Leveraging Locality of Reference for Certificate Revocation

Luke Dickinson[*]
Sandia National Laboratories
ldickin@sandia.gov

Trevor Smith, Kent Seamons
Brigham Young University
tsmith@isrl.byu.edu,seamons@cs.byu.edu

ABSTRACT

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 design assumes that locality of reference applies to the certificates accessed by an organization. The CRT periodically checks the revocation status of X.509 certificates recently used by the organization. Pre-checking the revocation status of certificates the clients are likely to use avoids the security problems of on-demand certificate revocation checking.

To validate both the effectiveness and efficiency of our approach, we simulated a CRT using 60 days of TLS traffic logs from Brigham Young University to measure the effects of actively refreshing revocation status information for various certificate working set window lengths. A working set window size of 45 days resulted in an average of 99.86% of the TLS handshakes having revocation information cached in advance. The CRT storage requirements are small. The initial revocation status information requires downloading a 6.7 MB file, and subsequent updates require only 205.1 KB of bandwidth daily. Updates that include only revoked certificates require just 215 bytes of bandwidth per day.

CCS CONCEPTS

• **Security and privacy** → *Network security*; *Web protocol security*.

KEYWORDS

certificate revocation, caching, working sets, locality of reference

ACM Reference Format:
Luke Dickinson and Trevor Smith, Kent Seamons. 2019. Leveraging Locality of Reference for Certificate Revocation. In *2019 Annual Computer Security Applications Conference (ACSAC '19), December 9–13, 2019, San Juan, PR, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3359789.3359819

[*]This work was completed while the author was a graduate research assistant at Brigham Young University.

ACSAC '19, December 9–13, 2019, San Juan, PR, USA

ACM ISBN 978-1-4503-7628-0/19/12…$15.00
https://doi.org/10.1145/3359789.3359819

1 INTRODUCTION

Browsers rely on X.509 certificates for host authentication when establishing HTTPS connections. A trusted Certificate Authority (CA) usually signs each certificate. Whenever a certificate's private key is compromised, the website administrator should notify the CA to revoke the certificate. All clients that may receive the certificate require notification of the revocation. Otherwise, connections to compromised websites are vulnerable to a man-in-the-middle attack until the certificate expires.

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 [27], protect a small percentage of certificates [1, 2], or are vulnerable to downgrade attacks when access to revocation information is critical [12, 16, 27]. The remaining strategies have other debilitating weaknesses slowing and halting adoption such as requiring significant infrastructure changes [32, 34], requiring participation and additional costs by CAs and third parties [24, 34], or exposing new attack surfaces [3]. Because every revocation strategy has a significant cost or flaw, certificate revocation is mostly being ignored by clients, leaving many computers and smartphones vulnerable to man-the-middle attacks [25].

In this paper, we propose Certificate Revocation Table (CRT), a new revocation status dissemination strategy to regularly check the revocation status of the certificates recently used by an organization, such as clients on a university's private network. Network administrators deploy this strategy at scale to protect their clients. By leveraging locality of reference of web requests, a CRT provides the revocation status of a high percentage of TLS handshakes while minimizing bandwidth and reducing the attack surface.

Evidence shows [9] that a vast majority of TLS handshakes made by an organization's clients reuse recently seen certificates within the space of an hour. We find that this idea continues to hold even when a client's cache includes multiple weeks of data. Because effectiveness is maintained even in the presence of this delay, we can design a flexible revocation strategy that allows administrators to adjust the rate clients receive updates from a supporting server based on their organization's needs, while still protecting clients. The bandwidth consumption of our design is competitive with alternative solutions with higher data density because even large private networks, such as a university network [7, 18], use only a small fraction of the globally available X.509 certificate space.

In this work, we make two significant contributions: (1) We provide a novel certificate revocation strategy that is competitive with alternative state-of-the-art strategies. It enables network administrators to protect their clients without relying on third parties and preserves the privacy of clients' browsing patterns. (2) We are the first study to examine how an organization's X.509 certificate working sets change over time. We examine passively collected TLS traffic logs from the Brigham Young University (BYU) campus network over a two-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 [9, 20].

In the rest of the paper, we present the system design of a certificate revocation table, including the operations and data items it contains (Section 3). Using 60 days of traffic logs from BYU, we measure and report the effectiveness and efficiency of a reference implementation of a CRT (Section 4). We discuss the implications of the results (Section 5) and compare CRT to alternative revocation strategies (Section 6).

2 RELATED WORK

Prior studies have passively measured certificate use [6, 7, 9, 18, 20]. Two studies provide evidence that the principle of locality of reference applies to certificate requests. Bates et al. [9] found that 99.3% of the HTTPS requests from a college campus during one hour were duplicates. While examining the effects of caching OCSP responses, Hu et al. [20] report that 95% of the certificate requests at their university during 30 minutes were duplicates.

Certificate revocation strategies fall into three groups: pull, push, and network-assisted. Clients using **Pull-Based Certificate Revocation** request revocation status information whenever it is needed. Certificate Revocation Lists (CRLs) [12] and Online Certificate Status Protocol (OCSP) [27] are the most common examples.

A CRL is a signed list of all the revoked certificates for a CA. Scalability is the main criticism against CRLs because they can grow quite large. For example, Apple published a 76 MB CRL [25]. Mozilla Firefox and Google Chrome have disabled revocation checking using CRLs because of bandwidth and latency concerns.

OCSP [27] responders provide a signed certificate revocation status for individual certificates. Clients issue blocking requests to obtain the revocation status for each HTTPS certificate and wait for the response before loading the page. OCSP requests are unencrypted, divulging detailed client traffic patterns to the CA and all nodes along the path. For a delayed response, clients typically soft-fail and continue the TLS handshake, assuming the certificate is likely not revoked. Despite these drawbacks, most modern desktop browsers provide support for OCSP.

Stark et al. [33] use content-based prefetching to eliminate the page loading delays created by OCSP and TLS as they gather parameters required for TLS Snap Start [22]. This strategy also soft-fails because revocation status checking occurs only seconds before it is needed.

Clients use **Push-Based Certificate Revocation** to regularly receive revocation information, which increases the probability that the status information is available in the cache when they need it. Even though clients gather the status of more certificates than they need, this approach does not reveal client traffic patterns.

