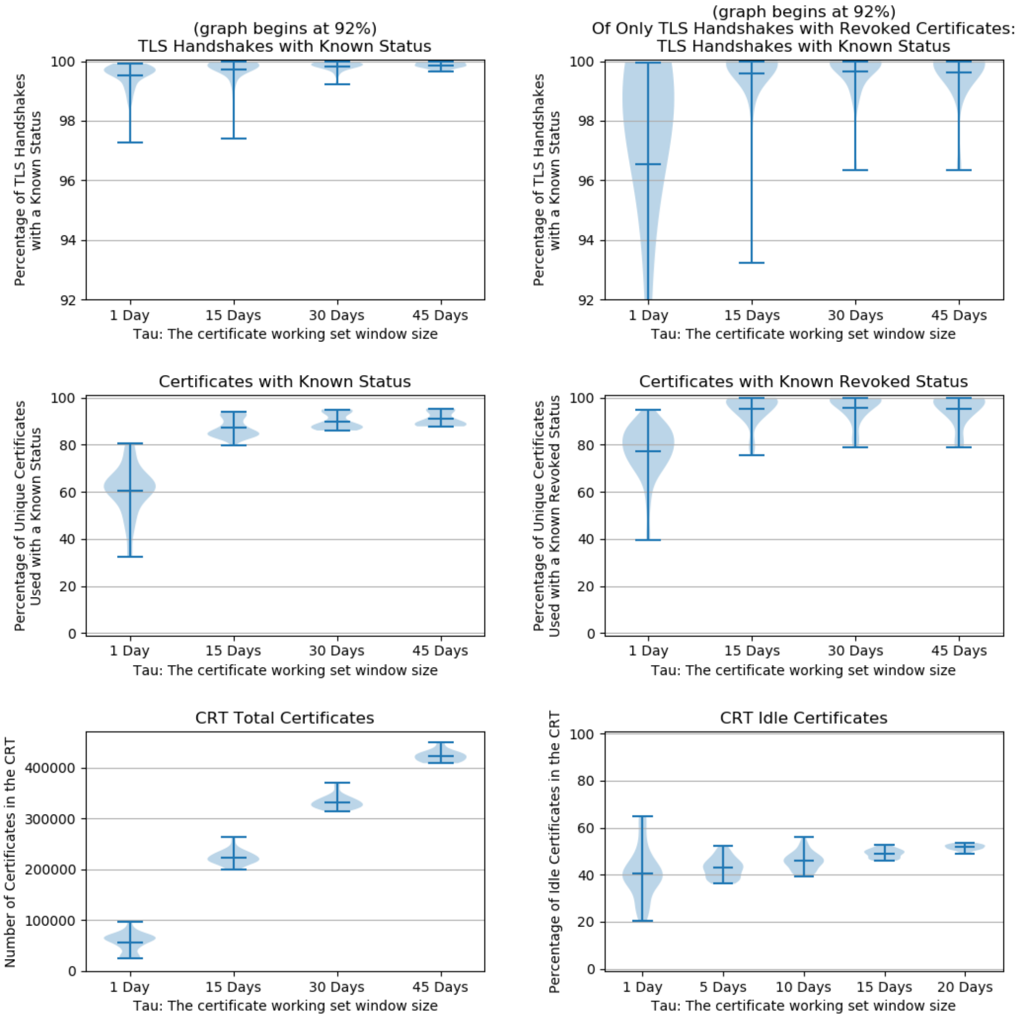

Figure 4.1: The distribution of our bi-daily results in our organization-based experiments for several values of τ for the following metrics: TLS handshakes with known status, TLS handshakes with known revoked status, Certificates with known status, Certificates with known revoked status, CRT total certificates, and CRT idle certificates. Values from a CRT's initial learning period are excluded.

τ: working set window length	TLS handshakes with known status	Certificates with known status	CRT total certificates	CRT idle certificates	Daily network bandwidth	Storage required
1 day	64.97%	42.26%	83.83	70.89%	0.11 MB	0.32 MB
5 days	79.92%	63.56%	247.86	65.29%	0.31 MB	0.96 MB
10 days	84.77%	72.06%	399.86	64.46%	0.51 MB	1.55 MB
15 days	87.03%	76.04%	528.73	63.99%	0.67 MB	2.04 MB
20 days	88.30%	78.25%	644.56	63.83%	0.82 MB	2.49 MB
25 days	89.33%	80.05%	753.88	64.23%	0.96 MB	2.92 MB
30 days	89.83%	81.16%	856.30	N/A	1.09 MB	3.31 MB
35 days	90.29%	81.99%	951.01	N/A	1.21 MB	3.68 MB
40 days	90.74%	82.74%	1,042.61	N/A	1.32 MB	4.03 MB
45 days	91.75%	84.45%	1,138.96	N/A	1.45 MB	4.40 MB

Table 4.2: The average values across our client-based experiments for each of the six-evaluation metrics for several values of τ. Values from a CRT's initial learning period are excluded from the aggregation.

