An End-to-End Measurement of Certificate Revocation in the Web's PKI

Yabing Liu* Will Tome* Liang Zhang* David Choffnes* Dave Levin†
Bruce Maggs‡ Alan Mislove* Aaron Schulman§ Christo Wilson*

*Northeastern University †University of Maryland
‡Duke University and Akamai Technologies §Stanford University

ABSTRACT

Critical to the security of any public key infrastructure (PKI) is the ability to revoke previously issued certificates. While the overall SSL ecosystem is well-studied, the frequency with which certificates are revoked and the circumstances under which clients (e.g., browsers) check whether certificates are revoked are still not well-understood.

In this paper, we take a close look at certificate revocations in the Web's PKI. Using 74 full IPv4 HTTPS scans, we find that a surprisingly large fraction (8%) of the certificates served have been revoked, and that obtaining certificate revocation information can often be expensive in terms of latency and bandwidth for clients. We then study the revocation checking behavior of 30 different combinations of web browsers and operating systems; we find that browsers often do not bother to check whether certificates are revoked (including mobile browsers, which uniformly *never* check). We also examine the CRLSet infrastructure built into Google Chrome for disseminating revocations; we find that CRLSet only covers 0.35% of all revocations. Overall, our results paint a bleak picture of the ability to effectively revoke certificates today.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations; E.3 [**Data Encryption**]: Public Key Cryptosystems, Standards

Keywords

SSL; TLS; PKI; HTTPS; X.509; Certificates; Revocation; Extended validation, Web browsers, CRLSet

1. INTRODUCTION

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols secure a significant fraction of Internet traffic today. Coupled with a Public Key Infrastructure (PKI), SSL[1] provides authentication via certificate chains and private communication via encryption.

Critical to the security of any PKI is the ability to *revoke* a previously-issued certificate, that is, to invalidate it before it expires. If the private key corresponding to a given certificate were compromised, the attacker could impersonate the certificate owner or eavesdrop on encrypted traffic sent to the certificate owner until the certificate's expiry date. Even more harrowing, if an *intermediate* certificate were compromised, an attacker could issue valid certificates for *any* domain (and, unfortunately, such attacks have occurred in the past [5,31,48]). The only recourse a certificate owner has is to request that their Certificate Authority (CA) create and disseminate revocations: signed attestations that the certificate should no longer be considered valid.

Successfully revoking a certificate requires action not only from website administrators and CAs; *clients*, especially web browsers, must request *and respect* revocations. If any of these principals fail to meet their responsibilities—if administrators fail to request revocations, CAs fail to distribute them, or browsers fail to fetch them—users risk being susceptible to impersonation attacks.

And yet, surprisingly, security practitioners have developed a strikingly negative view towards certificate revocation, even going so far as to doubt the fundamental usefulness of revocations [27, 28, 32, 42, 46]. These objections are largely rooted in the costs that the various parties incur in supporting revocations and the tradeoffs that clients (i.e., web browsers) face when they are unable to obtain the revocation status of certificates. While the incentives appear at odds with proper security, the extent to which they have led to inaction is largely unknown. This uncertainty leads to a chicken-and-egg problem: administrators argue that they need not revoke because clients rarely check, while clients argue they need not check because administrators rarely revoke. Without a concrete understanding of revocation in today's PKI, it is unclear how to break this logjam and improve the PKI's handling of certificate revocation.

In this paper, we empirically evaluate the extent to which all three parties involved—website administrators, CAs, and browsers—meet their revocation responsibilities. Our investigation extends prior work in the measurement of the SSL ecosystem; to date, most studies [7, 14] have focused on live certificates (proper key lengths, trust aggregation among CAs, etc), and there have been few studies of the critical revocation mechanisms available today. In particular, we

IMC'15, October 28–30, 2015, Tokyo, Japan.

ACM 978-1-4503-3848-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2815675.2815685.

[1] TLS is the successor of SSL, but both use the same certificates. We refer to "SSL certificates," but our findings apply equally to both.

provide the first end-to-end evaluation of the Web's certificate revocation ecosystem, with the following contributions:

First, we study website administrators' revocation behavior using 74 separate, full IPv4 port 443 scans between October 2013 and March 2015, collected by Rapid7 [40]. We select all valid certificates from these scans, and then check their revocation status every day, starting in October 2014. Overall, we find that a surprisingly large fraction ($>8\%$) of these certificates are revoked, and that almost 1% of certificates that continue to be advertised are actually revoked. Additionally, we observe that OCSP Stapling, which addresses many of the difficulties of obtaining revocation information, is not widely deployed: only 3% of certificates are served by hosts supporting OCSP Stapling.

Second, we examine the CA's role in distributing certificate revocation information. We find that the most well-established distribution method, CRLs, can impose significant bandwidth and latency overhead on clients: the median certificate has a CRL of 51 KB, and some certificates have CRLs up to 76 MB in size. At the same time, we observe that checking certificate status via OCSP is significantly cheaper but still requires the client to delay accepting the connection until the OCSP responder can be contacted.

Third, we examine client-side revocation checking behavior, since clients are responsible for checking certificates' revocation status. We develop a test suite for web browsers that includes 244 distinct certificate configurations. We deploy this test suite to the most recent versions of all major browsers on both mobile and desktop devices. Overall, we find that the fraction of times that revocation information is actually checked is surprisingly low: Firefox *only* checks leaf and EV certificates for revocations, and *only* if the certificates have OCSP responders; all browsers except for Internet Explorer assume that a leaf certificate is valid if revocation information cannot be obtained; and not a single mobile browser checks revocation information for *any* certificates. In the process, we found three bugs in revocation checking behavior and filed bug reports with browser vendors.

Fourth, we examine the revocation checking policy for Chrome in more detail, as Chrome also uses a Google-curated subset of all CRLs called the *CRLSet*. Essentially, Google pre-selects a subset of all revoked certificates and pushes this information to Chrome browsers. Unfortunately, our analysis shows that the CRLSet contains only 0.35% of revoked certificates.

Taken together, our findings paint a bleak picture of certificate revocation on the Internet today. While a significant fraction of certificates *are* revoked, many clients will never receive these revocations and will obliviously accept the certificates as valid. Fortunately, our findings also point towards potential improvements that could be implemented in the very near term, including an improved CRLSet construction based on Bloom Filters (§7).

Our analysis relied on both public sources of data and those we collected ourselves. We make our data and our browser "test suite" available to the research community at

`http://www.sslresearch.org`

2. BACKGROUND

SSL and TLS provide confidentiality and integrity for the vast majority of secure online communication. When combined with a PKI, they also allow parties to authenticate the identity of the other communicating party. In this section, we provide a brief background of SSL/TLS and PKIs relevant to our study, and detail the protocols that exist to support certificate revocation. We refer the reader to the book by Gutmann [20] for a more in-depth treatment of these topics.

2.1 Certificates

A certificate is an attestation signed by an *issuer* that binds a *subject* to a *public key*. In the web's PKI, the issuers are Certificate Authorities (CAs) such as Verisign or GoDaddy, who possess their own certificates binding their identities to their public keys. These CA certificates are, in turn, signed by other CAs, and so on, terminating at a small set of self-signed *root certificates*. Clients are assumed to obtain these trusted root certificates out-of-band (e.g., most browsers and operating systems ship with a set of root certificates).

The PKI does not include a widely accepted mechanism for delegating partial authority to issue certificates to CAs.[2] As a result, *any* CA can issue a certificate for *any* domain.

To verify a *leaf certificate* (i.e., a certificate that is not permitted to sign other certificates, used by most websites), a client needs to obtain a logical *chain* of certificates, leading from a root certificate through zero or more *intermediate* (*CA*) certificates, to the leaf certificate.[3] The private keys for each certificate are used to sign the certificate at the next level, with the exception of the root (which is signed by its own private key). To verify the leaf's identity, the client thus needs to verify that each certificate along this chain has a correct signature, is still *fresh* (certificates have a well-defined period during which they can be accepted), and has not been revoked.

The most common format for certificates on the Internet today is X.509 [11], which contains an ASN.1 [34] encoding of the certificate along with its signature. The X.509 format allows for additional information to be supplied, including a certificate's serial number (unique for the issuer), the certificate's validity period, and where to check whether the certificate has been revoked.

Validating Identities. Ultimately, CAs exist to bind identities to public keys, and are therefore responsible for validating their customers' identities. Typically, a CA challenges an applicant to prove ownership of the domain for which they are applying, for instance by requiring the applicant to post CA-chosen data at a URL in that domain; this process is called *Domain Validation* (DV). To provide greater assurance to clients that certificates were issued properly, *Extended Validation* (EV) certificates are a mechanism for CAs to assert that the identity verification process has followed a set of established criteria.

EV certificates follow the same format as X.509 certificates, but simply contain an additional `Policy Identifier` that indicates that a more thorough verification was performed. Many browsers display EV certificates differently (typically with a green box in the address bar). As we will see later in the paper, browsers often also perform a different set of revocation checks on EV certificates.

[2]The X.509 specification includes the `Name Constraints` extension for exactly this, but it is rarely used and few clients support it.

[3]Certificates can also be cross-signed [22] by other issuers for redundancy, providing multiple valid chains for a given certificate.

2.2 Revocation

Most CAs allow entities who have been issued certificates to later request to have them *revoked*. Revocations are represented by attestations signed by the same CAs who issued the corresponding certificates. CAs are also responsible for disseminating the revocation status for all certificates they have issued.

