Changing of the Guards: Certificate and Public Key Management on the Internet

Carl Magnus Bruhner[1], Oscar Linnarsson[1], Matus Nemec[1], Martin Arlitt[2], and Niklas Carlsson[1]

[1] Linköping University, Sweden
[2] University of Calgary, Canada

Abstract. Certificates are the foundation of secure communication over the internet. However, not all certificates are created and managed in a consistent manner and the certificate authorities (CAs) issuing certificates achieve different levels of trust. Furthermore, user trust in public keys, certificates, and CAs can quickly change. Combined with the expectation of 24/7 encrypted access to websites, this quickly evolving landscape has made careful certificate management both an important and challenging problem. In this paper, we first present a novel server-side characterization of the certificate replacement (CR) relationships in the wild, including the reuse of public keys. Our data-driven CR analysis captures management biases, highlights a lack of industry standards for replacement policies, and features successful example cases and trends. Based on the characterization results we then propose an efficient solution to an important revocation problem that currently leaves web users vulnerable long after a certificate has been revoked.

1 Introduction

Aided by several initiatives (e.g., [1, 16, 68]), the last decade saw a major shift from non-encrypted to encrypted web traffic. Today, most websites use HTTPS [9, 14, 37] and other TLS-based protocols (e.g., QUIC [48]) to deliver their content. These protocols rely heavily on X.509 certificates. At a high-level, before a secure and trusted connection can be established, the server must present the client with a valid X.509 certificate that maps the server's public key to the server's domain and that has been issued (and signed) by a recognized Certification Authority (CA) that is *trusted* by the client. Since users expect 24/7 secure access to trusted services, it is therefore important that the servers present clients with a valid and trusted certificate. This has made careful certificate management an important problem.

Careful certificate management is also a challenging problem, as not all certificates are created and managed in the same way. From a domain administrator perspective, there are many issues to consider. For example, there are many issuing CAs and certificate types with different issuing processes and costs, the trust and usage of different CAs is changing over time, different services have different security requirements, and the trust in individual keys may quickly change.

This version of the contribution has been accepted for publication, after peer review (when applicable) but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/978-3-030-98785-5_3. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms

To complicate the situation, for a number of reasons [76] modern browsers do not perform sufficient revocation checks [28] to protect users against man-in-the-middle attacks made possible by compromised keys even after they have been revoked by the domain owner and its CA [55]. While Chrome and Firefox browsers periodically (e.g., with software updates) push a proprietary set of revocations to their users [41, 59], the frequency and size of such revocation sets leave clients vulnerable long after most compromised certificates have been revoked. The situation appears most pressing for mobile browsers. For example, Liu et al. [55] found that not a single native mobile browser on iOS, Android, or Windows Phones checks the revocation status of certificates. Finally, regardless of the choices made by the websites and CAs to address these challenges, websites (with the help of the CAs) need to make sure that they always can present their clients with valid and trusted certificates.

In this paper, we (1) present a novel server-side characterization of the certificate replacement (CR) relationships observed in practice, which provides insights into biases in how services manage their certificates; (2) examine the subset of CRs that reuse the same key when a certificate is replaced; and (3) demonstrate how targeted modifications to how CRs with reused keys are handled can reduce the reliance of revocation checks and solve this revocation problem.

Our analysis is based on data extracted from all biweekly scans of port 443 (Oct. 30, 2013 to Jul. 13, 2020) done within Rapid7's Project Sonar [2]. After presenting our methodology (§ 2), we characterize the full set of CRs (§ 3) that highlights positive trends and behaviors. For this analysis, we use mismanagement indicators and study how much safety margin servers use (e.g., in terms of validity period overlap), differences in the timing of validity periods and when certificate changes actually are observed, and whether there are replacement differences based on validity type, key reuse, CA changes, and CA selection.

Our characterization demonstrates and highlights the effects of a lack of general industry standards for replacement policies [38]. This includes, for example, a clear discrepancy in the overlap patterns between the top-issuing CAs, dividing those having automated renewal/replacement support and those dependent on manual effort. However, despite several of the CAs issuing cheaper domain validated (DV) certificates with shorter validity periods using common validity-period overlaps, the least gaps (defined as CRs in which the validity periods of the replaced and replacing certificates are non-overlapping) and early/late usage of certificates are still associated with more expensive certificates using extended validation (EV). Positive trends include a decreasing fraction of CRs with gaps, and a decreasing fraction of certificates being observed in use before they are valid or after they have expired. We also observe that the decision to change CAs often is associated with gaps, but that the decision to reuse keys is not.

The later parts of the paper look closer at two particularly interesting aspects identified in the dataset and motivated by our findings, respectively. First, we study the subset of CRs in which the same key is reused by the replacing certificate. Here, we also examine the "replacement chains" that are formed when the same key is reused for a series of consecutive CRs in which the replacing

certificate of CR i is the replaced certificate in CR $i+1$ of the series. Throughout the paper, we call such a CR and chain a *Same Key CR* (SKCR) and an *SKCR chain*, respectively. Our analysis highlights big differences in how customers of different CAs reuse keys. While the customers of three CAs (Sectigo, GlobalSign, Go Daddy) had higher than 65% key reuse, the customers of several other CAs (e.g., Google, cPanel, Amazon, Microsoft) typically did not appear to reuse keys. Encouragingly, the three CAs with the most key reuse achieved substantially fewer gaps when reusing keys than when not reusing keys. However, while SKCRs make up only 14% of Let's Encrypt's customers' CRs, they present the perhaps most interesting use case. For example, by combining longer key-reuse chains with consistent issuing of 90-day certificates with 30-day overlaps, their customers achieve high relative key utilization (e.g., aggregated lifetime compared to aggregate validity period over the certificate making up the reuse chains) without having to frequently replace the public keys used on their servers.