Some approaches, such as Google's CRLSets and Mozilla's OneCRL, minimize bandwidth by including only a small, hand-picked set of revocations for popular websites. Google pushes periodic updates to Chrome with a small list of revoked certificates called a CRLSet. CRLSets have a maximum size of 250 KB [2], which equates to a capacity of about 40,000 revoked certificates. A CRLSet is intended to include only revocations where the risk of a compromised certificate is suspected. 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. Rabieh et al. [31] showed how using two Bloom filters in tandem, one identifying revoked certificates and the other identifying non-revoked certificates, drastically reduced false-positive rates. Larisch et al. [24] 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. The algorithm repeats this process until finding a Bloom filter with no false-positive entries. The server requires substantial network resources to build 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 MB.[1]

Network Assisted Certificate Revocation eliminates the need for a client to request a revocation status. Instead, it modifies the TLS ecosystem to address revocation. Several approaches support revocation via middleboxes. Revocation in the Middle (RITM) [34] is a solution that distributes revocation information to middleboxes throughout the Internet via a CDN. These middleboxes intercept traffic and perform revocation checks. Hu et al. [20] proposed Certificate Revocation Guard (CRG), which uses a middlebox to intercept all TLS traffic for an entity and check revocation. The paper suggests 95% of TLS handshakes use a cached revocation status, but it only reports the results of a short 30-minute experiment involving approximately 2,000 connections.

OCSP Stapling [16] requires each website administrator, instead of end clients, to make an OCSP request for their certificates. The 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 a man-in-the-middle downgrade attack. OCSP Must-Staple [17] was proposed to solve the downgrade attack, but introduces the risk of the website becoming inaccessible to clients during a DoS attack targeted at OCSP responders. However, others [25] have argued that the potential for a DoS attack is not a fundamental problem as a CDN could distribute static revocation information. Even so, OCSP Must-Staple has suffered other problems, including CA inconsistencies and bugs in server implementations, that have slowed adoption to 0.02% [10, 11]. Due to these concerns, Google Chrome currently does not support the certificate extension for OCSP Must-Staple [3, 36].

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 [19] to just a few days [35]. This strategy requires the server to renew its certificate regularly. While it was previously

[1] We used source code provided by the authors.

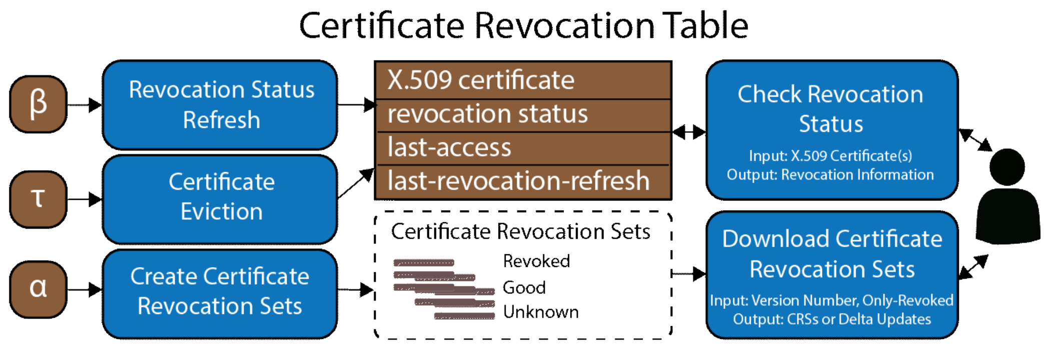

Figure 1: The design of a certificate revocation table. Periodically, the CRT runs the Revocation Status Refresh, Certificate Eviction, and Create Certificate Revocation Sets functions. Clients access revocation information through the Download Certificate Revocation Sets and Check Revocation Status APIs.

not practical to change public keys on renewal [35], the emergence of new technology such as the ACME protocol [8] and the EFF's CertBot[2] enables automatic public key rotation on renewal. If a private key compromise occurs, the server administrator does not renew the certificate.

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 working set W(t, τ) of a process is frequently reused by a future working set W(t + α, α), as long as α is small. As α grows, the frequency of reused entries decreases. 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 follows this same principle of locality. A certificate working set W(t, τ) contains a majority of the certificates in the working set W(t + α, α).

A certificate revocation table (CRT) (see Figure 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 4.1 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.

3.1 Certificate Revocation Table Design

A certificate revocation table (CRT) enables clients to create and share a certificate working set. A CRT updates or removes table entries at regular intervals. A CRT also inserts incoming certificates from clients, 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.[3] The revocation status reports the last received status, which can be *Good*, *Revoked*, *Unknown*, or *New*. A *New* status indicates the CRT has not yet attempted a revocation status request and an *Unknown* status indicates the CRT has never successfully retrieved a revocation status. Each time a CRT inserts a certificate, it updates the last-access timestamp 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.

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

(1) Effectiveness: Vulnerable to Downgrade Attacks. — If a revocation status cannot be determined because the authoritative server is not accessible, modern browsers will *soft-fail* by assuming the certificate is not revoked. Soft-failing values website accessibility over the risk of accepting a revoked certificate. The soft-fail approach 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 [23] 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, 2, 24] reduces the number of vulnerable on-demand checks required. To successfully

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

[3] The use of OCSP for revocation includes information from an issuer certificate.

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 in contrast to blocking access to a revocation endpoint only at the time of the malicious connection. We expect this makes it more difficult for attackers to remain undetected.

(2) 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. Strategies repeat these downloads or requests from once a week to many times per day to maintain a sufficiently up-to-date certificate revocation status. This cost can limit the participation of clients, especially those with a monthly bandwidth limit. Liu et al. [25] found that no mobile browser evaluates any certificate revocation status and suggests bandwidth concerns are to blame.

A CRT provides the revocation status for certificates the clients are likely to use soon (i.e., an organization's certificate working set). The network bandwidth consumption of a CRT's and its clients scales to the organization a CRT is protecting and parameters set by the network administrators. A CRT that reports only revoked certificates consumes a small fraction of client bandwidth.