When a client establishes an SSL connection, the server presents a chain of certificates as part of the SSL handshake. In addition to verifying the chain itself, the correct behavior of the client is to ensure that *all* certificates in the chain[4] have not been revoked before continuing with the connection. Each certificate typically includes information about how and where to check for revocation information (i.e., a protocol and a URL).

There are two predominant methods for disseminating revocations: Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP):

CRLs. CRLs are the most well-established means of disseminating revocations. A CRL is simply an ASN.1-encoded file that contains a list of (serial number, revocation timestamp, revocation reason) tuples, all of which are collectively signed by the CA. Thus, checking the revocation status of a certificate using a CRL involves downloading the CRL file specified in the certificate and checking whether the certificate's serial number is listed in the CRL.

Similar to X.509 certificates, each CRL contains information specifying the range of time that it is good for; CAs are therefore required to re-issue CRLs periodically even if no additional certificates have been revoked. Clients can therefore cache CRLs, but must download an updated CRL once they expire.

CRLs have been criticized [15] for being an inefficient means of disseminating revocation information: clients must download information about *all* of the CA's revoked certificates even though they are typically only interested in the validity status of a single certificate. Thus, CRLs impose on CAs and clients a burden of excessive communication overhead.

OCSP. OCSP was designed to reduce the overhead of CRLs by allowing clients to query the CA for the revocation status of a *single* certificate. OCSP allows a client to generate an HTTP request for the status of a given certificate's serial number. The URL that clients should query is stated in the certificate. The CA returns a signed response that includes the certificate's current status (`good`, `revoked`, or `unknown`[5]), as well as a validity period, meaning clients can cache the response, typically on the order of days (longer than most CRLs).

OCSP addresses many of the inefficiencies of CRLs, but still requires request(s) to the CA(s) before a certificate can be trusted. Additionally, using OCSP reveals information about the users' browsing behavior to CAs, a potential privacy risk. Thus, checking the revocation status of certificates via OCSP still imposes a significant burden on clients.

[4]The client need not check the revocation status of the root certificate, as root certificates can only be "revoked" by removing them from the list of trusted certificates on clients.

[5]The OCSP specification [44] states that the `unknown` response "indicates that the status could not be determined by this responder". While the correct client behavior when receiving an `unknown` response is the subject of some debate, it is clear that the response does not indicate that certificate in question should be trusted.

OCSP Stapling. OCSP Stapling is an SSL/TLS extension that allows a server to cache OCSP responses and send them to clients as part of the SSL handshake. Thus, when communicating with a server that supports OCSP Stapling, a client receives *both* the server's certificate *and* an OCSP statement of the certificate's validity. The client can verify the OCSP statement and therefore be assured that the certificate is not revoked. OCSP Stapling removes most of the latency penalty associated with verifying a certificate's revocation status.

Unfortunately, OCSP Stapling does not entirely remove the latency penalty for clients, as OCSP Stapling only includes the OCSP response for the *leaf* certificate (the protocol does not allow the server to include cached OCSP responses for intermediate certificates). A recently proposed extension to OCSP Stapling [37] addresses this limitation by allowing the server to include stapled OCSP responses for intermediates and the leaf certificate, but has yet to see wide adoption.

2.3 Certificate Validation

Despite its critical importance for securing the integrity of the PKI, the X.509 specification [11] is somewhat vague when discussing revocation checking during certificate validation. While it is clear that a client should not trust a revoked certificate that appears in a CRL or OCSP response, there is an active discussion [20, 28] regarding what to do when a client is unable to access the revocation status of a certificate (e.g., if the browser cannot resolve the domain name of the CRL server, if the OCSP server is down, etc). The X.509 specification states that, in this case, the certificate should be given the status `undetermined` [11], but does not specify how this should be interpreted.

For maximum security, the client should not trust the certificate chain if any of its revocation information is unavailable; after all, the client cannot be certain that one of the certificates has not been revoked. But from a user's perspective, this would look like the browser's inability to load a page that an incorrect browser would successfully load. Thus, browser developers often *soft-fail* by deciding to trust certificates when revocation information is unavailable, so as not to rest their perceived reliability on the shoulders of a disparate set of third-party CAs.

Soft-failing when revocation information is unavailable may at first appear to be an innocuous trade-off for usability, but in practice it has surprisingly extensive implications on the security of the PKI. Any attacker who can block a victim's access to specific domains (e.g., an attacker on the same wireless network) could leverage soft-failures to effectively turn off the victim's revocation checking. As a result, some browser maintainers have concluded that requesting revocation information in direct response to visiting a website does not actually increase security [28], and have advocated instead for pushing incomplete subsets of revocation information to clients (§7).

To ensure user security, however, there is simply no replacement for maintaining as complete and up-to-date revocation information as possible. And when it is not possible, it is our opinion that hard-failing would better inform users of the potential security risks, and may apply useful customer pressure on CAs with unreliable services.[6] Thus, we

[6]Browser maintainers have also claimed that hard-failing browsers would also cause CRL servers and OCSP responders to become single

maintain that revocation is a critical component of a PKI, and in the remainder of this paper, we analyze to what extent it is supported by administrators, CAs, and clients.

3. DATA COLLECTION

In this section, we describe the data we use to understand certificate revocation behavior, and define several terms that we use in our analysis.

3.1 SSL Certificates

Obtaining representative statistics on certificates that are revoked is not entirely straightforward, as revoked certificates (by definition) should no longer be advertised. Thus, simply measuring the number of entries in CRLs is sufficient to count revocations, but does not reveal *which* certificates were revoked (CRLs only contain the certificates' serial numbers). As a result, it is necessary to collect data on advertised certificates over time, and then periodically check to see which of the certificates have been revoked. However, obtaining a representative sample of advertised certificates is also difficult; relying on data gathered from passive traces [16, 23] is likely to have good coverage of popular certificates but may miss unpopular ones.

We obtain our collection of SSL certificates from (roughly) weekly scans of port 443 over the entire IPv4 address space, made available by Rapid7 [40]. In this paper, we use 74 scans conducted between October 30, 2013 and March 30, 2015. Overall, we observe 38,514,130 unique SSL certificates.

Many of the certificates that we find in the scans are invalid (e.g., self-signed certificates on WiFi routers). Thus, we pre-process the dataset by verifying all observed certificates. We do so by first building the set of all *intermediate* certificates that can be verified relative to the roots.[7] This is an iterative process (as certain intermediates can only be verified once other intermediates are verified); in the end, we discover 1,946 intermediate certificates, which we refer to as the *Intermediate Set*.

We then proceed to verify all leaf certificates using this set of intermediate and root certificates. We configure OpenSSL to ignore certificate date errors (as our scans cover over 1.5 years), but to fail on any other error (except revocation, which we address later). This allows us to only keep certificates that were valid at some point in time. We find a total of 5,067,476 such leaf certificates; this *Leaf Set* is the set of certificates that we examine throughout the remainder of the paper. Of these, we observe that 2,291,511 (45.2%) were still being advertised in the latest port 443 scan.

3.2 Obtaining Revocation Information

We collect our data on certificate revocations using the revocation information present in the certificates. In the Leaf Set, we observe that 99.9% of the certificates list a potentially reachable[8] CRL distribution point, and 95.0% of certificates list a potentially reachable OCSP responder. Interestingly, we find 4,384 (0.09%) Leaf Set certificates that have *neither* a CRL distribution point nor an OCSP responder. These represent certificates that can *never* be revoked.

points of failure and attractive DDoS targets. However, CRLs and OCSP responses are cacheable, and existing techniques for distributing static content (e.g., CDNs) can be applied to them as well.

[7] We use the root store in OS X 10.9.2 [35] as our set of trusted roots; this includes 222 unique root certificates.

[8] We only consider `http[s]://` CRL URLs, and ignore distribution points in private networks such as `ldap://` and `file://` URLs.

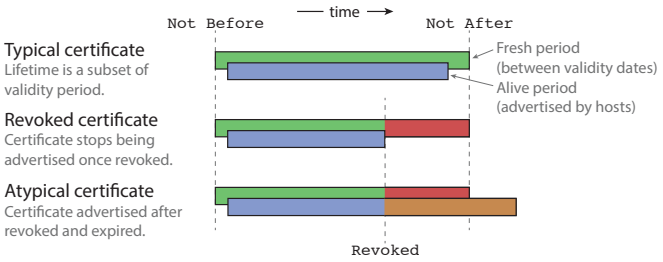

Figure 1: Diagram of events in a certificate's lifetime. Two different timelines exist for any certificate: when it is *fresh* (between validity dates) and when it is *alive* (advertised by servers). Certificates may also be revoked during the time they are fresh.

In the Intermediate Set, we observe that 98.9% of the certificates list a potentially reachable CRL distribution point, while 48.5% list a potentially reachable OCSP responder. Similar to the Leaf Set, we find 18 (0.92%) intermediate certificates that have no CRL distribution point or OCSP responder.[9] Being unable to revoke a CA certificate is particularly worrisome, as possessing a CA certificate's private key allows one to generate certificates for *any* Internet domain (and private keys for CA certificates have be inappropriately given out multiple times in the past [5, 31, 48]).

CRLs. For the certificates that include a CRL distribution point, we use this CRL to obtain revocation information for the certificate. We observe a total of 2,800 unique CRLs, and we configure a crawler to download each of these CRLs once per day between October 2, 2014 and March 31, 2015.