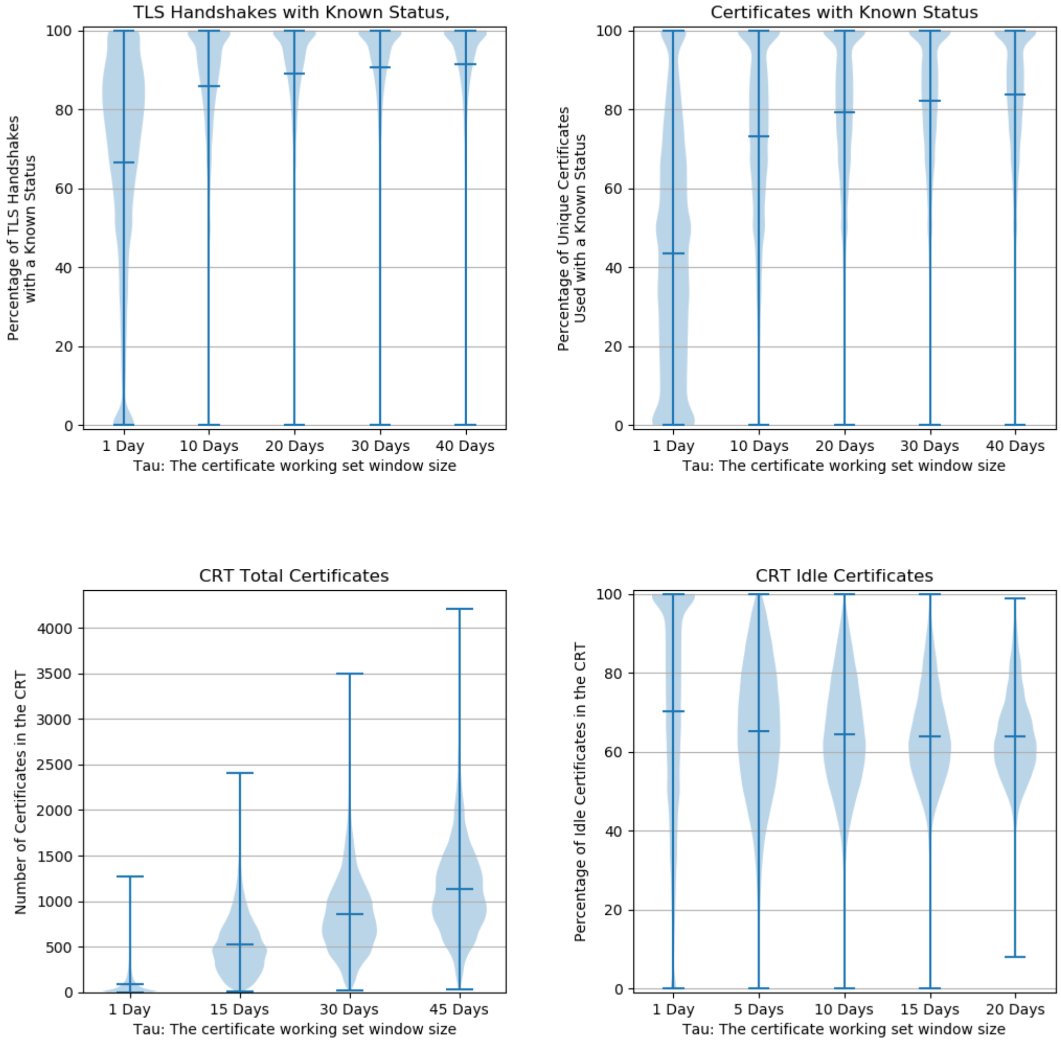

Figure 4.2: The distribution of our bi-daily results across our client-based experiments for several values of τ for the following metrics: TLS handshakes with known status, Certificates with known status, CRT total certificates, and CRT idle certificates. Values from a CRT's initial learning period are excluded.

4.5.3 CRT total and idle certificates

Only a small number of certificates were used in single-client CRTs compared to a CRT shared by an organization. Even so, the percentage of idle certificates was high for both the organization (40.73% to 54.15%) and single-client experiments (64.23% to 70.89%). However, while the percentage of idle certificates increased with larger τ values in our organization-based experiments, the average percentage of idle certificates in CRTs used by single clients followed a concave pattern with a minimum at τ equal to 20 days.

Notably, the percentage of idle certificates for τ of 30 days and above are not measured or reported. This is because determining whether a certificate will not be used again requires examining τ days of future traffic, which were unavailable in our dataset for larger values of τ.

4.5.4 Daily network bandwidth

To estimate the daily bandwidth consumption of a CRT in our experiments, we assumed OCSP was used to check the revocation status of each stored certificate every β hours[7] and that each OCSP request used 1.3 KB [25] of bandwidth. Our experiments found that this cost ranged between 72 MB to 537 MB for τ values of 1 day and 45 days respectively (see Table 4.1). This cost is significantly less compared to the bandwidth required for every individual client in an organization to perform their own OCSP requests. Single-client systems averaged 112 KB of daily bandwidth consumption when using a τ value of 1 day and averaged 1.45 MB daily bandwidth consumption when using a τ value of 45 days.

For our organization-based implementation, end clients would utilize CRSs instead of regularly refreshing the revocation status of each certificate in a CRT through OCSP requests. Initially, a client would download a compressed group of CRSs, which was only 6.7 MB on average even at a τ value of 45 days (0.89 MB at a τ value of 1 day) (see Table 4.3 and Figure 4.3). Then the client would download a delta update once a day averaging

[7] β was set to 24 hours for both our organization-based experiments and our single-client experiments.

τ: working set window length	Average file size of good and revoked CRSs				Average file size of revoked CRSs only			
	Entire data structure		Data structure's delta		Entire data structure		Data structure's delta	
	Uncompressed	XZ Compressed	Uncompressed	XZ Compressed	Uncompressed	XZ Compressed	Uncompressed	XZ Compressed
1 day	1.71 MB	0.89 MB	1,441.41 KB	747.31 KB	1.27 KB	0.87 KB	0.55 KB	0.46 KB
5 days	3.83 MB	2.00 MB	773.71 KB	401.45 KB	1.80 KB	1.16 KB	0.29 KB	0.30 KB
10 days	5.41 MB	2.83 MB	583.09 KB	302.39 KB	2.10 KB	1.32 KB	0.24 KB	0.26 KB
15 days	6.70 MB	3.51 MB	511.17 KB	265.04 KB	2.34 KB	1.45 KB	0.22 KB	0.25 KB
20 days	7.86 MB	4.11 MB	472.75 KB	245.00 KB	2.56 KB	1.56 KB	0.22 KB	0.24 KB
25 days	8.96 MB	4.70 MB	441.94 KB	229.07 KB	2.78 KB	1.68 KB	0.20 KB	0.23 KB
30 days	10.00 MB	5.24 MB	417.15 KB	216.17 KB	2.96 KB	1.78 KB	0.17 KB	0.21 KB
35 days	10.94 MB	5.75 MB	403.45 KB	209.08 KB	3.05 KB	1.83 KB	0.16 KB	0.20 KB
40 days	11.83 MB	6.21 MB	402.62 KB	208.71 KB	3.13 KB	1.87 KB	0.16 KB	0.20 KB
45 days	12.75 MB	6.71 MB	395.64 KB	205.09 KB	3.23 KB	1.92 KB	0.18 KB	0.21 KB

Table 4.3: The average file size of CRSs and their deltas created in our organization-based experiments for several values of τ. CRSs created during a CRT's initial learning period are excluded from the aggregation.