Finally, motivated by the effectiveness and potential of some of the observed automation solutions and trends, we outline a new way (§ 5) to address the currently open revocation problem discussed above. Our solution framework is based on observations highlighted in the paper, takes some current trends to the extreme, and combines the use of short-lived three-phase certificates (modification of an idea by Rivest [66]). It also introduces the new concept of parent-child certificate relationships and new simple management rules. The framework ensures efficient use of certificates in such a way that it does not need to increase how frequently servers change their public keys or how frequently certificates must be logged in Certificate Transparency (CT) logs [50]. Using our CR datasets, we also demonstrate and quantify the reduced overhead that these efficiencies of our approach would provide when some set of CAs select to reduce their certificate lifetimes using our approach rather than naively.

In summary, the paper provides both new insights into the status of current HTTPS certificate management (§3), including the reuse of keys (§4), and novel solutions to improve certificate management and to address the currently unresolved revocation problem so far unsatisfactorily handled by browsers (§5).

2 Analysis methodology

Rapid7 dataset: We used two certificate datasets [3] from Project Sonar [2]. These datasets consist of biweekly scans of the IPv4 address space, collected using Rapid7's extensions [4] of ZMap [35]. First, we used all HTTPS `_certs` files between 2013-10-30 and 2020-07-13 to extract the full Privacy-Enhanced Mail (PEM) [44,54] encoded certificates and their SHA-1 fingerprints. Second, for our observation-based statistics, we used the corresponding `_hosts` files collected for port 443 to determine at what IP addresses and time these certificates were observed (using the SHA-1 fingerprints for mapping between the files).

Identifying and extracting CR relationships: Using the above datasets, we identify certificate replacement (CR) relationships. Here, we define a CR to exist between a pair of certificates under the following conditions. (1) The two

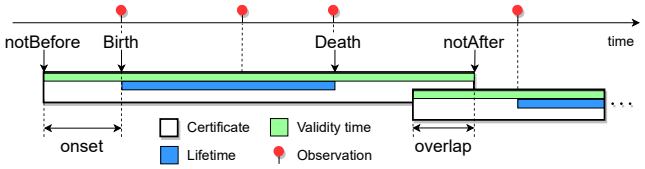


Fig. 1: Replacement relation between two certificates.

certificates were observed at the same IP address (and port number). (2) The two `subjectCN` either matched perfectly or matched after following three wildcard rules: wildcards are only allowed to be used at the lowest domain level, at the third domain level and down, and only one wildcard is allowed per match. (3) The validity period of the *replacing* certificate must begin later than the beginning of the validity period of the *replaced* certificate and must extend past the end of the validity of the replaced certificate. Figure 1 shows a toy example with overlapping validity periods (green color) and the first certificate of the CR only were observed during three scans (first three red markers). Here, the *validity period* is defined as the time between the `notBefore` and `notAfter` values in the certificate, and following the terminology used by Chung et al. [29], the *lifetime* is defined as the time period between the first scan when a certificate is observed (referred to as its *birth*) and the last scan it is observed (referred to as its *death*).

In addition to extracting information about the individual certificates and different metrics related to their relative validity periods (e.g., the *overlap* in Figure 1), we also extract information regarding when the two certificates were seen in use. Of particular interest here are cases when the servers present their certificates *before* the validity period has started or *after* it has expired.

Multi-step CR identification and extraction: We performed a series of processing steps to create an aggregated dataset including all CR relationships.

- **Step 1 (parse + process certificates):** Using a Node.js library *node-forge*[1] and OpenSSL (when node-forge was unable to parse a certificate) we extracted data from the certificates, including (1) certificate identifiers and basic information, (2) issuer and subject identifiers, (3) CA status and chain info (e.g., we determined whether the subject is a CA and whether it is self-signed, self-issued, or signed by third party), (4) validity period, (5) verification type (determined based on the Object Identifiers (OIDs) [5, 7]), and (6) public key properties.
- **Steps 2+3 (extract birth and death):** We next identify the *birth* and *death* of each certificate, respectively. In these steps, the output files were sorted based on the first birth (step 2) and last death (step 3). We also keep track of IP addresses and the number of observations.
- **Step 4 (extract CR relations):** CRs were identified one certificate at a time based on each certificate's birth. For each certificate, we search backwards in time from its time of death (increasing overlap); stopping as soon as we find a matching CR. If no such CR is found, we instead search for

[1] Available at: https://www.npmjs.com/package/node-forge/v/0.9.0

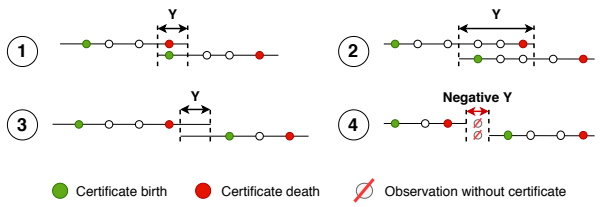


Fig. 2: Certificate replacement search order.

births forward in time (decreasing overlaps) until either such CR is found or no CR can be identified for the certificate. (Figure 2 illustrates the search order.) For every match, the precision difference between the `subjectCN` is stored, indicating if it was an exact match, if it is more precise, or if it is less precise. To allow wildcard certificates to be replaced by multiple certificates, a wildcard certificate is considered for multiple matches if it has the same birth date as the first found CR.

– **Step 5 (identify SKCR chains):** Finally, we used the SHA1 hashes of the *replaced* and *replacing* certificate of each CR in which the two certificates contained the same public key (i.e., an SKCR relationship) to identify chains of SKCRs. Starting from the base case of a single SKCR, which has a chain length of one, we search for additional CRs for which the *replaced* certificate was the *replacing* certificate of the most recent SKCR. The chaining was repeated until ether a new public key was used in the chain, or until no matching CR could be found.

Our analysis mostly focuses on the outputs from steps 4 and 5.