OCSP. We observe a total of 499 unique OCSP responders across all certificates. However, querying each of these responders to check the status of each certificate is prohibitively expensive, and also unnecessary in the case where a CRL distribution point is also provided. Thus, we only query the OCSP responders for the 642 certificates that *only* have an OCSP responder provided (i.e., no CRL distribution point). This data was collected on March 31, 2015.

3.3 Definitions

Throughout this study, we will consider a variety of events in the lifetime of an SSL certificate. For a given certificate, we are concerned with two interrelated timelines (Figure 1 demonstrates a few possible configurations of these timelines):

Fresh. We define a certificate's *fresh period* to be the time between its `Not Valid Before` date and its `Not Valid After` date. This is the period of time in which a client will potentially accept the certificate.

Lifetime. We consider a certificate to be *alive* during the time from when we first saw the certificate advertised (its birth) to when we last saw it advertised (its death). Typically, the lifetime is a strict subset of the certificate's fresh period, but we do observe many instances where expired certificates are still advertised.

Note that neither of these definitions account for revocations: we consider a certificate in its fresh period to still be

[9] This figure excludes root certificates, as these certificates have no CRL distribution points or OCSP responders by design.

fresh, and an advertised certificate to be alive, regardless of whether or not it has been revoked. We do so because clients that do not check revocation information will still accept revoked fresh certificates.

4. WEBSITE ADMIN BEHAVIOR

The revocation process begins with the website administrator making a request to its CA, and offering a reason for the revocation. We begin our analysis at the source by analyzing the frequency and reasons for revocations.

4.1 Frequency of Revocations

First, we examine the characteristics of individual certificates. Figure 2 shows the fraction of fresh and alive certificates that are revoked in our data set between January 2014 and March 2015. We first observe that a surprisingly large fraction of all fresh certificates are actually revoked: as of this writing, over 8% of them are revoked. The majority of these revocations are caused by the Heartbleed vulnerability [52]; this shows up in the graph as the "spike" starting in May 2014.[10] However, even before this vulnerability, over 1% of fresh certificates were revoked, indicating that a non-trivial fraction of SSL certificates are revoked even in steady-state.

Second, we observe that the fraction of alive certificates that are revoked is much smaller—less than 1% of all alive certificates—but still non-zero. Certificates in this category are surprising, as it suggests that the site administrator went to the effort to revoke the certificate, but failed to update (all of) their machines with the new certificate. Examples of such certificates include `https://west-secvpn.apple.com`, `https://vpn.trade.gov`, and `https://gamespace.adobe.com` (the last of which is both expired *and* revoked).

Third, when considering only EV certificates, we observe largely similar trends: Over 6% of fresh EV certificates are currently revoked, and over 0.5% of alive EV certificates are revoked. EV certificates have much stricter entity validation requirements by CAs, so it is surprising to see that a *higher* fraction of EV certificates are revoked but still advertised.

Compared to the revocation behavior shortly after Heartbleed [52], we find that certificate owners quickly returned to pre-Heartbleed behaviors, despite the fact that there were still many vulnerable certificates that needed to be revoked. Moreover, we find that revocation alone may not be enough, as there are many revoked certificates still being advertised. Whether they have any impact on client security depends on whether clients are downloading revocation information, which we study in §6.

4.2 Reasons for Revocation

We have observed that a significant fraction of all fresh certificates are revoked, but a key question remains: *Why* are these certificates revoked? Unfortunately, the reasons for revocation are typically difficult to measure; only the certificate owners (or, potentially, the CA) knows why a certificate is being revoked, and many administrators are likely to consider this information to be private or sensitive.

Zhang et al. [52] used CRL *reason codes*—which range from "Unspecified" to "Key Compromised" to "Privilege Withdrawn" [11]—to attempt to gain some insight, but found reason codes not to be particularly informative. We repeated their methodology with our dataset, extracting the reason code for all revocations. Even on our larger data set, we find a similar distribution of reason codes: in particular, the vast majority of revocations actually include no reason code. Thus, CRL reason codes provide some information (and as we will see in §7, they form the basis of Google's decision to include the revocation in CRLSets), but should likely be viewed with caution.

[10] Previous work demonstrated that, a full month after Heartbleed was announced, only about 13% of site owners that *should have* revoked did revoke [52]. Thus, an even larger fraction of fresh certificates *should be* revoked, but are not.

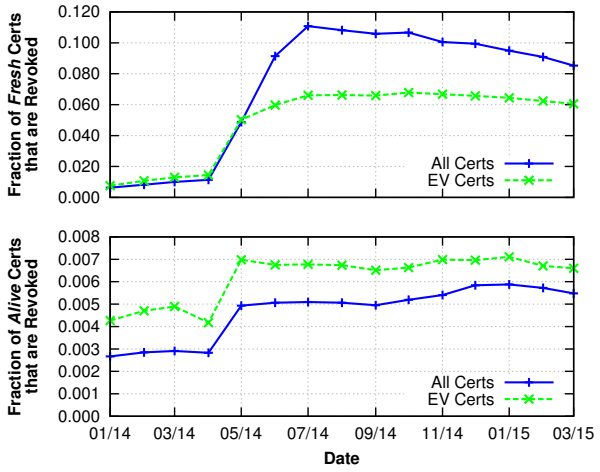

Figure 2: Fraction of fresh (top) and alive (bottom) certificates over time that are revoked (for all certificates and EV-only certificates). The large "spike" in April 2014 is due to certificate revocations caused by the Heartbleed vulnerability. We observe that today, over 8% of all fresh certificates are revoked, and more surprisingly, over 0.6% of *alive* certificates are revoked.

4.3 OCSP Stapling

Recall from §2.2 that OCSP Stapling was developed to address the limitations of CRLs and OCSP. One would thus expect that OCSP Stapling has been embraced at web servers. However, OCSP Stapling requires the website administrators to enable it on their web server (unlike CRLs and OCSP, which only involve the CA), so it is unclear how often website administrators do so.

To determine what fraction of certificates are hosted on servers that support OCSP Stapling, we use the IPv4 TLS Handshake scans conducted by the University of Michigan [49]. Each scan contains details about the port 443 SSL handshake with every IPv4 server, including the support for various TLS extensions. We examine the scan of March 28, 2015, and look for servers that were advertising certificates in the Leaf Set.

Some web servers that support OCSP Stapling (e.g., Nginx) may not include a staple in the response if they do not have a valid staple cached (in this case, they will then attempt to fetch a fresh staple). Thus, the scan data from above may underestimate the fraction of servers that actually support OCSP Stapling. To measure this effect, we chose 20,000 random servers and connected to them repeatedly (sleeping 3 seconds between connections) to see if we ever received a staple, and present the results in Figure 3. Overall, the results suggest that using only a single connection is likely to underestimate the true fraction of OCSP

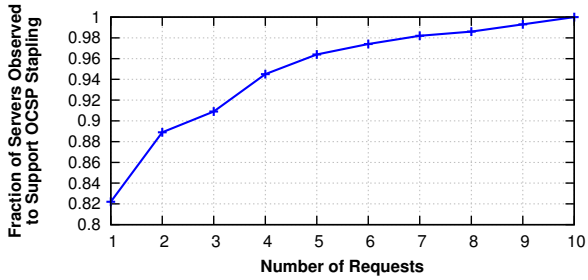

Figure 3: Fraction of servers observed to support OCSP Stapling as multiple requests are made; certain servers will only provide a staple if a fresh one is cached (note that the y-axis does not start at 0). Using only a single request is likely to underestimate the fraction of OCSP Stapling support by approximately 18%.

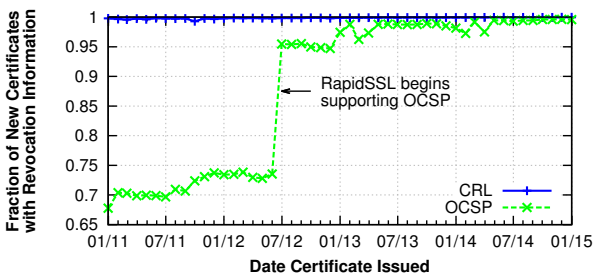

Figure 4: Fraction of new certificates that include CRL or OCSP revocation information over time (note that the y-axis does not start at 0). We observe that, today, almost all new certificates contain both types of revocation information.

Stapling support by about 18%; this effect should be taken into account when interpreting our results below.

Looking at the TLS Handshake scan data, we observe 12,978,883 servers advertising fresh Leaf Set certificates (this is larger than the number of Leaf Set certificates, as one certificate could be advertised by many servers), of which only 337,856 (2.60%) support OCSP stapling. However, this only tells us the fraction of servers that support OCSP Stapling, but not the fraction of *certificates* that we see it supported for. Of the 2,298,778 fresh certificates advertised in this scan, we observe that only 119,519 (5.19%) are served by at least one server that supports OCSP Stapling, and only 70,996 (3.09%) are served by servers that *all* do so.

If we narrow our focus to only EV certificates, we find that of the 83,974 fresh EV certificates advertised during the scan, only 2,644 (3.15%) are advertised by at least one server that supports OCSP Stapling, and only 1,640 (1.95%) are advertised by servers that *all* support OCSP Stapling. This is a smaller fraction than all certificates, and may be caused by many large hosting providers (commonly used by owners of EV certificates) not supporting OCSP Stapling.

Thus, while OCSP Stapling is supported by all major web server implementations, OCSP Stapling has still seen very little adoption. Even taking into account the underestimation of OCSP Stapling support from a single-connection scan, only about 6–7% of certificates are served by at least one server that supports OCSP Stapling. Our results stand in contrast to scans that focus only on popular web sites (e.g., Qualys' SSL Pulse [38] reports ∼20% OCSP Stapling support), suggesting that less popular sites are significantly less likely to support OCSP Stapling.