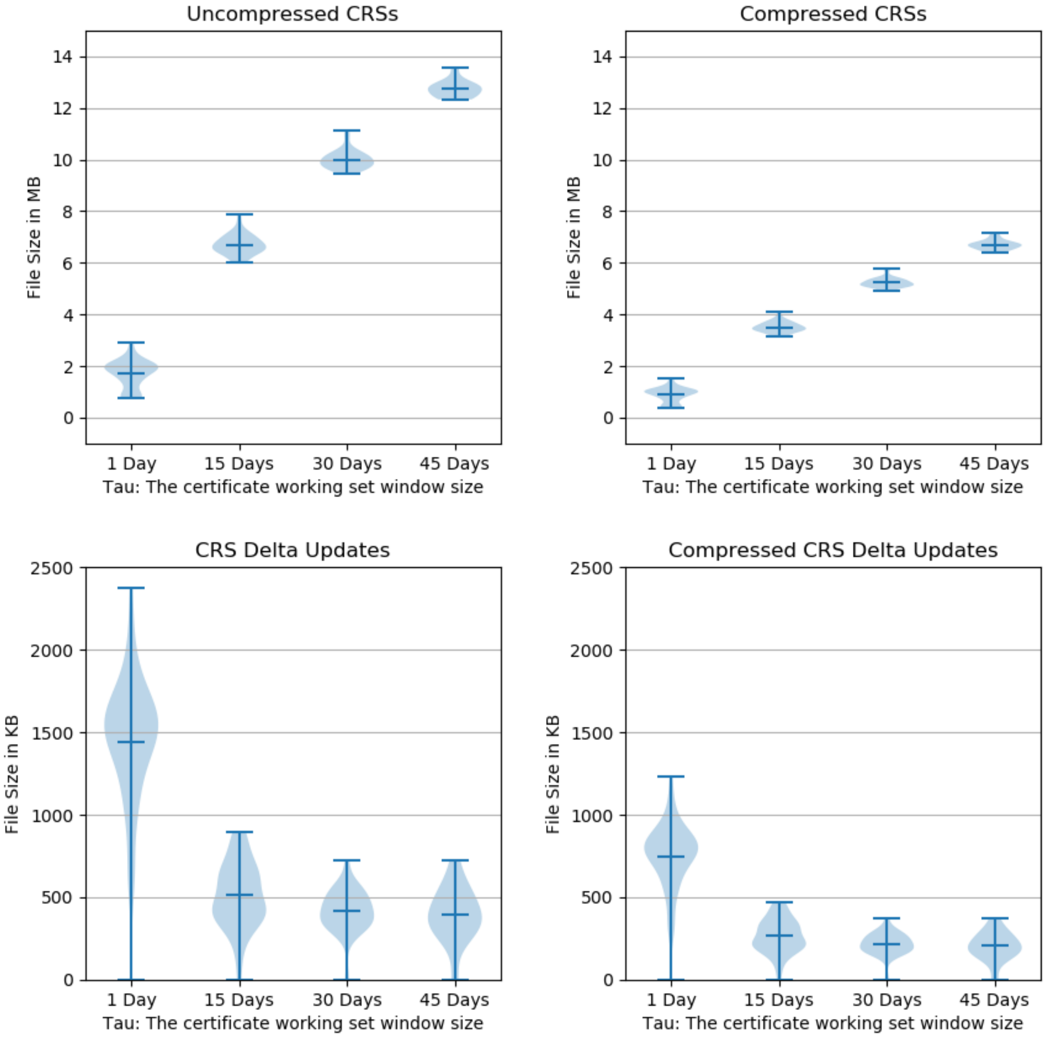

Figure 4.3: The distribution of file sizes for the CRSs and deltas created in our organization-based experiments for several values of τ. CRSs created during a CRT's initial learning period are excluded.

only 205 KB each at a τ value of 45 days. Notably, at a τ value of 1 day, each delta update is similar in size to the full CRS (0.73 MB versus 0.89 MB). On average, delta updates to the CRS with revoked certificates were only 215 to 471 bytes depending on the value of τ used.

While the CRS file sizes grew as the value of τ increased, the file size of delta updates to these data structures were reduced as the value of τ increased. We hypothesize that certificates may be evicted too early when τ is too small. As delta changes include additions and subtractions from the previous data structure, early eviction would explain why the delta updates are larger at small τ values.

4.5.5 OCSP request rate

For an organization-shared CRT using a τ value of 45 days with a β value of 24 hours, an average of 421,032 OCSP requests were made per day. Making 421,032 OCSP requests per day requires 4.87 OCSP requests per second and approximately 6.3 KB per second of bandwidth. At a τ value of 1 day (56,958 OCSP requests), a CRT only would require 0.66 OCSP requests per second.

A single-client CRT a τ value of 45 days with a β value of 24 hours makes 1,139 OCSP requests on average per day. These clients only need to make 95 OCSP requests per hour. At a τ value of 1 day (84 OCSP requests), clients only need to make 3.5 OCSP requests per hour.

4.5.6 Storage required

CRTs require both the certificate and the issuer certificate to perform an OCSP request. The average certificate size in our experiments is only 1.98 KB in PEM format. Using this average value for a certificate, we estimated the upper bound for the storage requirement of an organization-shared CRT is only 0.2 GB for a τ value of 1 day and 1.6 GB for a τ value of 45 days. This upper bound assumes every certificate in the CRT has a unique issuer. We expect the true cost to be just over half of this upper bound because most certificates are issued by a small number of CA's [4]. We estimated that the average upper bound for single-client implementation with a τ value of 1 day only needs 0.3 MB of storage and with a τ value of 45 days only needs 4.40 MB of storage for these certificates.

End clients of organization-based implementations only need to store an uncompressed group of CRSs on their device (see Table 4.3 and Figure 4.3). Our experiments found this file size to be between 1.71 MB and 12.75 MB depending on the value of τ chosen.

Chapter 5

Discussion

In this section, we address some of the implications of our results and discuss how our solution can be used in various scenarios.

5.1 Managing Privacy Concerns

Ideally, clients who regularly check the revocation status of the certificate of each website they visit should not have to reveal their traffic patterns to untrusted third parties. We discuss options that can be followed for preserving client privacy depending on the deployment scenario.

5.1.1 Organization-Based Middlebox

A private network administrator could use a middlebox to capture the certificate usage of all clients that connect to the Internet through their network gateway[1]. In private networks, clients already rely on network administrators to properly handle network traffic logs containing sensitive data. Network administrators insert the certificates captured from the middlebox into a CRT to create the organization's certificate working set. This action does not impact client privacy because this traffic information is already available to and often analyzed by network administrators.

[1] The middlebox only needs to be used to collect traffic information, however it could be used to additionally check and block live TLS handshakes with revoked certificates.

5.1.2 Private Cloud-Service

If clients in an organization do not share a physical or virtual network, not all traffic may go through a middlebox. A CRT could be deployed as a cloud service. We assume the all clients in the organization trust the administrator deploying the CRT to properly process their traffic information, removing the need to anonymize clients from the CRT. In this system, the CRT working set is generated from the certificates obtained through the "Check Revocation Status" API[2]. To maintain privacy, communication between clients and CRTs should be encrypted to prevent eavesdropping.

5.1.3 Honest-but-Curious Third Parties

CRTs could also be deployed by an honest-but-curious collection of independent third parties, as the case may be if it were to be used by a group of security enthusiasts. We assume that each CRT is benign but traffic information entering the system may be read by one or many curious third parties. For example, CRTs could be made transparent to all clients as an auditing mechanism. In addition to encrypted communication, clients must be able to anonymously share their certificate usage with a CRT.