Limitations: Like most internet measurements, the Rapid7 dataset has its limitations. First, the biweekly scans limit how fine a granularity we can consider for CRs. Second, Project Sonar only tries each IP address once during a scan. While many certificates are seen across many IPs, this could potentially introduce biases against certificates of services with few servers or that are further away from the scanners. Third, the dataset does not capture how many real users download each certificate or how popular the services using the certificates are. Here, we treat all certificates observed in the Rapid7 datasets equally. Fourth, the Rapid7 dataset misses many certificates that may be found in CT logs [78]. While this may cause us to miss some certificates that may be of interest, the Rapid7 dataset has the advantage that it allows us to measure when a certificate was used (not only what its intended validity period is) and helps focus on certificates actually observed in the wild.

Fifth, some HTTPS servers listen on ports other than 443. The addition of scans of non-443 ports could have increased the observed lifetimes of some certificates. However, the majority of HTTPS servers use port 443. Sixth, long-lived certificates can bias the CRs observed in the beginning of the measurement period and CRs with large gaps may be missed towards the end. Given current validation period and overlap distributions, these biases should have limited impact on the set of CRs identified between 2016-2019. Despite these limitations,

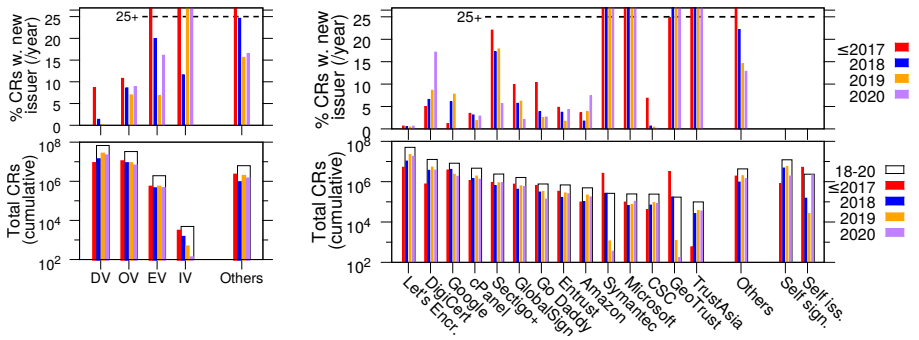


Fig. 3: Total number of certificate replacements (bottom panes) and the fractions of these for which the second certificate had a new issuer (top panes). The ranking of CAs are shown based on the certificates from the last three years (2018-2020) shown using combined boxes around those years.

we believe our analysis provides an insightful glimpse into HTTPS certificate and public key management on the Internet over the past seven years.

Finally, we acknowledge that our paper would benefit from a measure of how often security incidents exploited revoked certificates. For example, such a measure would enable a risk assessment and help determine the ideal validity period of a certificate. However, to the best of our knowledge, such a measurement has not been published and is out of scope of our paper.

3 Certificate Replacement Analysis

In total, we observed 217,221,681 unique certificates and identified 129,382,646 CRs. After filtering out self-issued and self-signed certificates, the number of CRs reduced to 108,751,863. (21.4 million CRs for the set $\leq$2017, 22.8 million for 2018, 35.9 million for 2019, and 28.6 million for 2020 (Jan-Jun).)

3.1 Certificate selection characterization

Not all certificates are created and managed in the same way. For example, different CAs offer different trust, issuing processes, and costs, and there are several validation types. Different websites therefore make different choices, and some may change CAs. Figure 3 summarizes the most common certificate choices in the last few years. The bottom panes show the total number of CRs per certificate type and issuer, and the top panes show the fraction of those CRs that changed issuer. Throughout the paper we label CRs using the characteristics of the replaced certificate and say that the issuer has changed whenever the issuer's common name (issuerCN) is different. For the per-CA breakdown we rank the CAs based on the number of CRs between 2018-2020 (shown as combined bars in the plot) and only show results for CAs with at least 100,000 CRs and for

which the majority of the certificates are approved by the major browser vendor's trust stores. When interpreting these plots, it is important to note that a CR represents a successful certificate replacement.

For the analysis in this paper, we omit self-signed and self-issued certificates. Prior works have shown that these certificates are responsible for the majority of invalid certificates in the dataset [29]. To illustrate how big a portion of the CRs that these two certificate types represent we include them in the right-hand panes here, but exclude them from all other analysis (including the results shown in the left-hand panes). Here it is important to note that the majority of the studied certificates are issued by the top-CA.

Certificate types and issuer changes: There are big differences in the issuance requirements of different validity types. Domain Validated (DV) certificates have the least requirements and Extended Validation (EV) certificates the most rigorous (and time consuming) requirements [7]. Organization Validated (OV) and Individual Validation (IV) certificates fall between the two, adding somewhat to the requirements (and costs) of DVs [5]. DV certificates is the dominating certificate type in the dataset. Customers using DV certificates seldomly change issuer. A key reason for the higher issuer change rates of the other types is likely customers switching to cheaper services (e.g., free DV certificates). Two contributing factors for users having moved away from EV certificates during this period may be (1) the introduction of Let's Encrypt's free and easy-to-use DV certificates and (2) several major browsers (e.g., Safari, Chrome, Firefox) ending or announcing the ending of user interfaces displaying EV certificates differently than DV certificates. Both these aspects are expected to have reduced the incentive to spend extra money/effort for EV certificates.

Selected CAs and their retention rates: For the per-CA breakdown, we rank CAs based on CRs between 2018-2020 and only show results for CAs with at least 100,000 CRs and for which the majority of the certificates are approved by the major browser vendors' trust stores (e.g., Apple, Microsoft, Mozilla/NSS). With these root stores having been responsible for most TLS user agents [56] at the time the dataset was collected (and before Chrome released their own root store in Dec. 2020 [6]) and all of them having significant overlaps in their root selections [45], we expect these CAs to have very good end-user reach. Both Symantec and GeoTrust have had very few CRs the last two years (purple+orange bars in bottom-right pane of Figure 3). This is also reflected by the high rate of new issuers associated with CRs involving these two CAs (constantly above 25% for all four time buckets). Domains leaving Symantec is perhaps not surprising given that Google over this time period implemented a plan to distrust Symantec [65]. Microsoft and TrustAsia have also seen high issuer churn over this time period. Of the dominant CAs, Let's Encrypt has the lowest change rate, suggesting that they have a high customer retention rate. DigiCert's change rate is increasing over time, while Google, Sectigo (formerly Comodo), and GlobalSign were able to improve their retention rates in 2020.