4.4 Summary

Our results show that a considerable number of certificates are advertised even long after they are revoked. This should have no impact on security if clients are vigilant in downloading revocation information, but otherwise, could make impersonation attacks possible. Moreover, our results suggest that website administrators rarely enable OCSP Stapling, the most efficient and client-friendly protocol for distributing revocations. Next, we investigate the behavior of the distributors of revocation information: the CAs.

5. CA BEHAVIOR

Upon receiving a request to revoke a certificate, CAs are responsible for maintaining a highly available service for disseminating revocation information. In this section, we turn our study towards the CAs by investigating the characteristics of the distribution mechanisms they make available.

5.1 Availability of Revocation Information

We briefly explore how the inclusion of revocation information in certificates has evolved in Figure 4, where we plot the fraction of new certificates that contain potentially reachable CRL distribution points or OCSP responders each month. We observe that CRLs have been included in almost all certificates issued over the past four years, but that OCSP was slower to be adopted (unsurprisingly, as it was standardized much later than CRLs). Moreover, we observe a "spike" in the prevalence of OCSP information in July 2012, corresponding to the adoption of OCSP by RapidSSL (a low-cost CA owned by GeoTrust).

These encouraging results show that, today, both forms of revocation are included in nearly all certificates. Unfortunately, both of these schemes have their shortcomings: CRLs impose bandwidth burdens (which we investigate later in this section), and having to query an OCSP responder imposes slower page-load times for browsers. As a result, there has been push-back from both the CA and browser communities. Unfortunately, as we demonstrated in the previous section, OCSP Stapling is still only deployed on a small fraction of web servers.

5.2 Size of Revocation Information

One of the common criticisms of CRLs is the network traffic required to download them. We now examine the sizes of CRLs, and how this has evolved over time.

Recall that CRLs contain one entry for each certificate that is revoked. Thus, the size of the CRL (in bytes) is expected to correlate with the number of entries. We plot this correlation as a scatterplot in Figure 5, and observe a strong linear relationship (except for CRLs with just a few entries, which have some fixed overhead).[11] On average, each entry is 38 bytes.

Figure 5 suggests that most CRLs are small. We explore this trend further in Figure 6, which presents the CDF of CRL sizes as the *Raw* line. We immediately observe that half of all CRLs are under 900 B. However, this statistic is deceiving: if you select a certificate at random from the Leaf

[11] The variance between CRL sizes at the same number of entries is largely due to differences in serial number assignment policies for different CAs; some CAs use serial numbers of up to 49 decimal digits, which results in larger CRL file sizes.

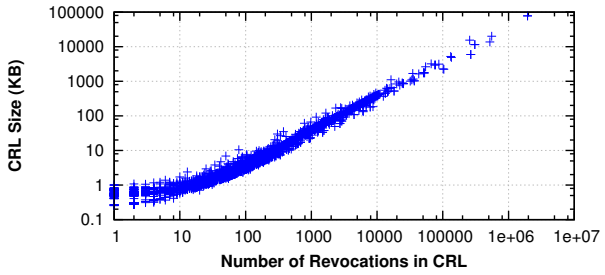

Figure 5: Scatterplot of the number of entries in CRLs versus CRL file size, for all 2,800 CRLs we crawled. As expected, a linear correlation is observed.

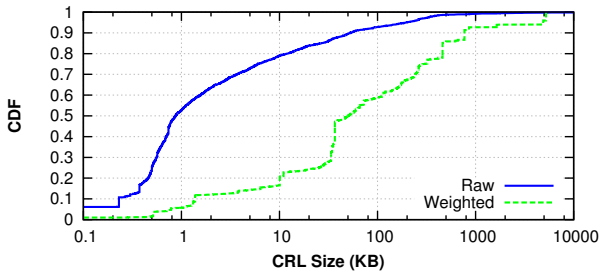

Figure 6: Cumulative distribution of the sizes of CRLs (shown under the *raw* line). Most CRLs are small (median size is less than 1 KB), but most CRLs cover few certificates. The *weighted* distribution shows the CRL size distribution across certificates; in this case, the mean certificate has a CRL size of 51 KB.

Set, it is unlikely to point to a tiny CRL, since the tiny CRLs cover very few certificates. To adjust for this issue, we plot the CRL size *per certificate* as the *Weighted* distribution in Figure 6.[12] The Weighted distribution tells a different story: the CRL size for the median certificate is 51 KB, and we observe CRLs ranging up to 76 MB.[13] Although one can argue that weighing CRLs is unfair (as clients can cache them), 95% of CRLs expire in less than 24 hours [52], reducing a client's ability to save bandwidth through caching.

One mechanism that CAs can use to reduce the size of CRLs is to maintain multiple CRLs [20], assigning only a subset of all issued certificates to each CRL. We explore the extent to which different CAs do this in Table 1, and observe that, in general, CAs use only a small number of CRLs. The CA that uses this technique to the largest extent is GoDaddy with 322 unique CRLs, but given the large number of GoDaddy revocations, their average certificate's CRL is still over 1MB in size. Moreover, the trend is not entirely consistent, as various CA policies will affect the resulting CRL size.[14]

Our findings confirm that fetching CRLs can be an expensive operation for clients, especially ones on bandwidth-constrained networks (e.g., mobile devices). Moreover, properly verifying a certificate requires the client to download the CRL before fully establishing the SSL connection; thus, for interactive applications like web browsers, downloading the CRL also comes with a latency penalty. The large size of CRLs is especially glaring when compared to the size of an OCSP query and response (which is typically less than 1 KB and often comes with a latency penalty under 250 ms [33]).

5.3 Summary

Overall, our results show that, while there are potentially high costs involved in disseminating revocation information, many CAs have not widely adopted smaller CRLs. Combined with the lack of wide OCSP Stapling adoption, we see that there are multiple opportunities to significantly lower the costs incurred by CAs and browsers when obtaining certificate revocation information.

[12]For certificates that have multiple CRLs, we pick the smallest.

[13]The 76 MB CRL is hosted by Apple at `http://crl.apple.com/wwdrca.crl` and contains over 2.6 million revoked certificates.

[14]One notable example in the table is StartCom, which has an outsized average CRL size relative to the number of revocations. This is due to a single CRL (`http://www.startssl.com/crt1-crl.crl`) that is over 22 MB and has over 290 K revocations. This CRL is for StartSSL "Free" certificates that, after a year of validity, *require* customers to revoke an expiring certificate (at a fee of $24.90) before a new certificate for the same domain will be reissued [45].

6. CLIENT BEHAVIOR

Recall that the burden of certificate validation is placed entirely on the client: it is up to the client to check the validity of the presented certificate chain and to obtain revocation information from CAs to make sure that none of the certificates have been revoked. We now explore the extent to which the most popular SSL clients—web browsers—do so. Because web browser developers care deeply about minimizing latency, and (as we observed above) checking certificate revocation information can take significant effort, web browser developers have a strong tension between minimizing latency and ensuring security.

6.1 Methodology

Our goal is to build a test harness that is able to determine whether a web browser chooses to check certificate revocation information for a variety of different kinds of certificates and chains. Ideally, we would like to use real certificates, but doing so would require obtaining access to a real intermediate certificate (an unlikely prospect). Instead, we generate our own root certificate and install it so that the web browser trusts it. This allows us to then generate and sign intermediate and leaf certificates as we wish.

We build a test suite that covers many combinations of chain length, protocols, etc. For each test, we generate a *unique* set of certificates (i.e., the intermediates and leaf certificates are not re-used across tests, in order to eliminate caching effects). Each intermediate contains a unique subject name, and each leaf contains a unique common name. Each test is served by a dedicated Nginx web server. When generating test cases, we consider four different dimensions:

CA	Unique CRLs	Certificates Total	Certificates Revoked	Avg. CRL size (KB)
GoDaddy	322	1,050,014	277,500	1,184.0
RapidSSL	5	626,774	2,153	34.5
Comodo	30	447,506	7,169	517.6
PositiveSSL	3	415,075	8,177	441.3
GeoTrust	27	335,380	3,081	12.9
Verisign	37	311,788	15,438	205.2
Thawte	32	278,563	4,446	25.4
GlobalSign	26	247,819	24,242	2,050.0
StartCom	17	236,776	1,752	240.5

Table 1: Number of CRLs, certificates (total and revoked), and the average CRL size per certificate for the largest CAs.

		Desktop Browsers										Mobile Browsers			
		Chrome 44			Firefox	Opera		Safari	IE			iOS	Andr. 4.1–5.1		IE
		OS X	Win.	Lin.	40	12.17	31.0	6–8	7–9	10	11	6–8	Stock	Chrome	8.0
CRL															
Int. 1	Revoked	EV	✓	EV	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	EV	✓	–	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
Int. 2+	Revoked	EV	EV	EV	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Leaf	Revoked	EV	EV	EV	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	A	✓	✗	✗	✗	✗
OCSP															
Int. 1	Revoked	EV	EV	EV	EV	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	L/W	✗	✓	✓	✓	✗	✗	✗	✗
Int. 2+	Revoked	EV	EV	EV	EV	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Leaf	Revoked	EV	EV	EV	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	A	✓	✗	✗	✗	✗
Reject `unknown` status		✗	✗	–	✓	✓	✗	✗	✗	✗	✗	–	–	–	–
Try CRL on failure		EV	EV	–	✗	✗	L/W	✓	✓	✓	✓	–	–	–	–
OCSP Stapling															
Request OCSP staple		✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	I	I	✗
Respect `revoked` staple		✗	✓	–	✓	✓	L/W	–	✓	✓	✓	–	–	–	–

Table 2: Browser test results, when intermediate (Int.) and leaf certificates are either revoked or have revocation information unavailable. ✓ means browser passes test in all cases; ✗ means browser fails test in all cases. Other keys include EV (browser passes only for EV certificates), L/W (browser passes only on Linux and Windows), A (browser pops up an alert), and I (browser requests OCSP staple but ignores the response).