One deployment option to anonymize traffic is to use a similar solution to the "notary bounce" as defined by the Convergence Notary [28]. This would require multiple third parties to each independently deploy a CRT. First, A client chooses two CRTs, one to act as a privacy proxy and the other to act as the final destination. Second, the client encrypts its certificate usage and a symmetric key with the final destination's public key. Third, the client sends the encrypted payload to the privacy proxy along with the address of the final destination. Fourth, the privacy proxy forwards the payload to the final destination. Fifth, the final destination reads the client's request and then uses the

[2] This will increase a client's daily bandwidth cost compared to implicit traffic collection. However, if clients send this information as a list of SHA-256 certificate hashes and the number of unique certificates used in a day per client is similar to those in our study (113.56), the additional bandwidth can be reduced to an average of 3.55 KB a day.

client's symmetric key to encrypt the response. Finally, the final destination returns the encrypted response to the privacy proxy who forwards it to the client. The privacy proxy is unable to read the data while the final destination does not know the origin. This solution maintains client anonymity as long as the servers do not collude.

By allowing clients to insert anonymized data into the system, the system is vulnerable to a Sybil attack. Malicious clients can send invalid entries with many different pseudonyms in order to hinder the model's efficiency. The depletion of CRT resources by malicious clients may require early eviction of active certificates. Solutions to mitigate invalid entries include increasing the cost to participate, limiting participants, or connecting clients to a reputation system. Further discussion on how to defend against Sybil attacks is out of scope of our research.

5.1.4 Single-Client CRT

While operating a CRT directly on an end client avoids the privacy concerns of organization-based implementations, it inherits the privacy concerns of OCSP. This is because the only currently deployed general-purpose revocation strategy that can check the revocation status for almost any certificate is OCSP. The only other deployed general-purpose revocation strategy is the use of CRLs, which in June of 2018 could only be used to validate 30% of the publicly used CA-trusted certificates (see Table A.3)[3]. Because the aggregation of many clients is not used to anonymize revocation status requests, privacy-exposing strategies such as OCSP reveal traffic patterns to third parties.

Because currently OCSP is required to check the revocation status of many certificates used by a CRT, we suggest two ways OCSP can be supported to preserve the anonymity of a client's certificate usage from all third parties. First, if OCSP traffic can be encrypted[4], the encrypted traffic can be forward through a proxy to anonymize the

[3] The CA Let's Encrypt issues a majority of OCSP-only certificates globally.

[4] According to the RFC defining OCSP [29], OCSP requests may be encrypted in some form, although this does not seem to be the case in practice.

client. Second, a client using a service equivalent to the notary bounce [28] or a multi-hop privacy proxy (see Section 5.1.3) would preserve the anonymity of the client.

5.2 Enabling System and Network Administrators

Revocation strategies that can be deployed without third-party support and at scale enable system and network administrators to defend themselves. Solutions [21, 30, 33] such as ours correctly align deployment incentives as the tools are used by those who are directly concerned with and invested in the targeted clients' security. We believe the dependency on third-party adoption has debilitated the deployment of many otherwise promising revocation strategies. We encourage further research into scalable tools that empower security conscious end users and organizations.

5.3 Flexible Solution

System administrators are able to choose the parameters (τ. β, α) used in the CRT to balance effectiveness, efficiency, and revocation timeliness depending on the needs of the organization or individual. For example, if there is ample bandwidth and storage, setting τ to infinity may increase the effectiveness of the system because certificates are not evicted from the CRT until they expire. Decreasing the values of β and α would improve the revocation timeliness. We expect many organizations could improve the revocation timeliness without a significant resource burden because a CRT regularly rechecks only a small subset of the globally trusted certificates. Based on our experimental results, cutting alpha and beta in half (12 hours) would only increase the daily bandwidth cost to 9.74 OCSP requests per second and total consume 1.05 GB of network traffic.

5.4 Popular Websites Avoid Failing Revocation Checks

The revocation status for certificates of popular websites will be in a CRT and therefore accessible to end clients. We define a 'popular website' as a website (and its certificate) that clients in the organization visit at least once every τ days[5]. Arguably, a majority of websites important to clients will be visited enough to be defined as 'popular' and the revocation status of each of these certificates will be regularly rechecked.

5.5 Overlapping Certificate Working Sets Between Clients in an Organization

There is a substantial difference in the percentage of TLS handshakes with a known status in our organization-based experiments versus our single-client experiments. We believe the primary reason for this difference is the certificate working sets of different clients in an organization overlap. Because clients visit some of the same websites as other clients in an organization, it is highly likely that a first-time visit to a website for each individual client will have the requested revocation status already cached. However, the revocation status of each certificate seen by a client in a single-client system will be unknown to the client at least once. It follows that TLS handshakes with a known status will be smaller for a single-client systems compared to an organization-shared system.

5.6 Application to Mobile Clients

Bandwidth consumption has previously dissuaded browsers on mobile clients to check any certificate revocation status [26]. This is because mobile clients often use a metered Internet connection with limited bandwidth. An implementation of a CRT can be made mobile-friendly by dynamically changing the behavior of the system based on the network

[5] As some websites have multiple certificates and because certificates expire, the number of required visits is slightly higher.

conditions. For example, the system should attempt to update CRSs or perform OCSP requests while connected to a Wi-Fi network.

If extended time has passed without connecting to a Wi-Fi network, clients should only collect the most critical revocation information. If a client is using an organization-shared CRT, they can download the CRS containing only revoked certificates. In our dataset, regularly downloading this CRS and its updates for an entire month will use fewer than 0.0003% (6.30 KB) of a 2 GB monthly bandwidth allotment. Clients operating their own CRT should only check the revocation status of certificates used within the last few days. In our dataset, a client who checks the revocation status of certificates used within five days through OCSP every day for an entire month will use 0.45% (9.3 MB) of a 2 GB monthly bandwidth allotment[6].

[6] This assumes that mobile clients follow the same average certificate usage as clients in our experiments.

Chapter 6

Comparison to Alternative Solutions

This chapter compares the two CRT deployments alternatives with three state-of-the-art revocation strategies: OCSP Must-Staple, CRLSets, and CRLite. We identify whether CRT is less than, equivalent, or exceeds these existing systems for each of the seven concerns facing revocation strategies as discussed in the Chapter 1. Table 6.1 contains a summary of the results.

6.1 Organization-Based Implementation

The following results are based on a τ value of 45 days[1], a β value of 24 hours, and an α value of 24 hours. We chose a τ value of 45 days because it protected the highest percentage of TLS handshakes in our experiments.

- **Effectiveness: Less Than**— If clients use CRLite or all web servers adopt OCSP Must-Staple, clients would avoid failing a revocation check for 100% of their TLS handshakes. Because CRLSets are used to protect a small selection of critical certificates, we are unable to evaluate this property for CRLSets. Clients using a CRS will on average avoid failing a revocation check over 99.86% of the time. A CRT with these chosen design parameters will fail the revocation check for a small percentage of TLS handshakes.