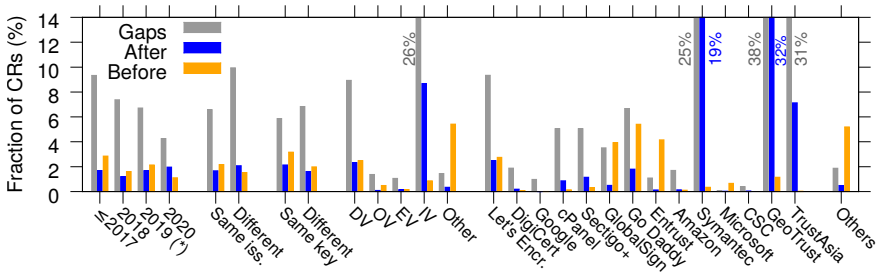


Fig. 4: Fraction of CRs with *gaps* or for which the first certificate was presented *after* it had expired or *before* it became valid. Only 2019 CRs are used here (with exception for the three categories "2020", "2018" and "≤2017" to the left).

3.2 Analysis using mismanagement indicators

Changing a certificate (or CA) is not always seamless. We next look closer at potential mismanagement indicators, including CRs with *gaps* between the validity periods of the two certificates in the CR, and the certificates that are used either *after* their expiry or *before* their validity period started. These results are summarized in Figure 4. For this and all later figures, all results are based only on the CRs from 2019 (the most recent complete year of data we have; marked "*"), unless a different year is indicated (labeled "≤2017", "2018" or "2020"), and we use the same order of the CAs as provided by the rankings in Figure 3.

Decreasing fraction of gaps: There has been a clear improvement over time. For example, since CRs including 2017 (9.34% gapped), the fraction of CRs with a gap has steadily decreased and was more than halved by 2020 (4.26%). This suggests that servers may be becoming better at replacing their certificates on time. One possible explanation is that Chrome and other browsers increasingly inform and/or block users from accessing websites that do not meet current HTTPS standards and practices, incentivizing websites to be compliant.

Changing CA more frequently results in gaps: We have observed a disproportionate fraction of gaps associated with issuer changes. This may in part be due to some administrators leaving updates until it is too late. We have also found that the first certificate in a CR with overlapping validity periods typically is used for the better part of the overlap period, suggesting that server administrators may not be in a rush to switch to the replacement certificate or that they do not always get access to them right away, even when the certificate has an overlapping validity period.

Reuse of keys: At an aggregate level, the reuse of keys does not appear to change the fraction of CR gaps, post-usage of expired certificates, or the pre-usage of not-yet valid certificates. Section 4 analyzes this case further.

CRs with EV and OV certificates have the fewest gaps: The CRs with the fewest-to-most gaps are: EV, OV, DV, and IV. This suggests that services that pay extra for EV (and OV) certificates indeed manage to ensure that they have fewer CR gaps than organizations that use cheaper DV certificates.

This could potentially be due to differences in operational support between such websites. As IV usage is becoming increasingly rare, people may find fewer reasons to keep them up-to-date. Furthermore, with the use of timestamped code, code-signing certificates can be used for validation also after expiration [67].

Management indicators differ substantially across CAs: First, there is a big difference between the CAs with the largest fraction of gapped (bad) and overlapping (good) CRs. Like for validation types, the largest fraction of gapped CRs is associated with the CAs with the lowest retention rates and decreasing usage: Symantec (25%), GeoTrust (38%), and TrustAsia (32%). As expected, these three CAs also have the largest fraction of certificates used after their expiry date. We also observed 9.3% gaps associated with the free DV certificates issued by Let's Encrypt. In contrast, CRs with certificates issued by Microsoft (0.08%), CSC (0.4%), Google (1.0%), Entrust (1.1%), Amazon (1.7%), and DigiCert (1.9%) are much less likely to have gaps. The relatively low fraction of gaps suggests a significant level of automation and/or better process for certificate replacements. Second, we have observed a substantially higher fraction of certificate observations timestamped *before* they were valid when issued by four (different) CAs: Go Daddy (5.4%), Entrust (4.2%), GlobalSign (3.3%), and Let's Encrypt (2.8%). While the exact fraction of certificates observed early may be inflated by the granularity and accuracy of our *birth* estimates, the significant differences between the CAs are substantial and shows that some CAs may use significantly bigger safety margins than others to ensure that some clients (with clock offsets, for example) does not invalidate an okay certificate. Careful certificate management include both using sufficient overlap in the validity periods and deciding when to switch from using one certificate of a CR to the next.

The above differences may also be an indication that different classes of organizations are more likely to choose certain CAs. For example, one would expect significant differences in the fraction of gaps between organizations that depend on HTTPS for their business and those that simply want a web presence. The latter likely lack either the incentives or the means to prevent problems like gaps. Our results also suggest that organizations that are looking to switch CAs (possibly due to cost) are likely to contribute to yet additional one-time gaps.

EV and OV certificates are more carefully managed: Similar to gapped CRs, the fraction of certificates that are used after or before their validity period is much smaller for OV and EV certificates than for DV certificates. This again shows that organizations employing such certificates indeed appear to manage their certificates more carefully.

3.3 Overlap analysis

Overlapping validity periods are typically used to protect against service outages. To better understand the safety margins used in practice, we next compare and contrast the overlaps of different CR sets. Figure 5 shows a box-and-whisker plot of the CR overlaps and the validity periods, for different categories of CRs.