Chain Length. Recall that a certificate chain starts at a root certificate, has 0 or more intermediates, and finally contains a leaf certificate. As certificate chains get longer (i.e., more intermediates), revocation checking becomes more expensive for the client. We therefore generate tests that contain between 0 and 3 intermediate certificates (between 2 and 5 total certificates, including the root and leaf), with separate tests where each element of the chain is revoked.

Revocation Protocol. Browsers may treat different revocation protocols differently, so we generate some certificate chains that only contain CRL information and others that only contain OCSP information (i.e., for each chain, all certificates contain *either* CRL distribution points or OCSP responders), as well as some chains that contain both. We also implement tests where the server is configured to perform OCSP Stapling if the client requests it.[15,16]

Extended Validation. Browsers may have special rules for EV certificates, so we generate additional tests where the leaf certificate contains an object identifier (OID) indicating that it is an EV certificate.[17]

Unavailable Revocation Information. Finally, if the browser is unable to fetch certificate revocation information (e.g., the OCSP responder or CRL server is down), the browser must choose whether or not to accept the certificate or to try and obtain revocation information using a different protocol. We generate additional tests where the revocation information is unavailable due to four different failures: the domain name of the revocation server does not exist, the revocation server returns a HTTP 404 error code, the revocation server does not respond, or the OCSP responder generates a response with status `unknown`.

Considering all possible combinations, the result is a suite of 244 different tests. When running tests, we also capture network traces to examine the SSL handshake and communication with revocation servers.

6.2 Test Suite Implementation

We implement our test suite with a script that, for each test, (1) generates a unique DNS name, (2) uses OpenSSL to generate a certificate chain and CRLs, (3) configures OpenSSL OCSP responders[18] (if OCSP is used), and (4) generates an Nginx server configuration for the test. Thus, each test has a dedicated DNS name, Nginx instance to serve the certificate chain, and set of OCSP responders.

We create a web page containing JavaScript that iteratively tries to fetch a file from each test's domain using XHRs.[19] If the browser determines that one of the certificates has been revoked, the file fails to load and our script gets an error callback. Thus, we can programmatically determine the tests where the browser checks the revocation status of different certificates.

6.3 Desktop Browsers

We begin by examining the behavior of popular desktop web browsers. For each browser/OS combination, we create

[15] In this case, the OCSP responder is firewalled from the client, so the only way the client can access the revocation information is via the stapled OCSP response.

[16] By default, Nginx refuses to staple an OCSP response that contains the status `revoked` or `unknown`. We modified Nginx to disable this behavior.

[17] We use OID 2.16.840.1.113733.1.7.23.6, an OID used by Verisign to distinguish EV certificates.

[18] By default, the OpenSSL OCSP responder implementation only supports `POST` OCSP queries; we modified it to also support the more commonly used `GET` queries.

[19] Our Nginx instances were configured to enable CORS.

a distinct virtual machine (VM), configured with the default settings. Our only VM configuration was to install our root certificate.[20] Unless otherwise stated, all tests were done on Ubuntu 14.04, Windows 8.1, and OS X 10.10.2. Overall, we tested 30 different combinations of OS and browser; our results are summarized in Table 2.

Chrome. Chrome uses the Mozilla NSS library [1] for SSL connection establishment, but it uses platform-specific libraries for certificate validation [3]. Because Chrome aggressively auto-updates (i.e., it checks for new versions every few days), we only test with the latest version of Chrome (44.0).

On OS X, Chrome treats EV and non-EV certificates differently. For non-EV certificates, Chrome does not check any revocation information. For EV certificates, Chrome checks all elements of the chain (using either CRLs or OCSP responders). Chrome does request OCSP staples, but Chrome *does not* respect OCSP staples with the status of `revoked` (instead, it attempts to contact the OCSP responder), and it incorrectly treats OCSP responses with the `unknown` status as trusted. When certificate revocation information is unavailable, Chrome does try to fetch revocation information from the CRL if the OCSP responder is not available. If revocation information is still unavailable, Chrome only rejects the chain if the first intermediate's CRL is unavailable *and* the leaf is an EV certificate. Otherwise, Chrome silently accepts the certificate.

On Windows, Chrome also treats EV and non-EV certificates differently. For non-EV certificates, Chrome *only* checks the first intermediate in the chain, and *only* if it only has a CRL listed (Chrome does not check any non-EV OCSP responders). For EV certificates, Chrome checks the revocation status of all elements of the chain, both OCSP and CRLs. Additionally, Chrome requests OCSP staples on Windows for all certificates. Chrome's behavior when certificate revocation information is unavailable or when the revocation status is `unknown` is the same as above, except that it rejects the chain for non-EV certificates as well.

On Linux, we were able to import our root certificate, but we were unable to get Chrome to use our root certificate to verify CRL or OCSP signatures (i.e., we observed that Chrome would request the same certificate revocation information multiple times). As a result, we rely on packet traces to measure Chrome's behavior, and are unable to test Chrome with unavailable revocation information or different OCSP response statuses. Overall, we found that Chrome's behavior is largely consistent with the other platforms: it only checks CRLs and OCSP responders for EV certificates, and does request OCSP staples.

In addition to the distribution mechanisms analyzed in this section, Chrome also has a built-in list of revoked certificates called the CRLSet; we explore CRLSets in more detail in §7.

Firefox. Firefox uses Mozilla's NSS library [1]. We tested the most recent version of Firefox (40.0) and found it to have the same behavior on all platforms.[21] Firefox does not check *any* CRLs. For OCSP, Firefox treats EV and non-EV certificates differently: for Non-EV certificates, Firefox only queries the OCSP responder of the leaf certificate. For EV certificates, Firefox checks all OCSP responders. Firefox correctly rejects OCSP responses with the `unknown` status, but does not try to fetch the CRL if the OCSP responder is not available. Instead, if the revocation information is not available, Firefox accepts the certificate. Firefox does request OCSP staples.

Opera. In mid-2013, Opera re-architected its browser: up through version 12.17, Opera used its own rendering engine; for all versions afterwards, Opera uses a fork of the Chromium project. These two versions have different behavior, and we examine them separately.

On Opera 12.17, the behavior is consistent across all OSes. Opera checks the revocation information for all certificates in the chain if they have CRLs listed, but only the leaf certificate if the certificates have OCSP responders listed. If certificate revocation information is not available, Opera accepts the certificate. Opera correctly rejects OCSP responses with the `unknown` status, and requests OCSP staples on all platforms.

On Opera 31.0, the behavior is largely consistent across all OSes: Opera checks all certificates in the chain when certificates have CRLs or OCSP responders listed. If the revocation information is not available for the first intermediate certificate (or the leaf certificate if no intermediates exist), Opera rejects the certificate on all platforms (if using CRLs) and only on Linux and Windows (if using OCSP); in all other cases, Opera accepts the certificate. Opera incorrectly treats OCSP responses with the `unknown` status as trusted. Opera requests OCSP staples on all platforms; however, on OS X (similar to Chrome) Opera does not respect `revoked` staples and attempts to contact the OCSP responder directly.

Safari. We examined the three most recent major versions of Safari (6.0–8.0) and found them all to have the same behavior.[22] Safari checks all certificates in the chain using CRLs or OCSP. Safari incorrectly treats OCSP responses with the `unknown` status as trusted. Safari does attempt to fetch the CRL if the OCSP responder is unavailable. If the revocation information is not available for the first intermediate certificate (or the leaf certificate, if there are no intermediates) and the certificate has a CRL, Safari rejects the certificate; in all other cases, it accepts the certificate. Safari *does not* request OCSP staples.

OS X offers system-wide settings via Keychain Access that allow users to change certificate revocation checking behavior. The default behavior, which we used in our above analysis, is referred to as "Best attempt". If a user selects "Require if certificate indicates", Safari does indeed reject all chains where any of the revocation information is unavailable.

Internet Explorer. We tested every major version of Internet Explorer (IE) between 7.0 (on Vista) to 11.0 (on Windows 10, 8.1, and 7). We found that the behavior of IE changed between version 9.0 and 10.0, and again with IE 11.0; we therefore consider them separately.

[20] We must create separate VMs for each browser/OS, as some browsers use system-wide libraries and daemons for checking revocation information, which can lead to caching effects at the OS level.

[21] Firefox contains a hardcoded list of root certificates that are allowed to issue EV certificates. We recompiled Firefox 40.0 after adding our root certificate to Firefox's list.

[22] We observed strange behavior with the most recent Safari/OS X update (8.0.5/10.0.3). With this update, Safari does not check *any* CRLs and only checks OCSP responders for EV certificates. The release notes do not mention any SSL policy changes, so we believe this behavior to be a bug; we have filed a bug report with Apple.

On IE 7.0–9.0, IE checks all certificates in the chain for both CRLs and OCSP responders. If the OCSP responder is unavailable, IE does download the CRL. If the revocation information is not available for the first certificate in the chain, IE rejects the certificate; in all other cases, IE accepts the certificate. IE incorrectly treats OCSP responses with the `unknown` status as trusted. IE does request OCSP staples.