(3) 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 issuing certificates 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 [5] 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 [24]) to December 2017 (78.5 million [4]), and then quadrupled again in just over half of a year from December 2017 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 [24]. Other strategies avoid the growth in size by opting to limit the number of certificates they protect as with CRLSets [2].

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

(4) 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.[4] In 2014, Liu et al. [25] 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 [30]. 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" [15].

During a mass revocation event, the CRT's bandwidth consumption remains steady because the CRT already provides the revocation status of each certificate in the working set regardless of its revocation status. Delta updates downloaded by clients are more expensive while revoking a large number of certificates but return to their normal size soon after. Bandwidth consumption increases for other revocation strategies such as CRLs [12] or CRLite's full Bloom filter cascade [24] whose bandwidth requirements depend on the number of globally valid, but revoked certificates. These strategies maintain this significant increase in bandwidth until the revoked certificates begin to expire.

(5) 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 [24], and the median OCSP response for the Alexa top 1 Million has a lifetime, and therefore suggested update rate, of 4 days [24]. A push-based revocation strategy that updates clients once a day [2, 24] has worst-case revocation timeliness of almost 48 hours (see 4.1).

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

(6) Privacy: Exposing Client Traffic Patterns. — Some revocation strategies, such as OCSP [27], share detailed client traffic patterns to revocation endpoints.

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

(7) Deployment Requirements and Incentives. — Challenging deployment requirements and weak incentives can slow or halt the adoption of a revocation strategy. For example, CRLite had acceptable deployment requirements in January of 2017; however, 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). OCSP Must-Staple faces even stronger deployment hurdles as it requires certificate owners to commit to obtain OCSP status regularly from historically unreliable OCSP responders [11]. Any mistakes cause websites to 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. A CRT's bandwidth consumption is scaled based on the size of the organization's certificate working set as oppose to the number of globally live certificates.

3.2 Periodic System Tasks

Each of the following tasks is periodically run to update a CRT and to prepare the revocation information served by the APIs.

[4]http://heartbleed.com

3.2.1 Revocation Status Refresh. The CRT rechecks the revocation status of each certificate it contains every β hours.[5] Decreasing β increases revocation status freshness and a CRT's bandwidth consumption.

The last-revocation-refresh field is used to manage revocation status rechecking. The certificate revocation status check occurs when adding a certificate 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, is inaccessible [11]. In this case, the CRT continues to repeat the request while introducing increasing delays between requests (i.e., exponential backoff) to avoid an accidental DoS attack on a revocation endpoint.

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 the number of requests reduces the bandwidth consumption of a CRT. Second, eviction reduces the amount of storage used by a CRT and its clients. Both of these values can grow without bound without certificate eviction.

Periodically, a CRT evicts certificates that are no longer in the organization's certificate working set. The working set at time t is W(t, τ), which includes all active certificates from time t - τ to t. We evict all certificates with a last-access timestamp older than t - τ. Future work describes the advantages of an approach that avoids evicting known revoked certificates even if they have left the working set (see 7.2).

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 that CRT effectiveness increases while the efficiency decreases.

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 (see 7.1).

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 overlap between working sets, the α value also defines how often clients should download updated CRSs.

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 whether to avoid a suspicious connection. By including each of these CRSs, clients can determine the trusted certificates, revoked certificates, certificates with inaccessible status, certificates that have never been accessible, and entirely new certificates. 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 [24] 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.

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

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 can download differing versions of CRSs depending on their needs.[6] 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*, *Good*, and *Unknown* certificates. A valid version number results in downloading either the group of CRSs or all deltas from the client's version to the most up-to-date version, whichever is smaller. If the only-revoked boolean parameter is true, the download includes only the CRS or group of deltas including revoked certificates. For the majority of clients today who soft-fail on an unknown revocation status [25], downloading only the CRS containing revoked certificates is as effective as downloading all three CRSs (*Revoked*, *Good*, 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, the group of CRSs will not always contain the revocation status of future traffic because of the creation of new certificates in between updates and clients visiting new websites. For certificates not included in these CRSs, clients must fall back to alternative validation methods[7], such as the Check Revocation Status API.

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. The first purpose is to collect traffic information. The API receives a new certificate and inserts it as a new row in the table. If the certificate is already present in the table, the last-access timestamp is updated. This API is the only way a CRT receives a client's certificate usage information.[8] The second purpose is to provide clients an on-demand revocation

[6] In practice, we could replace this API with a push-based model.

[7] Clients using only the CRS containing revoked certificates assume a missing certificate is not revoked and therefore do not fall back to alternative validation methods.

[8] 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 [34], automatically block revoked connections for the user [20, 28], or passively observe traffic. Without a middlebox, it may be preferable for clients to periodically share the set of recently seen certificates to this API.

check. When clients call this API, the CRT immediately returns the revocation status of each certificate requested as recorded in the CRT. The last-revocation-refresh timestamp accompanies an out-of-date revocation status. These timestamps inform clients how long a revocation status has been inaccessible. As the revocation status of newly added certificates is immediately checked and recorded (see 3.2.1), a strict client unwilling to trust an *Unknown* status on first use can delay finishing the TLS handshake and repeat their request to this API after a short delay.[9] Assuming the revocation endpoint is accessible to a CRT, a *Good* or emphRevoked response is returned to the client following the second request.

There are several issues to address 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 5.3). Second, care should be taken to enforce who can access this endpoint because the organization's working set includes all certificates sent to this API. If a client from outside the targeted organization uses the endpoint, the working set may become polluted with certificates not used by the targeted organization.

3.4 Threat Analysis

The threat model includes active network attackers that can (1) modify or replay revocation information transmitted over the network, and (2) block access to revocation status information. We assume the CRT and local clients using the CRT are trusted.

A CRT collects only signed and trusted revocation information. Assuming the CRT is trusted, CRSs and on-demand responses created by a CRT contain valid revocation information. Since information generated by a CRT includes a cryptographic signature and a timestamp, clients can detect replay attacks and attempts by malicious actors to generate or modify 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.

4 ANALYSIS

In this section, 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 Design Parameters

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

(1) The certificate working set window length, τ, determines when to remove certificates from a CRT. Lengthening the window increases the number of certificates in a CRT that 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.