Decreasing overlaps: Regardless of which percentile we consider, overlaps have decreased over time. This observation is both interesting and encouraging,

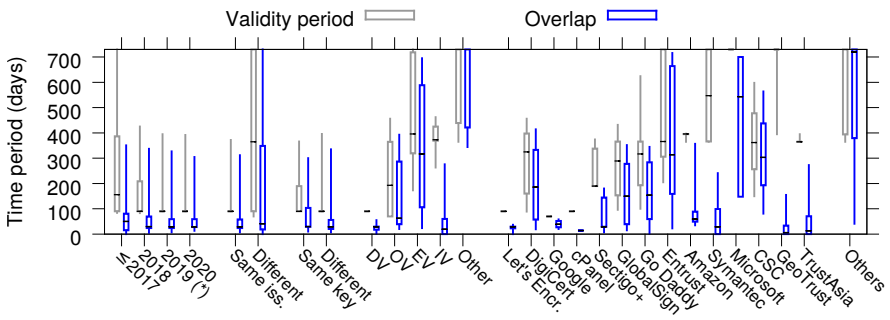


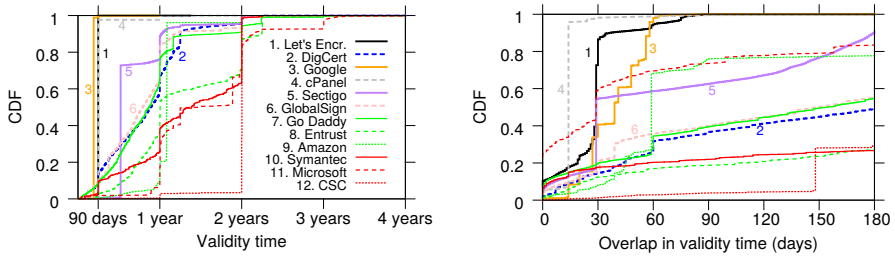
Fig. 5: Validity periods and CR overlaps. For each CR category, we show the 10-percentile (bottom marker), 25-percentile (bottom of box), median (middle/black marker), 75-percentile (top of box), and 90-percentile (top marker). Only 2019 CRs are used here (with exception for the three categories "2020", "2018" and "≤2017" to the left).

when considered in combination with our earlier observation that the fraction of gapped CRs has reduced substantially over the same period.

The above reductions have been achieved at the same time that the validation periods themselves have been reduced. The reduction in validity periods is particularly clear when considering the fraction of long-lived certificates (e.g., see 90-percentile values in Figure 5) that have been pushed away by new regulations and best practices such as the CA/Browser Forum Baseline Requirements (BR) [5]. This trend towards shorter validity periods is expected to continue. As an example, in March 2020, Apple decided to reduce the maximum allowed lifetime of certificates in its root policy to 398 days (previously 825 days) for certificates issued starting September 2020 [12, 31]. Chrome (June), Mozilla (July) and the BR (July) have since followed suit [25, 40, 81].

Gap issues with older (long-lived) certificates: There is a disproportionate fraction of gaps associated with instances where old long-lived certificates expire. One possible explanation is that without automated solutions, the use of long-lived certificates increases the chance that a customer forgets to renew the certificate in time. While we expect CAs using automated issuance to be more likely to use short-lived certificates and provide/support automated certificate replacement solutions for their customers, we could only see a weak correlation between the CAs with longer validation periods and those with more gaps, when excluding Let's Encrypt. Instead, Let's Encrypt had a surprising number of gaps. We expect this to have more to do with the domains that select to use free certificates than the service provided by Let's Encrypt. However, more work is needed to confirm the underlying reasons for the above observations.

Subject vs issuer dominated overlap decisions: Clearly, all subjects have some control over the certificate overlaps: first the size of the validity period overlaps and second the specific date and time when to replace a certificate (and a key). Subjects can also decide if and when to change their issuing CA. However, the variation in these overlaps and decisions differ substantially depending



(a) Validity periods (b) Overlap in validity periods

Fig. 6: CDFs for the 12 most frequently observed CAs. (Based on CRs in 2019.)

on which issuing CA the subject uses. For example, the CAs with longer validity times have higher variability in the overlap distribution relative to CRs with certificates issued by CAs with shorter validity periods. One possible explanation is that the overlap of short-validity certificates is influenced more by the issuer than the subject, whereas the overlap with long-validity certificates appears influenced more by the subject. We again do not see any major differences when conditioning on whether the key is reused in the certificates.

DV certificates use shorter validity periods and overlaps than OV and EV certificates: The validity periods are shorter for DV certificates regardless of whether Let's Encrypt certificates are included or excluded. While the average statistic (not shown) increases from 103 to 187 days when excluding Let's Encrypt, these values are still lower than for OV (241 days) and EV (461 days). One reason is that DV certificates typically use a faster validation process, simplifying use of short-lived certificates. Interestingly, DV certificates typically also use much shorter overlaps (safety margin). For example, the EV certificates have a median overlap of 317 days compared to 29 days for DV certificates.

Distinguishing features of CAs: There are big differences in the validity periods and the overlaps. For example, Let's Encrypt, Google, and cPanel always use 90-day validity periods; services using their certificates have fairly specific overlaps for the majority of their CRs (typically 15 or 30 days). In contrast, most of the other CAs use much longer validity periods and their customers use both larger and much more diverse overlaps.

To better understand the distinct behaviors observed for the customers of different CAs, Figure 6 shows the empirical cumulative distribution function (CDF) of the validity periods and CR overlaps observed in 2019 for each of the top-12 CAs in our dataset. To improve readability, the top-6 are shown with distinct color coding and enumeration, the CAs with ranks 7-9 are shown in green, and the CAs with ranks 10-12 are shown in yellow.

Increasing validity periods with decreasing rank: While there are exceptions, we note a clear shift in the CDFs based on the CA ranks. The three CAs with the shortest validity periods (left-most CDFs in Figure 6) are roughly followed by the three CAs with ranks 2, 5 and 6, which are followed by the three CAs with ranks 7-9 (green curves), which finally are followed by the CAs with

ranks 9-12 (yellow curves). This is in part due to some of the top-ranked CAs now offering attractive low-cost certificates with simple, automated validation checks and shorter validity periods. These distribution examples also show that many of the less popular CAs have had to make relatively bigger changes to comply with the recently imposed 398-day limit.