  We believe our CRT solution could approach 100% by modifying the chosen design parameters. For example, we believe reducing the time between delta updates, α, will

[1] We note that percentage of TLS handshakes with a known revocation status only slightly improves from τ values of 30 to 45 days and it would be reasonable to use a smaller τ value in practice.

	TLS Handshakes Protected	Client Bandwidth Consumption	Global Certificate Growth Scalability	Mass Revocation Event Scalability	Revocation Timeliness	Privacy Preserving	Deployment Requirements
OCSP Must-Staple	100%∗	1.67 MB per day†	Minimal Bandwidth Growth	No Changes	4 Days	Yes	Very High
CRLSets	Unknown∗∗	250 KB per day	Reduced Protection	Minimal Protection	1–2 Days	Yes	Deployed
CRLite (Jan. 2017; 30 Million Certificates)	100%	Initially 10 MB; 580 KB per day	Significant Bandwidth Growth	Significant Bandwidth Growth	1–2 Days	Yes	High
CRLite (Mar. 2018; 84 Million Certificates)	100%	Initially 18 MB; Unknown per day	Significant Bandwidth Growth	Significant Bandwidth Growth	1–2 Days	Yes	High
Organization-Based CRSs ($\tau = 45$ days)	99.86%	Initially 6.71 MB; 205 KB per day	Minimal Bandwidth Growth	Minimal Bandwidth Growth	1–2 Days	Yes	Medium
Revoked Only CRS	99.86%	Initially 1.92 KB; 0.21 KB per day	Minimal Bandwidth Growth	Significant Bandwidth Growth	1–2 Days	Yes	Medium
Single-Client CRT ($\tau = 45$ days)	91.75%	1.45 MB per day	Minimal Bandwidth Growth	No Changes	1 Day	No	Low

Table 6.1: Comparison of our CRT implementations to other revocation strategies.
†: On average, each client in our experiments made 1315.86 TLS handshakes per day. Ideally, every TLS handshake would include the stapled response. Each response is roughly 1.3 KB [25].
∗: While including the revocation status for 100% of TLS handshakes is possible, current server adoption rates (0.03% of certificates) result in OCSP Must-Staple protecting very few TLS handshakes.
∗∗: We have no way of measuring the effectiveness of CRLSets for this property. Because CRLSets contain only a small number of revoked certificates (roughly 40,000), the number of certificates prechecked for the client will be low.

increase effectiveness while only slightly increasing client bandwidth consumption. The further exploration of these design parameters are left as future work.

- **Efficiency: Exceeds**— The cost to download an OCSP staple for every TLS handshake requires roughly eight times more bandwidth than using CRSs[2]. When downloading delta updates for each CRS including non-revoked and revoked certificates, the daily client bandwidth and storage requirements for this deployment strategy are less than but comparable to CRLite and CRLSets. When downloading the delta updates for each CRS including only revoked certificates, the daily client bandwidth and storage requirements for this deployment are three orders of magnitude lower than any other strategy.

- **Certificate Growth Scalability: Equivalent**—

  The bandwidth consumption of our revocation strategy increases with the number of unique certificates used by an organization (i.e. the organization's working set). This is a significant improvement compared to CRLite whose bandwidth is a factor of the number of live CA-trusted certificates and the percentage of those that are

[2] This assumes that clients make the same number of TLS handshakes per day on average as clients in our experiments.

revoked globally. The bandwidth consumption of OCSP Must-Staple increases with the number TLS connections made, which is indirectly affected by certificate growth. Because CRLSets use a fixed size, they do not face bandwidth scalability concerns. However, growth in certificate issuance reduces the percentage of the globally revoked certificates CRLSets can protect at one time.

- **Mass Revocation Event Scalability: Equivalent**—

  During a mass revocation event, the bandwidth consumption of a CRT will not change because a CRT already collects each certificate's revocation status regardless of whether it has been revoked or not. Similarly, the revocation status of the certificates used does not affect OCSP Must-Staple. The delta updates will be more expensive while a large number of certificates are being revoked. The CRS with revoked certificates will grow in size but they will still only be a fraction of the size of the non-revoked version.

  Because CRLite's bandwidth is a partially a factor of the percentage of certificate that are revoked globally, the bandwidth consumption of clients using CRLite will increase during mass revocation events. CRLSets only protect a small number of certificates, so in a mass revocation event, many revoked certificates will excluded from a CRLSet.

- **Revocation Timeliness: Equivalent**— Clients using an organization-shared CRT (with β set to 24 hours), CRLite, and CRLSets will download revocation information once every day. In addition, supporting servers require some amount of time to collect the revocation information distributed in the daily downloads. In the worst-case for an organization-shared CRT (with β set to 24 hours), CRLite, and CRLSets, a revoked certificate could be trusted by a client almost 48 hours after the initial revocation (see Section 4.2).

The worst-case revocation timeliness for OCSP Must-Staple is equal to the OCSP response's lifetime. The median OCSP response expires after four days [25]. This worst-case scenario would require an attacker to obtain an OCSP response created right before the certificate was revoked. The active attacker advertising the revoked certificate would staple the OCSP response to the handshake. Clients would trust this certificate until the stapled OCSP response expires.

- **Privacy: Equivalent**— Ideally, solutions should not expose clients' traffic history. OCSP Must-Staple, CRLSets, and CRLite do not collect nor expose individual client traffic patterns. In an organization-shared CRT, there is the possibility of revealing individual client traffic patterns at collection points. As discussed in Section 5.1, there are multiple deployment options for managing client traffic privately.

  Our organization-based implementation does reveal the aggregated collection of certificates to all end clients and parts of this aggregation to individual OCSP responders. The organization's collection of certificates, which is not normally available to end clients, could be used to alert members of the organization if a controversial website (addiction, politics, religion, suicide help, etc.) was visited by some client on the network. While this implementation does not expose information besides that a particular certificate was used, this leaked data could be a critical problem for the reputation or privacy of some organizations. However, this concern does not identify a client nor expose a client's traffic history to other clients as long as there are a significant number of clients in the organization.

- **Deployment Requirements: Equivalent**– Our organization-based implementation requires deployment of a dedicated server and for end-clients to adopt the CRT solution. These requirements are also required by both CRLite and CRLSets.