On IE 10.0, IE has the same behavior except that when revocation information is not available for the *leaf* certificate, IE pops up a warning to the user and asks them if they wish to proceed.

On IE 11.0, IE has the same behavior except that when revocation information is not available for the *leaf* certificate, IE correctly rejects the certificate without popping up a warning.

In summary, we observe that *no* browser in its default configuration correctly checks all revocations and rejects certificates if revocation information is unavailable (as described in §2.2). In fact, many browsers do not bother to check revocation information at all. Overall, Internet Explorer performs the most checks, followed closely by Safari and newer versions of Opera.

We also found that many browsers do not correctly interpret `unknown` OCSP responses, not all browsers support OCSP Stapling, and many do not understand `revoked` staples. This is surprising, as OCSP Stapling is the most efficient way of obtaining certificate revocation information, and it imposes almost no extra latency on clients.

6.4 Mobile Browsers

Next, we study the certificate revocation checking behavior of browsers on mobile devices. To make the tests easier to conduct, we use the device simulators provided by the OS vendors where possible, rather than physical devices. These simulators run a full version of the mobile OS, allowing us to easily test multiple versions.

iOS. We begin by checking the behavior of Mobile Safari on iOS. We used the Xcode-based iOS simulator to test iOS 7 and 8, and use a physical iPhone 3GS device for iOS 6.[23] We observe that Mobile Safari does not check *any* certificate revocation information, nor does it request OCSP staples.

Android. We test the certificate revocation checking behavior of Android by using the Android Emulator. We test the "stock" Browser app and Chrome on the three most recent versions of Android: Lollipop (5.1), KitKat (4.4), and Jelly Bean (4.3). Similar to iOS, we find that neither application checks *any* certificate revocation information. However, by examining the network traffic, we found that both applications *do* request OCSP staples, but *do not* use them in certificate validation. Even when served an OCSP staple with status `revoked`, both applications validate the certificate and continue with the connection.

We also attempted to test the Android port of the Firefox browser, but were unable to successfully import our root certificate into the application.

Windows Phone. Finally, we tested the mobile version of IE running on Windows Phone 8.0 using the Windows Phone Emulator. Mobile IE performs identically to all other tested browsers, not checking revocation information and not requesting OCSP staples.

Overall, our results on mobile devices are disheartening. Despite the increasing fraction of web traffic that is generated by these devices, *none* of the major web browsers bother to check whether any certificates they are presented with are revoked. This design decision is likely driven by the higher cost (in terms of latency and power) of obtaining revocation information on a mobile device [20]. However, this decision does not excuse the universal lack of proper OCSP Stapling support, as requesting OCSP staples for the leaf certificate requires the client to create no additional connections.

6.5 Summary

If clients do not fetch *and respect* revocation information, the revocation efforts of website administrators and CAs are wasted. To analyze client behavior, this section presented our test suite, and the results from all major desktop and mobile browsers. While we see widely disparate behavior—often even among the same browser on different platforms—we find that *no browser* meets all necessary criteria for revocation checking. Moreover, we empirically demonstrate the extent to which EV certificates are validated differently. Perhaps most concerning is the fact that no mobile browsers check for revocations *at all*.

For desktop browsers, it is difficult to determine whether the absence of revocation checking is an explicit design decision or a set of common bugs. For mobile browsers, on the other hand, we believe the complete lack of revocation checking is an explicit decision to conserve users' bandwidth.

7. CRLSETS

In response to the cost of obtaining revocation information, Google adopted an additional approach in 2013 called *CRLSets* [26]. Chrome checks a small, pre-populated list of revoked certificates. This list is updated and sent to clients out-of-band, thus eliminating the cost of checking revocation information at page load time.

In this section, we characterize CRLSets and investigate their impact on security in the face of revocations. While CRLSets currently affect only users of the Chrome browser, this approach has recently been adopted by Firefox in the OneCRL [41] project[24], so understanding the implications of this approach can have a significant impact on a large number of web users.

7.1 Overview and Methodology

The high-level CRLSet approach is documented [2], but the process used for generating them is not public. Importantly, the developers state that: (1) the size of the CRLSet file is capped at 250 KB; (2) they populate it using an "internal list of crawled CRLs" (some of which are not public) fetched on the order of hours; (3) if a CRL has too many entries it will be dropped from the CRLSet; and (4) revocations are included only if they use one of a small number of reason codes[25] (we refer to these as *CRLSet reason codes*).

CRLSet files are published to a public URL and fetched periodically by Chrome. We fetched these files once per

[23] The iPhone simulator for iOS 6 does not provide an option for installing root certificates.

[24] In contrast to CRLSets, OneCRL is for intermediate certificates. As of this writing, there are only 8 revoked certificates on the list.

[25] No reason code, Unspecified, KeyCompromise, CACompromise, or AACompromise

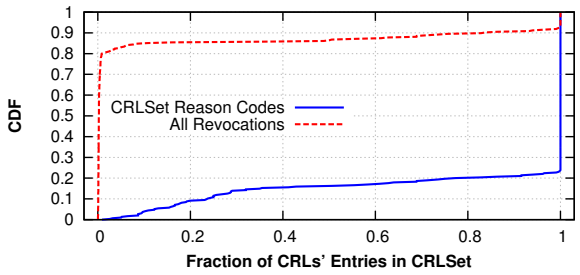

Figure 7: Cumulative distribution of the fraction of a CRL's entries that appear in CRLSets. Also shown are just those entries that have CRLSet reason codes. Most certificates with "valid" reasons are included the CRLSet, but most CRL entries overall are missing.

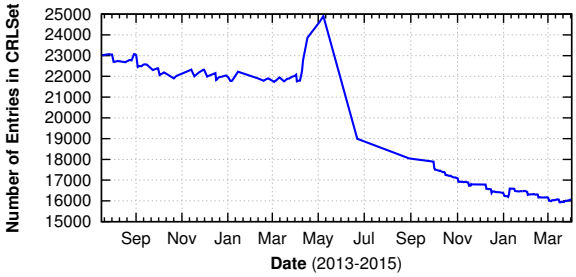

Figure 8: Number of entries in the CRLSet over time (note that the y-axis does not start at 0). The peak of nearly 25,000 entries is due to the Heartbleed vulnerability; since then the size of the CRLSet has decreased over time by more than a third.

day between September 23, 2014 and March 31, 2015, and crawled 110 historical CRLSets originally published between July 18th, 2013 and September 23, 2014; in total our dataset contains 300 unique CRLSets. CRLSets are internally organized as a list of key/value pairs, where the key is the SHA256 of the certificate issuer's public key (which we call the *parent*), and the values for that key are all the serial numbers of revoked certificates signed by that parent.[26]

Mapping a revoked certificate in a CRLSet to a CRL is nontrivial because CAs can have multiple signing certificates, public keys, and CRLs. When comparing CRLSet entries with CRLs (using the data described in §3.2), we compare the entries in the CRLSet for each parent with the entries in *all* CRLs for certificates signed by that parent.

7.2 CRLSet Coverage

We first determine the fraction of CAs covered by CRLSets. In our CRL dataset, we see 2,168 CA certificates (both intermediate and root) with 1,584 distinct public keys. In CRLSets, we observe 62 unique parents. Thus, CRLSets cover revocations for only 3.9% of CA certificates.[27] Of those 62, we can only map 52 to a CA certificate that we know of; the remainder are likely from non-public CA certificates.

Next, we investigate how entries in CRLSets compare with those in CRLs. Across all CRLs we crawled, we observe 11,461,935 revoked certificates; only 41,105 of these appear in CRLSets. Thus, only 0.35% of all revoked certificates ever appear in the CRLSet. Additionally, of the 2,800 total CRLs we observed, only 295 (10.5%) have ever had *any* entry appear in a CRLSet (this number is larger than the number of parents, as a given parent can sign multiple CRLs). We refer to these CRLs as *covered CRLs*.

Covered CRLs. For the covered CRLs, we examine the fraction of their entries that appear in the CRLSet. Figure 7 presents the cumulative distribution of CRLSet coverage across covered CRLs. Since the Chromium project states that only revocations with a certain reason codes will appear in the CRLSet, we also plot the cumulative distribution of coverage only for entries with CRLSet reason codes as well.

We find that for 75.6% of covered CRLs, *all* entries with CRLSet reason codes appear in the CRLSet. For the remaining CRLs, we find that the number of entries with CRLSet reason codes missing ranges from 1 to 5,826 (the latter comes from Verisign's "International EV" CRL, making it especially strange). Overall, we observe that for most covered CRLs, CRLSets provide perfect coverage; however, there are many CRLs for which CRLSets provide only partial coverage.

Un-covered Revocations. We now address the question of what certificates are *not* included in CRLSets. While there are millions of revoked certificates that do not appear in CRLSets, we focus on those that belong to popular websites, since they will impact the largest fraction of Internet traffic. Across all Alexa top one million domains, we see 42,225 revocations in CRLs, but only 1,644 (3.9%) in CRLSets. If we focus on the Alexa top 1,000 domains, we see 392 revoked certificates, but only 41 (10.4%) ever appear in CRLSets. Thus, unless these sites use EV certificates (which cause Chrome to perform the revocation checks described above), certificates presented by the many of the most popular domains are left unchecked by Chrome's CRLSets.