(2) The rate of CRS creation, every α hours, defines the frequency clients can retrieve updates from a remote CRT. 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 24 hours.

(3) The rate of certificate revocation status checking, every β hours, in a CRT affects both the revocation timeliness and bandwidth consumption. The worst-case revocation timeliness[10] is $\alpha + \beta$. To achieve a revocation timeliness equivalent to competing systems [2, 24] (48 hours), we set β to 24 hours.

To illustrate this worst-case scenario in revocation timeliness for a 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. 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 downloading the second day's CRS almost 24 hours later do not learn that the owner already revoked the certificate. 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 continue to trust this revoked certificate until they download the third day's CRS, which is almost 48 hours after the owner revokes the certificate. This worst-case scenario applies to similar revocation strategies [2, 24] that periodically provide revocation updates to clients.

4.2 Experiment Methodology

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

4.2.1 Passive Dataset Collection. Coordinating with the network administrators at BYU, we obtained logs generated by the Bro Network Security Monitor [29] for SSL/TLS traffic and X.509 certificate information from 2018-04-17 to 2018-06-17. The logs included outbound TLS traffic for the university campus network.

Several precautions were taken to protect the privacy of collected data. First, all researchers with access to the data signed non-disclosure agreements. Second, network administrators provided

[9] A client should wait long enough for the CRT to reach the revocation endpoint (i.e., an OCSP responder) and update its database. The length of this delay is dependent on network infrastructure; a suitable wait time can be determined through experimentation.

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

and secured a computer for performing all of the raw data processing. Third, all experimental results were reported in aggregate form, never revealing any hostnames, IP addresses, or identifying certificate information. Finally, BYU network administrators reviewed and verified that the results disclosed no private information.

4.2.2 Data Analysis. Using an implementation of a CRT (see 4.3) and TLS traffic logs from BYU, we measured the following six metrics to evaluate the effectiveness and efficiency of our revocation strategy.

(1) 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 a periodically-created CRS. We reported the total across all certificates regardless of status and also across only the revoked certificates.

(2) 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 clients used that may be exploitable. 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.

(3) CRT total certificates. — The total number of certificates in the CRT.

(4) CRT idle certificates. — The total number of certificates in the CRT not 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 we could identify and evict these certificates early, efficiency could increase while effectiveness would remain the same.

(5) 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 download CRSs regularly. In our experiments, we estimated the CRT's bandwidth cost by assuming OCSP was used to check each stored certificate's revocation status regularly. A client's bandwidth requirement is only the cost to download CRS delta updates regularly.

(6) Total storage. — The total storage used 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. 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. We measured the effects of regularly using the periodically created CRSs, created every α (24) hours, to check each revocation status. Revocation information was updated only every β (24) hours. We chose α and β to match the CRT's timeliness with the revocation strategies we selected for comparison. If a revocation status was unknown to 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 stabilizes because the CRT may begin to evict certificates.

4.2.3 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.

4.3 Reference Implementation

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

4.3.1 Revocation Status Refresh. An important goal of our experiments was to collect the revocation status of each certificate in a CRT so that we could 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 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. For each revoked certificate, we included the time at which it was revoked. While checking the revocation status for a certificate, we returned a *Revoked* status only when the query occurred after the time of revocation.

4.3.2 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 no client had used it from time t - τ to t.

4.3.3 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 stored as 28 to 34 characters in hexadecimal format, a significant saving compared to using SHA-256 hashes (64 characters). Because certificate serial numbers are unique only for a given issuer, we categorized entries by the SHA-256 hash of the issuer certificate. We created a separate list for two certificate status options: *Good* and *Revoked*.[11] 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, searching these lists on disk requires O(log(n)) time with two binary searches. Searching the lists in an in-memory set implementation requires O(1) time.

We utilized the LZMA compression algorithm 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.

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

τ: working set window length	TLS handshakes with known status		Certificates with known status		CRT total certificates	CRT idle certificates	Daily network bandwidth		Total storage	
	Any Certificate	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 1: The average values for each of the six evaluation metrics for several values of τ. We excluded values from a CRT's initial learning period from the aggregation.

We created delta updates 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.

4.4 Results

This section contains the results for the six evaluation metrics described in our methodology using the passive traffic logs and our reference implementation (see Table 1 and Figure 2). In the dataset, there were 524,597 Internet-usable unique certificates.[12] 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 transmitted within the 228,427 TLS connections in our dataset, including a minimum of 14 revoked certificates each day.

4.4.1 TLS handshakes with known status. 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. 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[13]. This locally cached revocation information is accessible even in the presence of Internet traffic manipulation at the time of the connection.

4.4.2 Total certificates with known status. On average, given a τ value of 1 day, the revocation status of 60.63% of unique certificates used by our organization was available to clients via 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, the percentages increased for each specified value of τ (77.42% for τ equal to 1 day; 95.28% for τ equal to 45 days).

4.4.3 CRT total and idle certificates. Both the CRT total and idle certificates increased with larger τ values.[14] We noticed that the percentage of idle certificates was high (> 40%) regardless of the value of τ used, leaving room for a significant efficiency gain if these idle certificates could be detected and evicted early.

4.4.4 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[15] and that each OCSP response used 1.3 KB [24] 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. This cost is significantly less than the bandwidth required for every individual client in an organization to perform OCSP requests (averaging 85.64 GB per day for the organization).

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 Figure 3). Then the client would download a delta update once per day, averaging 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 group of CRSs (0.73 MB versus 0.89 MB). On average, delta updates to the CRS containing 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 decreases 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 explains why the delta updates are larger at small τ values.

4.4.5 OCSP request rate. For the CRT created in our experiments 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

[12] 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.

[13] Unlike our experiments, in practice we advise to take extra care before evicting revoked certificates from the CRT before they have expired.

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

[15] We set β to 24 hours our experiments.