  However, CRLite additionally requires a significant daily burden for supporting servers. As of September 2018, CRLite would need to collect the revocation status for over 317 Million certificates [4], at least 213 Million of which require OCSP requests[3]. Currently, this would require at least 2,465 OCSP requests per second to be made to Let's Encrypt's OCSP Endpoints. While this is not a fundamental problem, in practice this would require coordination between the Let's Encrypt and the CRLite system in order to avoid overwhelming Let's Encrypt. This effort could be reasonably done if Let's Encrypt either provides read-only access to their revocation information to the CRLite system or maintains a significant infrastructure to reliability handle this bombardment of OCSP requests. Regardless, this requirement increases the difficulty to deploy a CRLite server.

Even with the costs of CRlite, OCSP Must-Staple is still significantly harder to deploy as it requires each website administrator to adopt the strategy and CAs to manage problems of availability and revocation status freshness. If mistakes are made, websites become unavailable to clients which disincentivizes website administrators to adopt OCSP Must-Staple.

Arguably though, there are more incentives and fewer resources required to deploy a CRT compared to the competing solutions. First, network administrators who would deploy a CRT already have an invested interest in protecting the clients in their organization. Secondly, because a CRT only targets a relatively small number of certificates compared to all certificates globally, a CRT only requires a fraction of the cost (3 orders of magnitude) to gather each revocation status compared to CRLite who is required to gather the revocation status of all known certificates. The cost to distribute CRSs to end clients across the Internet compared to both CRLite and CRLSets is also greatly reduced because there are fewer targeted clients. There is also a possibility that CRSs can be distributed through a private network reducing daily Internet bandwidth costs for the CRT. While there are reasons that suggest some deployment methods of a CRT are easier to deploy than the other

[3] These are issued by Let's Encrypt.

revocation strategies, we still label this property as ‘equivalent’ to CRLSets because of the requirement to deploy a server and install client software.

6.2 Single-Client Implementation

The following results are based on a τ value of 45 days, a β value of 24 hours, and an α value of 24 hours. We chose a τ value of 45 days because it protected the highest percentage of TLS handshakes in our experiments.

- **Effectiveness: Less Than** — Clients using our single-client implementation had the revocation status locally cached for an average of 91.75% of the TLS handshakes they make. This percentage is less than CRLite or OCSP Must-Staple.
- **Efficiency: Less Than**— This deployment option requires individual clients to perform relatively expensive OCSP checks for each certificate in their CRT themselves. This method uses significantly more client bandwidth than both CRLite and CRLSets.
- **Certificate Growth Scalability: Equivalent**— Similar to an organization-shared CRT, bandwidth consumption of a single-client CRT only grows based on the number of unique certificates included. Certificate adoption only indirectly affects this solution. This is similar to OCSP Must-Staple.
- **Mass Revocation Event Scalability: Equivalent**— Similar to OCSP Must-Staple, mass revocation events do not affect the bandwidth consumption as these solutions work independent of the revocation status. CRLite and CRLSets are both negatively effected by mass revocation events (see Section 6.1).
- **Revocation Timeliness: Exceeds** — CRTs refresh each revocation status directly from the CA once every 24 hours. This revocation timeliness is better than the best-case scenario for both CRLSets and CRLite.

- **Privacy: Less Than**— Without any form of aggregation or supporting services, clients will make many unencrypted OCSP requests. These requests reveal browsing patterns to OCSP responders, curious nodes along the path to OCSP responders, and malicious nodes. If supporting systems are available, such as encrypted OCSP or a notary bounce type of system, then the privacy concerns would be reduced, similar to CRLite, CRLSets, and OCSP Must-Staple.

- **Deployment Requirements: Exceeds** — A single-client implementation can easily be deployed today, and it only requires client software. Any client can begin using a CRT without the need for any other party to also adopt the solution. There are strong incentives to participate as clients directly protect themselves. This system is significantly simpler to deploy in comparison to any of the alternative systems we considered.

Chapter 7

Future Work

There are several ways to improve the effectiveness and cost of a CRT.

7.1 Anticipating Certificate Renewal

We suspect that some of the TLS handshakes with unknown status are using newly issued certificates for websites a client or organization already visit semi-regularly. Let's Encrypt issues certificates with a 90-day validity period On an average day, we estimate that approximately 1.11% of the certificates issued by Let's Encrypt will expire and be replaced with a new certificate.

The effectiveness of a CRT could improve if it could detect that a host has obtained a new certificate before any client receives the new certificate. A CRT could periodically check the host certificate for all the cached certificates that are nearing expiration.

7.2 Removal of Irrelevant Certificates

Our results report a large percentage of idle certificates independent of the certificate working set window size (τ). Future work could find ways to identify and evict these certificates early to increase efficiency, especially for single-client systems where the average bandwidth costs are significantly higher than other revocation solutions. We suspect many idle certificates could be identified early by searching for high-order correlations in client traffic patterns with machine learning models like a Multilayer Perceptron.

7.3 Reexamining Single-Client Traffic Patterns

The preliminary analysis of the single-client system suggests that it has significant weaknesses in effectiveness and efficiency compared to using an organization-based system. However, our single-client experimentation had several limitations. We could not identify the types of devices used nor confirm if the logs contained all of the connections made by the client (see Section 4.3.4). A future study could explore the single-client deployment in more depth by, gathering user demographic information, identifying device types, and installing logging software on client devices to capture all client traffic. This information could provide a more accurate and generalizable analysis of this simpler deployment option. If this single-client option could be made viable, then individual clients could deploy it and receive the benefits without having to rely on their organization for support.

Chapter 8

Conclusion

In this thesis, we present Certificate Revocation Table (CRT), a new revocation strategy that leverages locality of reference between the web requests for an organization. The CRT caches recently used certificates along with their revocation status, and periodically refreshes the revocation status so that it is available to end clients when needed. To measure the effectiveness and efficiency of this solution, we used 60 days of TLS traffic logs from a university to measure the effects of actively refreshing certificates for various certificate working set window lengths.

We found that using a CRT to regularly check the status of certificates in the working set of a large private network, such as a university campus network, is an effective strategy to defend against man-in-the-middle threats involving a revoked certificate. Using an organization-shared CRT is competitive with or exceeds alternative state-of-the-art solutions in effectiveness, efficiency, certificate growth scalability, mass revocation event scalability, revocation timeliness, privacy, and deployment requirements.

Additionally, we evaluated the use of a CRT deployed for a single client. Our initial findings suggest that the single-client deployment option performs worse than alternative state-of-the-art solutions. We provided avenues for future work that could improve its effectiveness and efficiency. An open question is whether the single-client deployment option can compete with alternative state-of-the-art solutions.

References

[1] CRLSets. URL `https://dev.chromium.org/Home/chromium-security/crlsets`.

[2] Feature request: OCSP Must Staple (RFC 7633). URL `https://groups.google.com/a/chromium.org/forum/#!topic/security-dev/-pB8IFNu5tw`.

[3] CA:RevocationPlan. URL `https://wiki.mozilla.org/CA:RevocationPlan#OneCRL`.