Automated replacement solutions: Some of the CAs have a clear "knee" in their overlap distributions. This behavior appears to be due to default values used in automated processes simplifying certificate management. For example, the two CAs with the most significant knee are cPanel (rank 4) and Let's Encrypt (rank 1). These CAs typically have an overlap of 15 and 30 days, respectively.

Both Let's Encrypt and cPanel automate some of their certificate services with the recently standardized Automatic Certificate Management Environment (ACME) [9,17]. Let's Encrypt, for instance, has created its own automation tool Certbot as an ACME agent [75]. The cPanel system also issues other certificates, and almost one out of five Let's Encrypt certificates are issued using cPanel [9].

Other CAs with sharp (although smaller) knees in the overlap distribution are Google (multiple steps), Sectigo (30 days), Amazon (60 days), Microsoft (30 days), and CSC (148 days). This suggests that websites using these services also use automated certificate replacement processes to a significant degree.

4 Reuse of Keys

There is a cost associated with mapping subjects to keys. Unless a key has been compromised, in some cases it may therefore be desirable to keep using the same key when issuing a new certificate. For example, servers do not have to replace their private keys and the CA could potentially simplify the domain validation process somewhat knowing that the domain already is in possession of the key. We call a CR where the public key is reused a Same Key CR (SKCR). The fraction of SKCRs is increasing and are today responsible for roughly 13% of all CRs. We next look closer at the SKCRs and the SKCR chains formed when a key is reused for consecutive replacements.

4.1 High-level SKCR analysis

Figure 7 shows the fraction of CRs that reuse the same key (black bars), and the fraction of those that have *gaps* (purple bars). As a reference point, we also include the overall fraction of gaps for each category (× markers). These reference point values are the same that were reported in Figure 4.

Small difference or reduced fraction of gaps: In most cases, reusing a key has limited effect on the results. We have only seen a few cases when services reusing keys have more gaps: the issuer changes, the certificate is of type EV, and (at a first glance) the first certificate in a CR was issued by certain CAs. However, these cases can be explained by a change in CA (the characteristic most likely resulting in a gap). For example, the three CAs (Google, cPanel, CSC) with noticeable higher fraction of SKCR gaps (i.e., higher purple bars

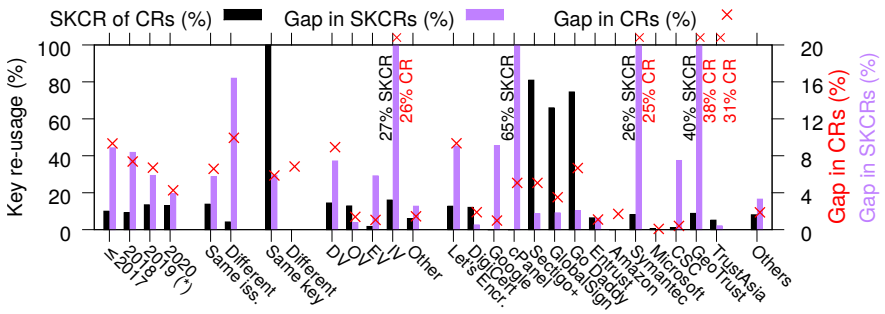


Fig. 7: Fraction of CRs that are SKCRs and have *gaps*. Only 2019 CRs are used here (with exception for the 3 categories "2020", "2018", "≤2017" to the left).

than × markers), as well as Amazon, have very small key reuse (black bars). It appears that these CAs typically do not allow reuse of keys. Instead, these rare cases are associated with a customer re-using their key with a different CA. (Google, cPanel, and Amazon has less than 0.003% reuse and CSC 0.6%.)

Three CAs with high key reuse and fewer gaps: The customers of three CAs have higher than 65% key reuse: Sectigo (81%), GlobalSign (66%), and Go Daddy (75%). The highest reuse among the other CAs is less than 13%. Interestingly, these three CAs (together with DigiCert and Trust Asia) also achieved less gaps when reusing a key (shorter purple bars than × markers). We believe that the reuse of keys is part of the operational practices of these CAs and may simplify the validation process as well as the key and certificate management process of the customers. In Section 5 we expand on this observation and show how key reuse can be used as a building block in an improved certificate management system.

4.2 SKCR-chain analysis

Let us next consider the *SKCR chains* formed when the same key is reused for a series of SKCRs (numbered from first to last) in which the replacing certificate in SKCR i is the certificate being replaced in SKCR $i+1$ of the series.

Short chains are common: We have found that most chains are short (e.g., CDF in Figure 8 shows that 80% of the chains of lengths at least two are no longer than five) and that the tail of the chain-length distribution has exponential characteristics (e.g., straight-line CCDF behavior on linear-log scale).

Long chains are dominated by automated services: Figure 9 shows the CDFs of the aggregate validity periods, when merging the validity periods of all certificates associated with an SKCR chain. When interpreting this figure, note that the single certificate line roughly captures the overall validity period distribution across all certificates. For example, as shown in Figures 5 and 6, most certificates have a validity period of 90 days (Let's Encrypt, Google, cPanel) or around either one or two years (most other top-CAs). While we see some chains of length two (pink line) that clearly include long-lived certificates (e.g., steps

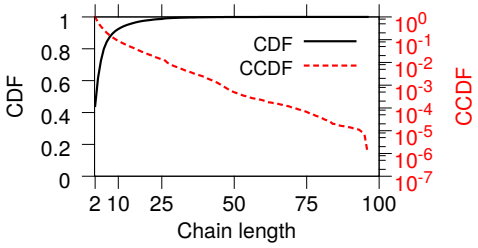


Fig. 8: Chain length distributions.