7.3 CRLSet Dynamics

We now examine CRLSet dynamics, and begin by investigating how the CRLSet size changes over time. Figure 8 shows that the number of entries in CRLSets ranges from 15,922 to 24,904, with the peak corresponding to the Heartbleed vulnerability, followed by a decrease over the following year. The significant decrease in May–June 2014 is largely explained by the removal of the "VeriSign Class 3 Extended Validation" parent (and its 5,774 entries) from the CRLSet, and general downward trend corresponds with the behavior of the underlying CRLs.

To better understand the dynamics of CRLSet entries, we zoom in on the number of entries added and removed from the CRLSet each day, and limit ourselves to the time period of October 1, 2014 to March 31, 2015 where we have daily scans of both CRLSet files and CRL entries. Figure 9 shows this, using a log scale on the y-axis. We observe that CRLs exhibit weekly patterns, with increased activity during the week and lulls during the weekend. We also observe that CRLSets are updated frequently, but there is a noticeable

[26] CRLSets also include a small set of leaf certificates that are explicitly blocked based on their public key; this list is called `BlockedSPKIs`. The list current contains 11 such certificates. Curiously, Chrome 44.0 declares these certificates as revoked in the URL Status Bar, but still completes the connection and renders the page. For an example, please see `https://revoked.grc.com/` and the corresponding discussion [8]. We have filed a bug with the Chrome developers regarding this behavior.

[27] We cannot say precisely what fraction of *CAs* are represented in CRLSets, as CAs often have many keys.

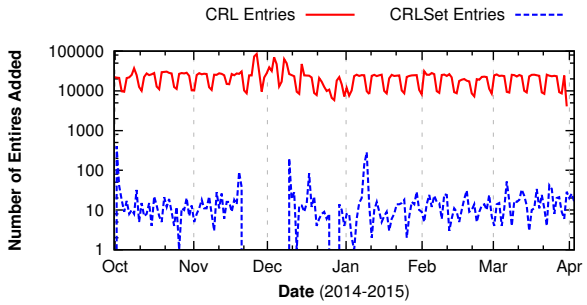

Figure 9: Number of daily new revocations appearing in CRLs (upper line) and CRLSets (lower line). CRL additions show weekly patterns and are much larger than CRLSet additions. For two weeks in late 2014, we observed no additions to CRLSets.

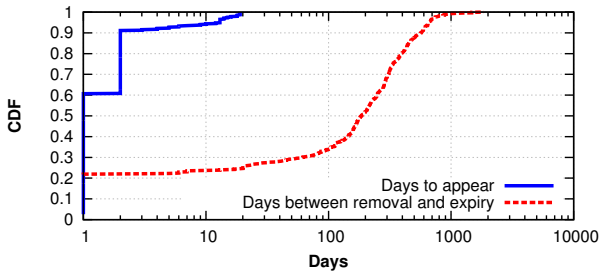

Figure 10: Number of days that clients are vulnerable to revoked certificates using CRLSets. Revoked certificates on covered CRLs are added quickly (within a day), but many revoked certificates are removed from the CRLSet well before they expire.

gap with no additions in November–December 2014. Otherwise we observe no direct correlation between peaks/troughs in CRL and CRLSet updates, consistent with the fact that CRLSets track only a small subset of the CRLs.

Lastly, we focus on the *security impact* of CRLSet dynamics, specifically for the set of revocations that appear in *both* CRLs and CRLSets. We focus on periods of time when a certificate appears in a CRL but not in the CRLSet (687 cases in our dataset), or was removed from a CRLSet before the revoked certificate expired (629 cases in our dataset). We refer to these as periods of *vulnerability*. Figure 10 shows this using the cumulative distribution of days of vulnerability over all certificates that appeared in CRLSets. We find that most revoked certificates appear in the CRLSets within one day (60%), and more than 90% appear within two days. This suggests that an automated process pulls CRL data into CRLSets, and thus there is a relatively small window of vulnerability after a revocation occurs.[28]

However, we observe that revoked certificates are sometimes removed from the CRLSet hundreds of days before they expire. This result is due to the significant decrease in CRLSet size seen in Figure 8. In one case, a revoked CA cert was removed from the CRLSet that is valid until the year 2020. In the median case, revocations are removed from CRLSets *187 days* before the certificate expires.

7.4 Improving CRLSets: Bloom Filters

As we have shown, Google's CRLSet implementation has very low coverage. An alternative data structure that could hold more revocations while also obeying Google's strict size constraints is a Bloom Filter [4]. A Bloom Filter has the advantage of no false negatives (if a certificate is revoked, it will be flagged as revoked by the filter); however, it will produce a tunable false positive rate (a non-revoked certificate may be flagged as revoked). Thus, when a certificate "hits" the filter, the client would need to check a CRL before flagging the connection as insecure. Although checking CRLs introduces lag, this lag would only impact performance for false positives (since true positives should be blocked anyway). Also note that the size of the Bloom Filter would depend on the number of revoked certificates (which is relatively easy to determine), not the number of fresh, issued certificates (which is unknown and much, much larger).

[28]Note that this does not take into account the time it takes to push updated CRLSets to clients, which may add to the window of vulnerability.

As a proof-of-concept, we briefly analyze the characteristics of a Bloom Filter-based revocation list. As shown in Figure 11, we examine the trade-offs between filter size m, the number of revoked certificates inserted into the filter n, and false positive rate p. The number of hash functions k is calculated using the optimal formula $k = \left\lceil \frac{m}{n} * \log 2 \right\rceil$.

Figure 11 highlights the advantages of our Bloom Filter-based implementation over CRLSet. The grey region shows the minimum and maximum revocations covered by CRLSets over time. For the same maximum file size (256 KB), our Bloom Filter can store an order of magnitude more revocations with only a 1% false positive rate (i.e., instances where the CRL would be checked to avoid actually blocking the connection). If we use a filter that is 2 MB in size, we can cover 1.7 million revocations at this false positive rate; this represents 15% of all revocations across all CRLs in our dataset.

The results of this analysis are similar to the examination of the suitability of Bloom Filters for distributing revocation information conducted by Langley [25]; Bloom Filters show significant promise as a mechanism for dramatically increasing the coverage of CRLSets at little cost to clients. In fact, Langley suggests that using a variant of Bloom Filters called Golomb Compressed Sets [17] may serve to reduce the space requirements even more.

7.5 Summary

To summarize, we find that CRLSets have limited coverage, their coverage of revoked certificates is diminishing over time, they are updated frequently but experience outages, and they leave browsers vulnerable to unexpired revoked certificates. While the CRLSet approach has a number of

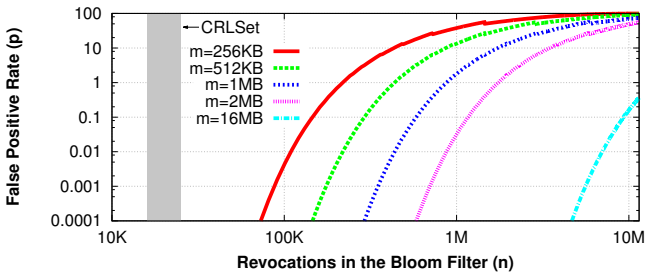

Figure 11: Number of revocations that would fit in Bloom Filters of various sizes, compared to CRLSets of today.

benefits to users, it is clear that the developers have chosen to provide security only for a small fraction of sites in order to decrease the amount of bandwidth required to disseminate revocation information; this leaves the vast majority of popular sites uncovered.

8. RELATED WORK

SSL Certificate Ecosystem. There has been a long thread of work that examines the SSL certificate ecosystem, ranging from the contents of different browser/OS root stores [36,50] to the trust relationships between CAs [14] and the patterns of certificates advertised by servers [22]. While Durumeric et al. [14] briefly examine certificate revocation in their study, our work builds on theirs by examining the dynamics of revocation behavior, different revocation protocols, and the cost of obtaining revocation information.

Researchers have proposed ideas to reduce the level of trust placed in CAs. Significant recent work has gone into increasing the transparency of the SSL ecosystem [24, 29, 30, 39, 43], ideally leading to better accountability. For example, the Certificate Transparency Project [29] introduces public logs to monitor all certificates issued by CAs. Others have proposed alternate architectures to the current CA-based system [9,12,47], often requiring modifications to both servers and clients. The DANE [47] proposal, for example, allows the owner of a DNS domain to distribute their public keys via DNS itself (thereby obviating the need for SSL certificates), while relying on DNSSEC to secure the binding.

Studying and Improving Revocations. There has been significant debate over the actual effectiveness of certificate revocation in practice [27, 28, 32], and researchers have attempted to improve the certificate revocation process and dissemination mechanisms. For example, there are proposals for very short certificate lifetimes [46], ideally making revoking a certificate as easy as not renewing it. Alternatively, Schulman et al. [42] proposed a broadcast system that disseminates revocation data over FM radio.

Two recent security incidents (a Debian vulnerability and the Heartbleed bug) provided opportunities to study whether site operators who should revoke their certificates actually do [13, 51, 52]. The results are disheartening, as only around 10% of vulnerable certificates were correctly revoked. Our work is complementary: we look at a larger set of certificates over a longer period of time, we investigate the burden that disseminating revocation information has on CAs, and explore when clients obtain this information.

Client-side Certificate Validation. Finally, much work has gone into understanding client-side behavior when creating SSL connections. Recent work has shown that many pieces of non-browser software do not correctly implement SSL checks, often due to mis-use of SSL APIs [18,19]. Others have shown that the SSL libraries themselves are buggy [10]. Because browsers often ask users whether or not to proceed when the SSL connection is suspicious, other work has focused on how browsers present warnings and how users react [6, 7]. Taken together, these results show that correctly implementing SSL validation is difficult to get right; our results show that this situation is further complicated by browsers that decline to check revocation information at all.