[4] Censys, 2017. URL `https://censys.io/certificates?q=tags.raw%3A+%22trusted%22`.

[5] Censys, 2017. URL `https://censys.io/certificates?q=%281.3.6.1.5.5.7.1.24%29+AND+tags.raw%3A+%22trusted%22`.

[6] Let's Encrypt Stats, 2017. URL `https://letsencrypt.org/stats/`.

[7] Devdatta Akhawe, Bernhard Amann, Matthias Vallentin, and Robin Sommer. Here's My Cert, so Trust Me, Maybe?: Understanding TLS Errors on the Web. In *Proceedings of the 22nd international conference on World Wide Web (WWW)*, pages 59–70. ACM, 2013.

[8] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. Mission Accomplished?: HTTPS Security After Diginotar. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 325–340. ACM, 2017.

[9] Richard Barnes, Jacob Hoffman-Andrews, Daniel McCarney, and James Kasten. Automatic Certificate Management Environment (ACME) draft-ietf-acme-acme-12. Internet-Draft, April 2018. URL `https://tools.ietf.org/html/draft-ietf-acme-acme-12`.

[10] Adam Bates, Joe Pletcher, Tyler Nichols, Braden Hollembaek, and Kevin R.B. Butler. Forced Perspectives: Evaluating an SSL Trust Enhancement at Scale. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 503–510. ACM, 2014.

[11] Hanno Bock. The Problem with OCSP Stapling and Must Staple and why Certificate Revocation is still broken, 2017. URL `https://blog.hboeck.de/archives/886-The-Problem-with-OCSP-Stapling-and-Must-Staple-and-why-Certificate-Revocation-is-still-broken.html`.

[12] David Cooper, Stephen Farrel, Sharon Boeyen, Russell Housley, and Tim Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, RFC Editor, May 2008.

[13] Peter J Denning. The Working Set Model for Program Behavior. *Communications of the ACM*, 11(5):323–333, 1968.

[14] Zakir Durumeric, James Kasten, David Adrian, J. Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, et al. The Matter of Heartbleed. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 475–488. ACM, 2014.

[15] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. A Search Engine Backed by Internet-Wide Scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 542–553, 2015.

[16] Donald Eastlake. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066, RFC Editor, January 2011.

[17] Phillip Hallam-Baker. X.509v3 Transport Layer Security (TLS) Feature Extension. RFC 7633, RFC Editor, October 2015.

[18] Scott Helme. Designing a new Security Header: Expect-Staple, 2017. URL `https://scotthelme.co.uk/designing-a-new-security-header-expect-staple/`.

[19] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. The SSL Landscape: A Thorough Analysis of the X. 509 PKI Using Active and Passive Measurements. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 427–444. ACM, 2011.

[20] Yung-Kao Hsu and Stephen Seymour. Intranet Security Framework Based on Short-lived Certificates. In *Proceedings of the Sixth IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 228–234, 1997.

[21] Qinwen Hu, Muhammad Rizwan Asghar, and Nevil Brownlee. Certificate Revocation Guard (CRG): An Efficient Mechanism for Checking Certificate Revocation. In *Proceedings of the 41st Conference on Local Computer Networks (LCN)*, pages 527–530. IEEE, 2016.

[22] Deepak Kumar, Michael Bailey, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, and J Alex Halderman. Tracking Certificate Misissuance in the Wild. In *Proceedings of the Symposium on Security and Privacy (SP)*, pages 785–798. IEEE.

[23] Adam Langley. Transport Layer Security (TLS) Snap Start. Technical report, IETF Internet Draft, June 2010.

[24] Adam Langley. Revocation checking and Chrome's CRL, 2012. URL `https://www.imperialviolet.org/2012/02/05/crlsets.html`.

[25] James Larisch, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. In *Proceedings of the Symposium on Security and Privacy (SP)*. IEEE, 2017.

[26] Yabing Liu, Will Tome, Liang Zhang, David Choffnes, Dave Levin, Bruce Maggs, Alan Mislove, Aaron Schulman, and Christo Wilson. An End-to-End Measurement of Certificate Revocation in the Webs PKI. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 183–196. ACM, 2015.

[27] Moxie Marlinspike. Defeating OCSP with the Character 3. *Black Hat*, 2009.

[28] Moxie Marlinspike. SSL and the Future of Authenticity. *Black Hat USA*, 6, 2011.

[29] Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560, RFC Editor, June 1999.

[30] Mark ONeill, Scott Heidbrink, Scott Ruoti, Jordan Whitehead, Dan Bunker, Luke Dickinson, Travis Hendershot, Joshua Reynolds, Kent Seamons, and Daniel Zappala. Trustbase: An Architecture to Repair and Strengthen Certificate-based Authentication. In *Proceedings of the USENIX Security Symposium (USENIX Security)*, pages 609–624, 2017.

[31] Vern Paxson. Bro: a System for Detecting Network Intruders in Real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[32] Matthew Prince. The Hidden Costs of Heartbleed, 2017. URL `https://blog.cloudflare.com/the-hard-costs-of-heartbleed/`.

[33] Khaled Rabieh, Mohamed MEA Mahmoud, Kemal Akkaya, and Samet Tonyali. Scalable Certificate Revocation Schemes for Smart Grid AMI Networks using Bloom Filters. *IEEE Transactions on Dependable and Secure Computing*, 14(4):420–432, 2017.

[34] Aaron Schulman, Dave Levin, and Neil Spring. RevCast: Fast, Private Certificate Revocation over FM Radio. In *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 799–810, 2014.

[35] Emily Stark, Lin-Shung Huang, Dinesh Israni, Collin Jackson, and Dan Boneh. The Case for Prefetching and Prevalidating TLS Server Certificates. In *19th Annual Network and Distributed System Security Symposium (NDSS)*, 2012.

[36] Pawel Szalachowski, Chuat Amann, Taeho Lee, and Adrian Perrig. RITM: Revocation in the Middle. In *Proceedings of the 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 189–200. IEEE, 2016.

[37] Emin Topalovic, Brennan Saeta, Lin-Shung Huang, Collin Jackson, and Dan Boneh. Towards Short-Lived Certificates. *Web 2.0 Security and Privacy*, 2012.

[38] Ahmad Samer Wazan, Romain Laborde, David W Chadwick, Francois Barrere, and Abdelmalek Benzekri. TLS Connection Validation by Web Browsers: Why do Web Browsers still not Agree? In *Proceedings of the 41st Annual Computer Software and Applications Conference (COMPSAC)*, pages 665–674. IEEE, 2017.

Appendix A

Examining Global Certificate Statistics

To better understand the global usage of certificate revocation, we measured the percentage of revoked certificates globally. No publication since 2015 [26], much before the recent growth in the certificate space, has published similar statistics. We collected the revocation status of each certificate tagged by Censys.io [15] as “Currently Trusted” (non-expired, trusted by Apple’s, Microsoft’s, or Mozilla NSS’s root store) on March 21st and June 17th in 2018.