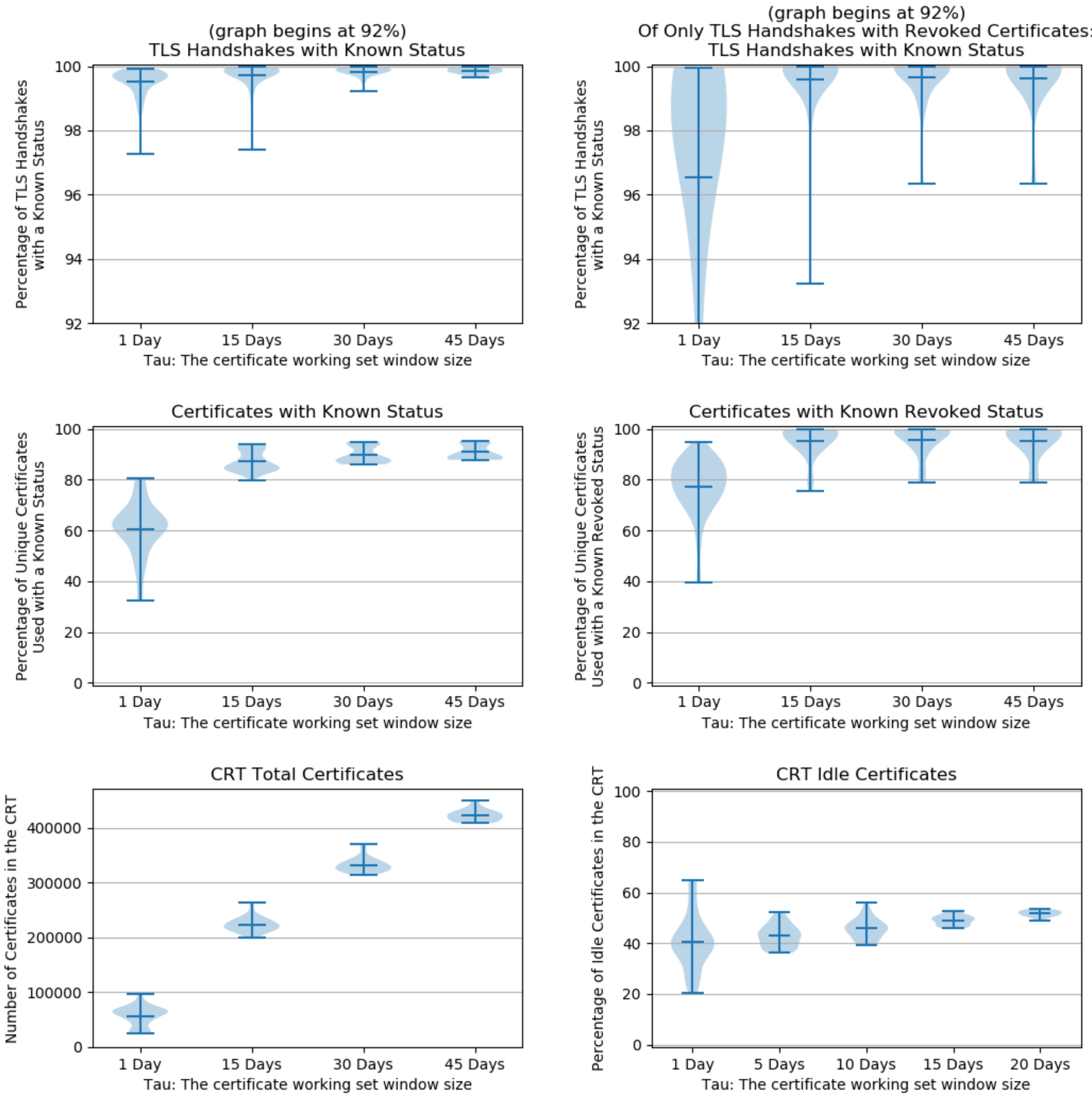

Figure 2: The distribution of our daily results for several values of τ for the following metrics: TLS handshakes with known status, TLS handshakes with known revoked status, Certificates with known status, Certificates with known revoked status, CRT total certificates, and CRT idle certificates. We excluded values from a CRT's initial learning period.

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.

4.4.6 Total storage. 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 total storage used by a 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 that every certificate in the CRT has a different issuer. We expect the actual cost to be just over half of this upper bound because a small number of CA's [4] issue most of the certificates.

End clients only need to store an uncompressed group of CRSs on their device (see Figure 3). Our experiments found this file size to be between 1.71 MB and 12.75 MB depending on the value of τ chosen.