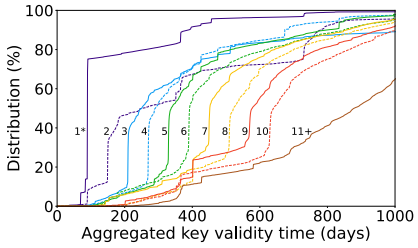
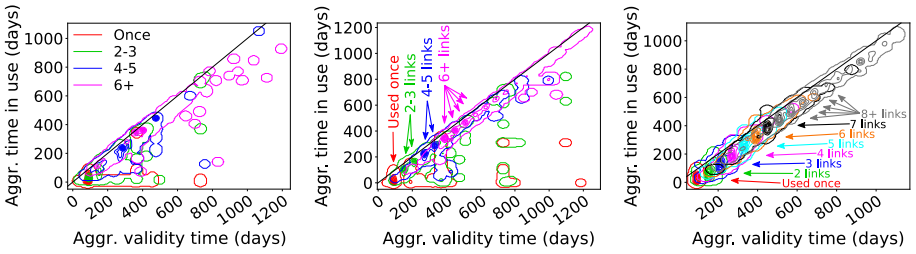


Fig. 9: Aggregate validity of SKCRs (one CDF per chain length)

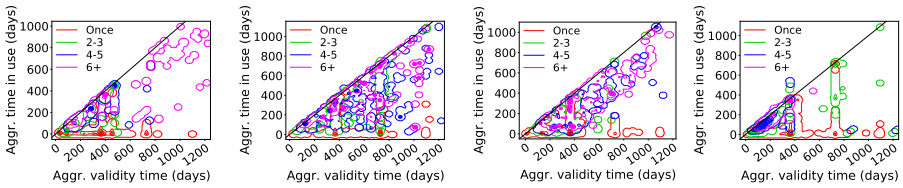


(a) Replaced by new chain (b) Not replaced (c) Only Let's Encrypt(#1)

Fig. 10: Aggregate validity period vs aggregate observed use for SKCR chains.

around 1 year and 2 years), close to 40% of the chains with length 2 (pink line) have an aggregate validity period of 150 days (90+60), matching our previous observation that most of the Let's Encrypt certificates have a validity of 90 days and an overlap of 30 days. For chain lengths of three, the higher age steps have almost disappeared, and the aggregate duration instead appears to be dominated by Let's Encrypt chains. This is seen by the consecutive CDFs being shifted by roughly 60 days up-to a chain length of 10 certificates. The 11+ curve includes a mix of longer chain lengths (mapped to different CAs) and therefore has a somewhat different general shape, without any distinct steps.

Keys used in chains are typically used for close to the full aggregate validity period: Figure 10 shows contour plots of the aggregate validity period versus (vs) the aggregate observed use for certificate-replacement chains of different replacement lengths. For easier visualization, we use contour plots. These plots are based on data between 2017-01-02 and 2020-07-13 and are generated using a matrix granularity of 5 days (meaning that any point falling within any of the 5×5 possible day combinations would add to the same counter), outliers are removed using a threshold of 0.15% of the total observations, and we have applied a Gaussian smoothing with a smoothing constant (sigma) of 2, where the smoothing can be seen as us simply taking the sum across twice as many buckets when doing a regional summation. We separate results based on whether the chain is eventually replaced by a new chain (Figure 10(a)), which uses a different key, or whether no additional certificate replacement is observed (Figure 10(b)).



(a) DigiCert (#2) (b) Sectigo (#5) (c) GlobalSign (#6) (d) Go Daddy (#7)

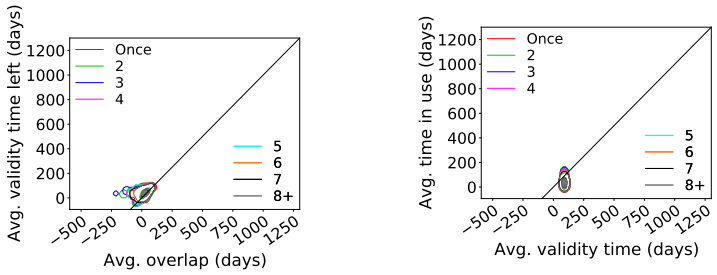
Fig. 11: CA-based comparison of the validity vs. use as aggregated over all certificates in certificate-replacement chains. (Rank in parenthesis.)

While we observe more variations in both use and validation periods for the second case, the general characteristics and observations are the same for both cases. First, the largest volumes (peaks or ridges in the plots) are observed just under the diagonal, suggesting that the observed use typically spans almost the full aggregate validity period. This shows that the websites typically make good use of the aggregate validity period. Second, we the shortest chains (red) have the largest portion of points well below the diagonal.

Behavior varies noticeably by CA: Looking closer at individual CAs, Let's Encrypt has the most interesting behavior. Figure 10(c) shows how it nicely stacks up longer and longer chains along the diagonal. As discussed above, this is desirable and demonstrates good use of the aggregated validity periods. This shift comes from Let's Encrypt customers consistently using a validity period of 90 days and often using an overlap of approximately 30 days.

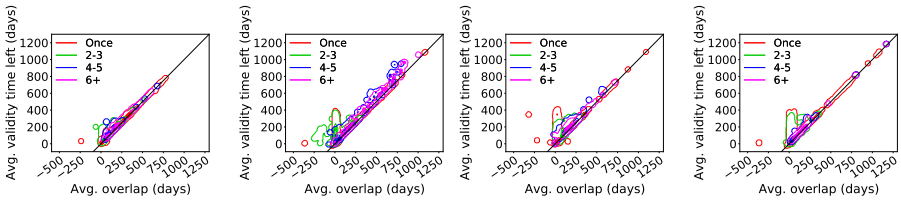
CA-based comparisons: Figure 11 shows the aggregated validity period vs. the aggregated use (over all certificates in the SKCR chains) for the other four CAs with key reuse of at least 12% in 2019. As a reminder, Let's Encrypt and DigiCert had just over 12% SKCRs and the other three CAs shown (Sectigo, GlobalSign, and GoDaddy) all had over 65% SKCRs. (Keys from certificates by Google and cPanel, with ranks 3 and 4, respectively, only were reused in $4.7 \cdot 10^{-6}$ and $1.2 \cdot 10^{-5}$ of their respective CRs.) For the customers of the other CAs we observe much more diverse behaviors. For these other CAs, many chains are also only observed for a small portion of the aggregate validity period (i.e., areas well below the diagonal). While four CAs have clear singularities in the aggregate validity period (e.g., around the 1-year and 2-year marks), their diversity differs substantially. Both DigiCert and GoDaddy primarily appear to have aggregated validity period of a year, suggesting that they limit the reuse and often have significant overlap in their SKCRs. In contrast, both Sectigo and GlobalSign has much more diversity in their SKCR chains (both with regards to aggregate validity period and aggregate usage period). For all four of these later CAs, we again see a shift towards the diagonal as the chains become longer.