A.1 Currently Trusted Certificate Dataset

Our first task to collect the revocation status of each certificate was to gather a large collection of certificates that closely represented all X.509 certificates globally. Similar to what has been done in previous revocation measurements studies [26], we started with any certificate seen in many previous Internet scans or Certificate Transparency Logs[1] and then filtered out certificates that have expired or are not trusted by any standard root store. We did not exclude certificates that were not currently being advertised as most of the revoked certificates we aim to protect against will not be normally advertised [26]. We used Censys.io [15], a search engine created to allow researchers to access data from daily Internet scans, as our initial dataset. We were given permission and access to read from their database through Google BigQuery[2].

We created our datasets by collecting all certificates tagged by Censys.io as “Currently Trusted” (non-expired, trusted by Apple’s, Microsoft’s, or Mozilla NSS’s root store). We additionally cleaned our dataset by removing duplicate, expired[3], private[4], and invalid certificates[5] (see Table A.1 for March 21st dataset cleaning). After cleaning

[1] https://www.certificate-transparency.org/

[2] https://cloud.google.com/bigquery/

[3] These expired by March 21st or June 17th receptively.

[4] Private certificates are those using a LDAP endpoint or are otherwise inaccessible. Most of these certificates returned an unauthorized status code on request.

[5] Each invalid certificate had at least one Zlint error [22].

	Raw Certificates	Expired Certificates	Private Certificates	Has Z-Lint Error	Duplicate Certificates	Unrevokable Certificates	Cleaned Certificates
Has CRL Endpoint	29,919,424	166,394	2,763	550,107	2,427,171	0	26,772,989 (89.48%)
Let's Encrypt	53,775,159	578,771	0	0	0	0	53,196,388 (98.92%)
Other	5,241,259	26,799	993,207	36,893	0	475	4,183,885 (79.83%)
Total	88,935,842	771,964	995,970	587,000	2,427,171	475	84,153,262 (94.62%)

Table A.1: The ordering and reasons we removed certificates from our March 21st dataset

the dataset, we were left with 84.2 Million certificates on March 21st and 183.9 Million on June 17th.

A.2 Performing Revocation Checking

As done in previous revocation collection efforts [25, 26], we separated certificates with CRL endpoints (March 21st: 29.9M, 33.6%; June 17th: 55.7M, 30.3%) from those with only OCSP endpoints (March 21st: 59.0M, 66.3%; June 17th: 128.2M; 69.7%). The remaining certificates (March 21st: 475; June 17th: unavailable) did not have any revocation endpoint and were unrevokable[6]. Notably of the OCSP-only certificate grouping, 53.7 Million (91.1%) and 121.0 Million (94.4%) respectively, were issued by Let's Encrypt. Because of the volume of our planned requests, we obtained permission from each CA who had issued over 1 Million OCSP-only certificates in our dataset for specific scan rates[7]. We limited our request rate to 10 requests a second for all other OCSP endpoints.

A.3 Data Collection Results

The results of our scans are summarized in Table A.2 and A.3. Overall, we found that 1.29% (1.08 Million) and 0.95% (1.75 Million) of the certificates in our datasets were revoked. This is very similar to the percent of revoked certificates seen in 2014 [26] before the Heartbleed vulnerability was discovered. When Heartbleed was discovered, the percentage of revoked certificates quickly rose to around 8% [26].

In our March 21st scan, we measured the reason codes included in CRLs or OCSP responses. We found that 520,968 revocations (48.04%) included the reason code field (see Table A.4). The vast majority (457,875; 87.89%) of revocations were reported as

[6] Of the 475 unrevokable certificates, all but 2 certificates were a root certificate, an intermediate certificate, or an OCSP Signing certificate.

[7] Let's Encrypt, Symantec, DigiCert

	Cleaned Certificates	Good Revocation Status	Revoked Revocation Status
Has CRL Endpoint	26,772,989	25,983,705	789,284 (2.9%)
Let's Encrypt	53,196,388	52,946,338	250,050 (0.47%)
Other	4,183,885	4,136,155	45,703 (0.11%)
Total	84,153,262	83.068,198	1,085,037 **(1.29%)**

Table A.2: The reported revocation status of certificates in our March 21st dataset

	Cleaned Certificates	Good Revocation Status	Revoked Revocation Status
Has CRL Endpoint	55,719,070	54,576,857	1,142,213 (3.15%)
Let's Encrypt	121,054,298	120,774,022	280,276 (0.23%)
Other	7,109,349	6,777,262	332,080 (4.67%)
Total	183,882,717	182,128,141	1,754,569 **(0.95%)**

Table A.3: The reported revocation status of certificates in our June 17th dataset

	Included a Reason Code	Unspecified	Key Compromise	CA Compromise	Affiliaction Changed	Superseded	Cessation Of Operation	Certificate Hold	Privilege Withdrawn	AA Compromise
Has CRL Endpoint	483,816	1,485	3,358	10	5,159	11,624	457,875	676	3,629	0
Let's Encrypt	0	0	0	0	0	0	0	0	0	0
Other	37,152	37,056	45	1	9	41	0	0	0	0
Total	520,968	38,541	3,403	11	5,168	11,665	457,875	676	3,629	0
Total Percentage	100%	7.40%	0.65%	0.002%	0.99%	2.24%	87.89%	0.13%	0.67%	0.00%

Table A.4: The reported revocation reason of revoked certificates in our March 21st dataset

"Cessation Of Operation". Only 3,403 (0.65%) were revoked for key compromise and 11 (0.002%) were revoked for CA compromise.

A.3.1 Privately Used Certificates

While there were 19.1 Million revocations included in the CRLs (703 MB) we collected in March, there were only 788,630 (6.6%) with a CRL endpoint. In June we found 14.1 Million revocations included in the CRLs (601 MB) with only 8.0% were certificates in our June dataset. A similar imbalance was noticed in 2017 [25] where 12.7 Million revocations were found while only small percentage of these certificates were previously seen in Internet scans and were not expired[8]. While we cannot be sure where these certificates are used, we suspect they are either privately used certificates or certificates that should have already been removed from a CRL, i.e. expired. If only revoked but otherwise valid certificates with publicly accessible CRL endpoints seen in our dataset (non-expired, CA-trusted, seen in previous Internet scan) were included in CRLs, we estimate the cost of downloading all accessible CRLs would be reduced to only 38 MB and 48 MB receptively. However, because these unseen certificates are signed by CAs and are trusted by browsers, it would be irresponsible for CAs to naively remove these from their CRLs.

[8] The authors communicated this information through email.