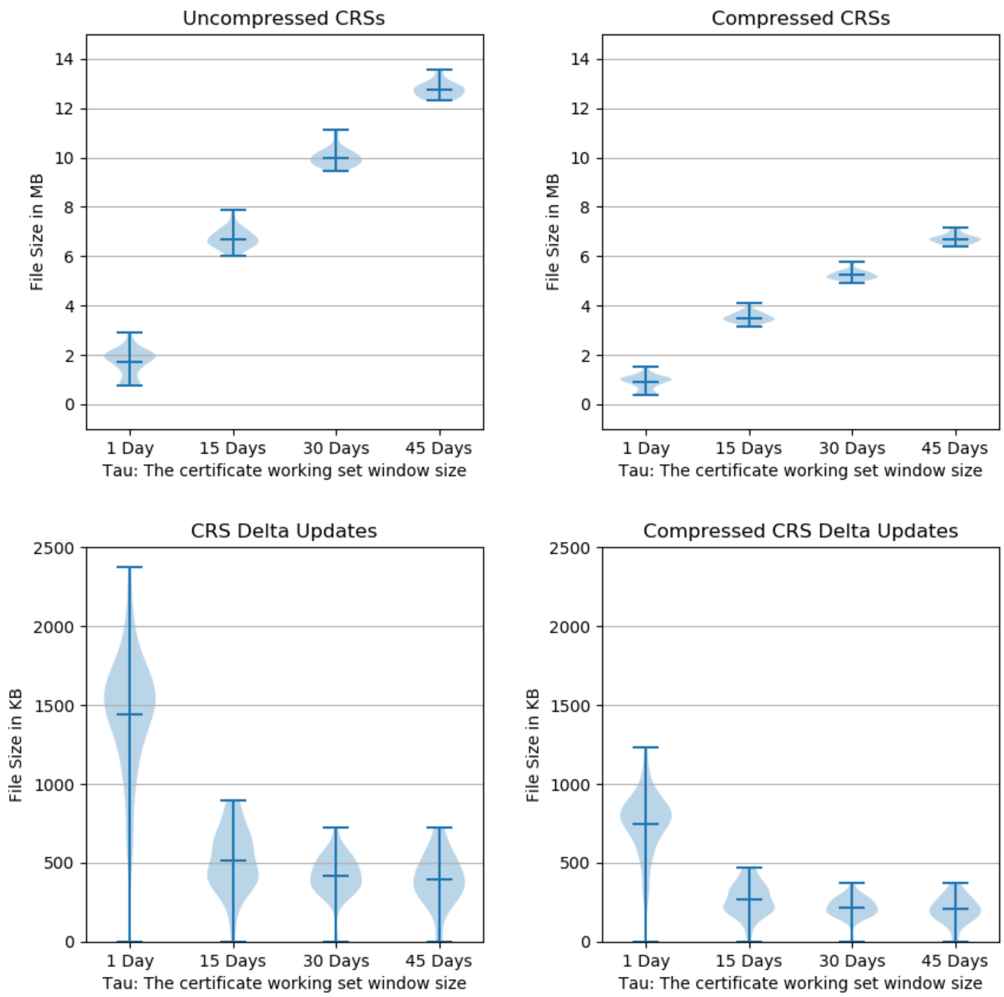

Figure 3: The distribution of file sizes for the CRSs and deltas created in our experiments for several values of τ. CRSs created during a CRT's initial learning period are excluded.

5 DISCUSSION

Our results have implications for system admins, mobile devices, and user privacy.

5.1 Empower System Admins

We believe the dependency on third-party adoption has hampered the deployment of many otherwise promising revocation strategies. Scalable revocation strategies with no reliance on third parties empower system admins to defend themselves. The incentives to deploy CRT and other similar solutions [20, 28, 31] are with the system admins responsible for securing their organizations.

System admins choose the CRT parameters (τ. β, α) to balance effectiveness, efficiency, and revocation timeliness depending on their needs. For example, assuming 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 improves revocation timeliness. Many organizations could improve revocation timeliness with low cost because a CRT regularly rechecks only a small subset of globally trusted certificates. Based on our results, cutting alpha and beta in half (12 hours) increases the daily bandwidth cost to 9.74 OCSP requests per second and consumes 1.05 GB of network traffic.

5.2 Mobile Platforms

Bandwidth consumption discourages browsers on mobile platforms to check revocation [25] since clients often have bandwidth limits. A CRT can be mobile-friendly by dynamically changing the behavior of the system based on the network conditions. For example, the system should attempt to update CRSs or use the Check Revocation Status API while connected to a Wi-Fi network. If significant time has passed without connecting to a Wi-Fi network, clients should only collect the most critical revocation information such as the

	TLS Handshakes Protected	Client Bandwidth Consumption	Global Certificate Growth Scalability	Mass Revocation Event Scalability	Revocation Timeliness	Privacy Preserving	Deployment Requirements
OCSP Must-Staple	100%†	1.3 KB per TLS handshake [24]	Minimal BG	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)∗	100%	Initially 10 MB; 580 KB per day	Significant BG	Significant BG	1–2 Days	Yes	High
CRLite (Mar. 2018)∗	100%	Initially 18 MB; Unknown per day	Significant BG	Significant BG	1–2 Days	Yes	High
CRT	99.86%	Initially 6.71 MB; 205 KB per day	Minimal BG	Minimal BG	1–2 Days	Yes	Medium
CRT (only revoked)	99.86%	Initially 1.92 KB; 0.21 KB per day	Minimal BG	Significant BG	1–2 Days	Yes	Medium

Table 2: Comparison of our CRT implementation to other revocation strategies. (BG = Bandwidth Growth)

∗: The number of globally live X.509 certificates was 30 million in January 2017 and 84 million in March 2018.
†: While protecting 100% of TLS handshakes is possible, current adoption rates (0.03% of certificates) practically protect 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 is low.

CRS containing revoked certificates. In our dataset, regularly downloading this CRS and its updates for an entire month uses fewer than 0.0003% (6.30 KB) of a 2 GB monthly bandwidth allotment.

5.3 Privacy

Clients should not have to reveal traffic patterns to untrusted third parties to check revocation status. We describe three different ways a CRT can be deployed in an organization while preserving privacy.

A private network administrator could use an **organization-based middlebox** to inspect network traffic. A CRT could be integrated into the middlebox to support certificate revocation, which would not sacrifice client privacy because network administrators already have access to the information.

If clients in an organization do not share a physical or virtual network, not all traffic may go through a middlebox. We could deploy a CRT as a **private cloud service**. We assume that 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. The service generates the CRT working set from the certificates obtained through the *Check Revocation Status* API. Communication between clients and CRTs should be encrypted to prevent eavesdropping and maintain privacy.

An **honest-but-curious third party**, such as a group of security enthusiasts, could deploy a distributed group of CRTs. Anonymization techniques such as the *notary bounce* defined by the Convergence Notary [26] or Tor may protect client privacy.

6 COMPARISON

This section compares CRT with OCSP Must-Staple, CRLSets, and CRLite based on the seven concerns facing revocation strategies discussed in Section 3.1. Table 2 summarizes the results. The CRT details in our comparison are from TLS traffic generated at BYU between 2018-04-17 to 2018-06-17 and assume a τ value of 45 days, a β value of 24 hours, and an α value of 24 hours.

Effectiveness. If clients use CRLite or all web servers adopt OCSP Must-Staple, they can check revocation on 100% of connections. On average, clients using either type of CRSs achieve 99.86% coverage. CRLSets protect only a small set of critical certificates. CRT could approach 100% by modifying the design parameters. For example, reducing the time between delta updates, α, may increase effectiveness while only slightly increasing bandwidth.

Efficiency. The daily client bandwidth and storage usage for CRT are slightly less than CRLite and CRLSets for the full group of CRSs and three orders of magnitude lower than any other strategy for the revoked-only CRS. The bandwidth requirement for OCSP must-staple depends on client activity and is roughly equal to the others assuming 200 − 500 connections per day.