The GRC browser revocation test [21] is similar to our "test suite", but ours is much more comprehensive (the GRC test only contains a single certificate). As a result, our conclusions differ: while Firefox performed well on their test, we found that Firefox fails in other configurations of certificate chains/revocation protocols.

9. CONCLUDING DISCUSSION

Certificate revocation is a necessary component of any PKI, but it comes with costs, both real and perceived: CAs carry the cost of disseminating revocation information, while browsers risk the cost of increased web page load times. In the trade-off between low communication overheads and security, both ends of certificate revocation (those who issue and those who fetch) are naturally tempted towards the former. Indeed, the very utility of revocations has been debated and doubted [28] by the security community, but to date, these debates have had to largely depend on anecdotal CA and browser behavior.

We have presented in this paper an empirical measurement of the options at all parties' disposal—website administrators, CAs, and browsers—in terms of the communication overhead costs they impose and the extent to which they are currently being employed.

Overall, our results show that, in today's Web's PKI, there is extensive inaction with respect to certificate revocation. While many certificates are revoked (over 8% of fresh certificates and almost 1% of alive certificates), many web browsers either fail to check certificate revocation information or soft-fail by accepting a certificate if revocation information is unavailable.

On the positive side, our results also demonstrate that there are several clear paths to improvement. To reduce CRL sizes, CAs can simply maintain more, smaller CRLs (in the extreme, each certificate could be assigned a unique CRL, resulting in an approximation of OCSP)—surprisingly few CAs currently take such an approach (§5) and therefore incur greater bandwidth costs than strictly necessary. A promising improvement is OCSP Stapling, as it reduces CA bandwidth costs as well as web page load times—yet, not all browsers support it, and some that do simply ignore the responses. A more pervasive deployment of OCSP Stapling, at both websites and browsers, could lead to an immediate improvement in user security at little additional performance cost, particularly if the Multiple OCSP Staple Extension [37] were adopted to allow intermediate certificates to be stapled. Finally, we show that a straightforward modification to CRLSets could increase their coverage by several orders of magnitude.

From these results, we conclude that certificate revocation ought not be given up on—that indeed it serves a critical yet overlooked role that, with proper support from all parties, can be achieved at a cost far outweighed by the benefits. To realize this, we believe continued measurement and validation of future browsers will be of utmost importance, and to this end have made our data and our browser test suite publicly available at `http://www.sslresearch.org`

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This research was supported by NSF grants CNS-1054233, CNS-1409191, CNS-1319019, CNS-1421444, and CNS-1345284, by USAF award FA8750-14-2-0150, and by the NSA as part of a Science of Security lablet.

10. REFERENCES

[1] Network Security Services. Mozilla Developer Network, 2014. http://mzl.la/1DRKqGZ.

[2] CRLSets. The Chromium Projects, 2015. http://bit.ly/1JPsUeC.

[3] Network Stack. The Chromium Projects, 2015. http://bit.ly/1GYuMhE.

[4] B. Andrei and M. Michael. Network applications of bloom filters: A survey. *Int. Math.*, 1(4), 2004.

[5] C. Arthur. DigiNotar SSL certificate hack amounts to cyberwar, says expert. *The Guardian*. http://www.theguardian.com/technology/2011/sep/05/diginotar-certificate-hack-cyberwar.

[6] D. Akhawe and A. P. Felt. Alice in Warningland: A Large-scale Field Study of Browser Security Warning Effectiveness. *USENIX Security*, 2013.

[7] D. Akhawe, B. Amann, M. Vallentin, and R. Sommer. Here's My Cert, So Trust Me, Maybe?: Understanding TLS Errors on the Web. *WWW*, 2013.

[8] An Evaluation of the Effectiveness of Chrome's CRLSets. Gibson Research Corporation. https://www.grc.com/revocation/crlsets.htm.

[9] A. Bates, J. Pletcher, T. Nichols, B. Hollembaek, and K. R.B. Butler. Forced Perspectives: Evaluating an SSL Trust Enhancement at Scale. *IMC*, 2014.

[10] C. Brubaker, S. Jana, B. Ray, S. Khurshid, and V. Shmatikov. Using Frankencerts for Automated Adversarial Testing of Certificate Validation in SSL/TLS Implementations. *IEEE S&P*, 2014.

[11] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, IETF, 2008.

[12] Convergence. http://convergence.io.

[13] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson. The Matter of Heartbleed. *IMC*, 2014.

[14] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS Certificate Ecosystem. *IMC*, 2013.

[15] C. Ellison and B. Schneier. Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure. *Computer Security Journal*, 16(1), 2000.

[16] EFF SSL Observatory. https://www.eff.org/observatory.

[17] P. Felix, S. Peter, and S. Johannes. Cache-, hash- and space-efficient bloom filters. *Experimental Algorithms*, Springer, 2007.

[18] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why Eve and Mallory Love Android: An Analysis of Android SSL (in)Security. *CCS*, 2012.

[19] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The Most Dangerous Code in the World: Validating SSL Certificates in Non-browser Software. *CCS*, 2012.

[20] P. Gutmann. Engineering Security. 2014. https://www.cs.auckland.ac.nz/~pgut001/pubs/book.pdf.

[21] S. Gibson. Security Certificate Revocation Awareness Test. 2014. https://www.grc.com/revocation.htm.

[22] R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSL Landscape – A Thorough Analysis of the X.509 PKI Using Active and Passive Measurements. *IMC*, 2011.

[23] ICSI SSL Notary. http://notary.icsi.berkeley.edu.

[24] T. H.-J. Kim, L.-S. Huang, A. Perring, C. Jackson, and V. Gligor. Accountable Key Infrastructure (AKI): A Proposal for a Public-key Validation Infrastructure. *WWW*, 2013.

[25] A. Langley. Smaller than Bloom filters. 2011. https://www.imperialviolet.org/2011/04/29/filters.html.

[26] A. Langley. Revocation checking and Chrome's CRL. 2012. https://www.imperialviolet.org/2012/02/05/crlsets.html.

[27] A. Langley. Revocation still doesn't work. 2014. https://www.imperialviolet.org/2014/04/29/revocationagain.html.

[28] A. Langley. No, don't enable revocation checking. 2014. https://www.imperialviolet.org/2014/04/19/revchecking.html.

[29] B. Laurie, A. Langley, and E. Kasper. Certificate Transparency. 2013. https://tools.ietf.org/html/rfc6962.

[30] S. Matsumoto, P. Szalachowski, and A. Perrig. Deployment Challenges in Log-based PKI Enhancements. *EuroSec*, 2015.

[31] Mozilla piles on China's SSL cert overlord: We don't trust you either. http://bit.ly/1GBPwfG.

[32] NetCraft. How certificate revocation (doesn't) work in practice. 2013. http://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html.

[33] NetCraft. OCSP Server Performance in April 2013. 2013. http://news.netcraft.com/archives/2013/05/23/ocsp-server-performance-in-april-2013.html.

[34] D. Olivier. *ASN. 1 communication between heterogeneous systems*. Morgan Kaufmann, 2001.

[35] OS X Yosemite: List of available trusted root certificates. https://support.apple.com/en-us/HT202858.

[36] H. Perl, S. Fahl, and M. Smith. You Won't Be Needing These Any More: On Removing Unused Certificates from Trust Stores. *FC*, 2014.

[37] Y. Pettersen. The Transport Layer Security (TLS) Multiple Certificate Status Request Extension. RFC 6961 (Proposed Standard), IETF, 2013.

[38] Qualys SSL Pulse. https://www.trustworthyinternet.org/ssl-pulse/.

[39] M. D. Ryan. Enhanced Certificate Transparency and End-to-End Encrypted Mail. *NDSS*, 2014.

[40] Rapid7 SSL Certificate Scans. https://scans.io/study/sonar.ssl.

[41] Revoking Intermediate Certificates: Introducing OneCRL. *Mozilla Security Blog*. http://mzl.la/1zLFp7M.

[42] A. Schulman, D. Levin, and N. Spring. RevCast: Fast, Private Certificate Revocation over FM Radio. *CCS*, 2014.

[43] P. Szalachowski, S. Matsumoto, and A. Perrig. PoliCert: Secure and Flexible TLS Certificate Management. *CCS*, 2014.

[44] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960 (Proposed Standard), IETF, 2013.

[45] StartSSL: Frequently Asked Questions. https://www.startssl.com/?app=25.

[46] E. Topalovic, B. Saeta, L.-S. Huang, C. Jackson, and D. Boneh. Towards Short-Lived Certificates. *W2SP*, 2012.

[47] The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. 2012. https://tools.ietf.org/html/rfc6698.

[48] Trustwave to escape 'death penalty' for SSL skeleton key. http://bit.ly/1RbPlNe.

[49] University of Michigan Daily Full IPv4 HTTPS Handshakes. https://scans.io/series/https-full-ipv4.

[50] N. Vallina-Rodriguez, J. Amann, C. Kreibich, N. Weaver, and V. Paxson. A Tangled Mass: The Android Root Certificate Stores. *CoNEXT*, 2014.

[51] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When Private Keys Are Public: Results from the 2008 Debian OpenSSL Vulnerability. *IMC*, 2009.

[52] L. Zhang, D. Choffnes, T. Dumitras, D. Levin, A. Mislove, A. Schulman, and C. Wilson. Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. *IMC*, 2014.