Certificate Growth Scalability. The CRT bandwidth consumption increases according to the number of unique certificates an organization uses (i.e., working set size), a significant improvement compared to CRLite whose bandwidth is a factor of the number of live certificates globally. OCSP Must-Staple bandwidth consumption increases with the number of TLS connections, which is indirectly affected by certificate growth. CRLSets is fixed size, so there is no bandwidth scalability concern. However, it protects a decreasing percentage of overall certificates.

Mass Revocation Event Scalability. The bandwidth for CRT and OCSP Must-Staple do not change because they already collect the status of each certificate. The size of delta updates for a full group of CRSs increases due to the re-issuing of certificates during mass revocation events. The CRS containing revoked certificates significantly increases in size during a mass revocation event but would still be only a fraction of the size of the other CRSs. CRLite bandwidth increases during mass revocation events because it is partially a factor of the percentage of globally revoked certificates. CRLSets miss many revoked certificates during a mass revocation event since they protect only a small number of certificates.

Revocation Timeliness. Clients using CRT (with α and β both set to 24 hours), CRLite, and CRLSets download revocation information once every day. Worst case, a revoked certificate could be trusted by a client almost 48 hours after the initial revocation (see 4.1). The worst-case revocation timeliness for OCSP Must-Staple is equal to the OCSP response's lifetime (median expiration is 4 days [24]) and occurs when an active attacker obtains an OCSP response immediately before revocation.

Privacy. OCSP Must-Staple, CRLSets, and CRLite do not collect nor expose client traffic patterns. CRT may reveal client traffic patterns at collection points, but there are deployment options to maintain privacy (See 5.1). CRT reveals the aggregated collection of certificates to all clients in an organization, which could alert members of the organization when some client visits a controversial website (e.g., addiction, politics, religion, suicide help). It does not identify a specific client or any client's browsing history, assuming a sufficiently large population, but inferences could cause harm and result in a loss of privacy or reputation.

Deployment Requirements. CRT requires a dedicated server and client software, similar to both CRLite and CRLSets. OCSP Must-Staple requires buy-in from each website and high availability from CAs to avoid service interruption to website customers. The risk of downtime is currently a huge disincentive.

CRLite places a significant load on supporting servers. As of September 2018, CRLite collects the revocation status for over 317M certificates [4]. This total includes at least 213M issued by Let's Encrypt that 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 to avoid overwhelming Let's Encrypt. In contrast, our experiments show that a CRT running in our organization required 3 orders of magnitude fewer OCSP requests per second compared to CRLite.

7 FUTURE WORK

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

7.1 Removal of Irrelevant Certificates

Our results report a large percentage (> 40%) of idle certificates independent of the certificate working set window size (τ). Future work could determine ways to identify and evict these certificates earlier to increase efficiency. This effort requires caution as certificates evicted prematurely could result in loss of effectiveness. 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. In addition, memory management eviction strategies such as Least Recently Used (LRU) and Least Frequently Used (LFU) may be effective for implementations that adopt a fixed-size cache.

7.2 Idle but Revoked Certificates

In our experiments, we chose to evict idle certificates after they had left the certificate working set regardless of whether they were a *Revoked* or *Good* certificate. In practice, a system administrator may choose to use two certificate working sets, one for revoked certificates (defined by τ'), and one for all other certificates (defined by τ). Increasing the window size of the revoked certificate working set would increase the total storage usage of both the CRT and the CRS containing revoked certificates. The delta update for the CRS containing revoked certificates would not increase in size, and in cases where a revoked certificate was removed and reinserted, the delta update would instead be slightly smaller. We suspect that increasing the window size of the revoked certificate working set would increase the number of *TLS handshakes with known revoked status* without significantly affecting efficiency.

7.3 Anticipating Certificate Renewal

We suspect that some of the TLS handshakes with an 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. We estimate that on an average day, approximately 1.11% of the certificates issued by Let's Encrypt expire and need replacing.

A CRT could increase its effectiveness by detecting that a host has obtained a new certificate before any client in the organization receives the new certificate. A CRT could periodically check the host certificate for all the cached certificates that are nearing expiration. A measurement study could be conducted to determine when hosts typically acquire a new certificate when the current certificate is about to expire.

7.4 Alternative Deployment Scenarios

In our research, we described and experimented using a CRT to create a shared certificate working set for all traffic from a large organization (i.e., a university). Future work could explore the effectiveness, efficiency, and deployment requirements of shifting to a different type of targeted population. We describe a few deployment options we have considered as possible use cases:

A **single client** could monitor its traffic patterns and manage a CRT. Deployment would be simplified, but we expect reduced effectiveness because the certificate working set is much smaller.

A **home network** could leverage a localized CRT that targets a personalized certificate working set to protect a small number of desktop computers, smartphones, and often-vulnerable IoT devices. CRT support built into a home router would simplify deployment requirements.

A certificate working set created by the Internet traffic of a**general region** (i.e., a city) could be collected and used by an Internet service provider to manage a CRT. Two critical issues to address are privacy concerns and how to manage the size of the CRT so that it doesn't grow too large.

As explained by Rabieh et al. [31], a revocation strategy is needed to support a **smart grid AMI network** using a multi-hop wireless mesh topology. In this topology, power meters can route messages through other power meters, in which validating the message's authenticity is essential. These meters could store CRSs to reduce the number of required online revocation status checks compared to traditional strategies.

8 CONCLUSION

This paper presents 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 determine the effectiveness and efficiency of this approach, we simulated a CRT using 60 days of TLS traffic logs from BYU to measure the effects of actively refreshing certificates for various certificate working set window lengths. Our results demonstrate that 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.

ACKNOWLEDGEMENTS

We thank the Brigham Young University Office of Information Technology for their help in gaining access to outbound TLS traffic for the university campus network. We thank Daniel Zappala, Casey Deccio, and the anonymous reviewers for their helpful comments and feedback. This research is supported in part by the National Science Foundation under Grants No. CNS-1528022 and CNS-1816929.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND No. 2019-11307 C.

REFERENCES

[1] CA:RevocationPlan. https://wiki.mozilla.org/CA:RevocationPlan#OneCRL.
[2] CRLSets. https://dev.chromium.org/Home/chromium-security/crlsets.
[3] Feature request: OCSP Must Staple (RFC 7633).
[4] Censys, 2017. https://censys.io/certificates?q=tags.raw%3A+%22trusted%22.
[5] Let's Encrypt Stats, 2017. https://letsencrypt.org/stats/.
[6] 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.
[7] 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.
[8] Richard Barnes, Jacob Hoffman-Andrews, Daniel McCarney, and James Kasten. Automatic Certificate Management Environment (ACME) draft-ietf-acme-acme-12. Internet-Draft, April 2018.
[9] 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.
[10] Hanno Bock. The Problem with OCSP Stapling and Must Staple and why Certificate Revocation is still broken, 2017. https://blog.hboeck.de/archives/886-The-Problem-with-OCSP-Stapling-and-Must-Staple-and-why-Certificate-Revocation-is-still-broken.html.
[11] Taejoong Chung, Jay Lok, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, John Rula, Nick Sullivan, and Christo Wilson. Is the Web Ready for OCSP Must-Staple? In *Proceedings of the Internet Measurement Conference 2018*, pages 105–118. ACM, 2018.
[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, 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.
[15] 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.
[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] 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.
[19] 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.
[20] 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.
[21] 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, 2018.
[22] Adam Langley. Transport Layer Security (TLS) Snap Start. Technical report, IETF Internet Draft, June 2010.
[23] Adam Langley. Revocation checking and Chrome's CRL, 2012. https://www.imperialviolet.org/2012/02/05/crlsets.html.
[24] 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.
[25] 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 Web's PKI. In *Proceedings of the Internet Measurement Conference (IMC)*, pages 183–196. ACM, 2015.
[26] Moxie Marlinspike. SSL and the Future of Authenticity. *Black Hat USA*, 6, 2011.
[27] 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.
[28] Mark O'Neill, 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.
[29] Vern Paxson. Bro: a System for Detecting Network Intruders in Real-time. *Computer networks*, 31(23-24):2435–2463, 1999.
[30] Matthew Prince. The Hidden Costs of Heartbleed, 2017. https://blog.cloudflare.com/the-hard-costs-of-heartbleed/.
[31] 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.
[32] 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.
[33] 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.
[34] 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.
[35] Emin Topalovic, Brennan Saeta, Lin-Shung Huang, Collin Jackson, and Dan Boneh. Towards Short-Lived Certificates. *Web 2.0 Security and Privacy*, 2012.
[36] 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.

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 [25], 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 [14] as *Currently Trusted* (non-expired, trusted by Apple's, Microsoft's, or Mozilla NSS's root store) on March 21, 2018 and June 17, 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 previous revocation measurements studies [25], we started with any certificate seen in

	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 3: The ordering and reasons we removed certificates from the March 21, 2018 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 4: The reported revocation reason of revoked certificates from the March 21, 2018 dataset

many previous Internet scans or Certificate Transparency Logs[16] 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 typically not be advertised [25]. We used Censys.io [14], a search engine created to allow researchers to access data from daily Internet scans, as our initial dataset. We obtained permission to access and read from their database through Google BigQuery.[17]

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). Additionally, we cleaned our dataset by removing duplicate, expired[18], private[19], and invalid certificates[20] (see Table 3 for March 2018 dataset cleaning). After cleaning the dataset, 84.2 Million certificates on March 21, 2018 and 183.9 Million on June 17, 2018 remained.

A.2 Performing Revocation Checking

As done in previous revocation collection efforts [24, 25], we separated certificates with CRL endpoints (March 21, 2018: 29.9M, 33.6%; June 17, 2018: 55.7M, 30.3%) from those with only OCSP endpoints (March 21, 2018: 59.0M, 66.3%; June 17, 2018: 128.2M; 69.7%). The remaining certificates (March 21, 2018: 475; June 17, 2018: unavailable) did not have any revocation endpoint and were unrevokable.[21] 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.[22] We limited our request rate to 10 requests a second for all other OCSP endpoints.

	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 5: The reported revocation status of certificates from the March 21, 2018 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 6: The reported revocation status of certificates from the June 17, 2018 dataset

A.3 Data Collection Results

The results of our scans are summarized in Table 5 and 6. Overall, we found a total of 1.29% (1.08 Million) and 0.95% (1.75 Million) revoked certificates in our datasets. This percentage is very similar to the percentage of revoked certificates seen in 2014 [25] before the discovery of the Heartbleed vulnerability. Following the discovery of Heartbleed, the percentage of revoked certificates quickly rose to around 8% [25].

In our March 2018 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 4). The vast majority (457,875; 87.89%) of revocations were reported as *Cessation*

[16] https://www.certificate-transparency.org/
[17] https://cloud.google.com/bigquery/
[18] These expired by March 21m 2018 or June 17, 2018 repectively.
[19] Private certificates are those using an LDAP endpoint or are otherwise inaccessible. Most of these certificates returned an unauthorized status code on request.
[20] Each invalid certificate had at least one Zlint error [21].
[21] Of the 475 unrevokable certificates, all but 2 certificates were a root certificate, an intermediate certificate, or an OCSP Signing certificate.
[22] Let's Encrypt, Symantec, DigiCert

Of Operation. Only 3,403 (0.65%) revocations were due to key compromise, and 11 (0.002%) revocations were due to 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%) of these certificates in our March dataset. In June, we found 14.1 Million revocations included in the CRLs (601 MB) with only 8.0% that were certificates in our June dataset. We noticed a similar imbalance in 2017 [24] where we found 12.7 Million certificate revocations with only a small percentage that were previously seen in Internet scans and were not expired.[23] While we cannot be sure who uses these certificates, we suspect they are either privately used certificates or certificates that should have been previously removed from a CRL, i.e., expired. If we included only revoked but otherwise valid certificates with publicly accessible CRL endpoints seen in our dataset (non-expired, CA-trusted, seen in previous Internet scan) in CRLs, we estimate the cost of downloading all accessible CRLs would be reduced to only 38 MB and 48 MB receptively. However, because CAs have signed these unseen certificates and browsers trust them, it would be irresponsible for CAs to naively remove these from their CRLs.

[23]The authors communicated this information through email.