Let's Encrypt highly automated: As we have seen, Let's Encrypt's highly automated services stand out in many ways. Another way to highlight this is shown in Figures 12(a) and 13. Here, we compare the replacement timing, measured using the average overlap (based on validity periods) versus the average of the validity left for the first certificate of each SKCR in the SKCR chains



(a) (b)

Fig. 12: Replacement timing for Let's Encrypt (rank #1), measured using (a) the average overlap vs the average validity left for the certificates in CR chains and (b) the average validity vs. use for certificates in CR chains.



(a) DigiCert (#2) (b) Sectigo (#5) (c) GlobalSign (#6) (d) Go Daddy (#7)

Fig. 13: Example CA-based comparison of the replacement timing as measured using the average overlap vs the average validity left for the certificates in certificate-replacement chains. (Rank in parenthesis.)

when the replaced certificate is last observed. Again, Let's Encrypt displays a highly distinct pattern, as all CRs in the SKCR chains are equally treated. In particular, the validity periods are always 90 days and Let's Encrypt appears to aim to use a 30-day overlap, regardless of whether a key is re-used or not.

Replacement certificates typically used close to the time they become valid: For all five CAs, the replacement certificates are typically being seen in use soon after they have become valid. This is seen by almost all points falling along (or slightly above) the diagonal in Figure 13. However, exactly as we observed for the full set of CRs (Figure 6(b)), compared with Let's Encrypt, the overlaps of the other four CAs are much more diverse and are therefore spread much more evenly along the diagonal. Yet, these results clearly show that certificates typically are replaced almost immediately even when there are large overlaps. While this may suggest that the old certificates in some cases are invalidated prematurely (especially in the case they are replaced with a certificate of the same key), it should be noted that websites or CAs may select to change certificates prematurely for other reasons (e.g., to add/remove domains or subdomains that can use the key).

5 Towards short-lived certificates

Motivated by the success of recent automation systems (observed in prior sections), in this section we present a *data-driven case study* that take current trends of the validity periods and reuse of keys (characterized in prior sections) to the extreme. After presenting the problem and highlighting current trends, we first sketch out a solution that combines several new ideas to address the revocation problem that currently leaves all web users vulnerable to man-in-the-middle attacks on most compromised keys long after a certificate has been revoked. Second, we demonstrate the effectiveness of our solution using the different subsets of the CRs identified and characterized in the previous sections as baselines.

5.1 Motivation

Revocation problem: While we have shown that some CAs (e.g., Let's Encrypt, Google, cPanel) mostly issue certificates with roughly 90-day validity periods, even these validity periods can leave users vulnerable to attacks for a long time period. One reason for this is that most browsers (especially mobile browsers) do not sufficiently verify whether an X.509 certificate has been revoked or not [55]. While, as discussed in the introduction, Chrome and Firefox browsers periodically push a by-them-selected set of revocations to their users [41,59], the frequency and size of these revocation sets still leave users of most revoked (leaf) certificates vulnerable long after the compromised certificates have been revoked.

A case for shorter validity periods: One way to address the lack of revocation checks is to use short-lived certificates. This is not new [62,71,76]. One reason this idea has not been widely adopted is due to the lack of automation in past systems (e.g., wide-scale automation was first implemented and deployed by Let's Encrypt [9]), but also due to the significant increase in the number of certificates that would need to be handled.

Current status: We have seen several success stories of automated solutions, including the effectiveness of Let's Encrypt's automated solutions. In addition, our results show that Let's Encrypt allows its customers to effectively reuse the same key over multiple certificates in a resource effective way.

Overhead tradeoffs: There are important security-overhead tradeoffs to consider with short-lived certificates. On one hand, short validity periods reduce the attacker's time window and the potential impact of a compromised key. However, its use also increases the issuance and replacement overheads, and puts much tighter and less flexible timing requirements on certificate replacements.

It is easy to see how automation can help resolve timing issues in the certificate distribution between CAs and their customers. However, there still are significant overheads associated with the subject-key verification during issuance and it is unclear how Certificate Transparency (CT) logs [42,50,69] would handle the increased submission rates resulting from use of short-lived certificates. For example, while splitting a log into several logs may offset the load that a single log would observe, it does not reduce the combined load of the logs. To

provide similar response times, for example, the combined set of resources of such solution would hence still need to scale with the load.

The primary purpose of CT is to provide public immutable records that help detect maliciously or mistakenly issued certificates. Since 2018, Chrome and Apple require all newly issued certificates to be included in CT logs [13,63]. These are public, auditable, append-only logs that at submission return a Signed Certificate Timestamps (SCTs) that the servers can then deliver with their certificates so to prove that the certificate has been logged. However, CT logs do not log revocations and do not protect from misuse of a revoked certificate.

Without new methods to reduce the overheads associated with short-lived certificates, it is unlikely that short-lived, CA-issued certificates with validity periods of one or a few days will see extended use in the near future.

5.2 Parent-child certs: Limiting the cost of short-lived certificates

We next propose a novel approach to address the above tradeoff so as to achieve the advantages of short-lived certificates while keeping the overheads low for CAs. Our approach makes use of three key observations.

First, and most importantly, we note that significant overhead savings can be achieved by decoupling the subject-key verification done by CAs and their issuance of certificates confirming the validity of these pairings. Such decoupling allows CAs to easily create many short-lived certificates that reuse the same key without requiring new domain validation checks. As long as the owner of a key does not report to the CA that the key has been compromised, the CA can continue to generate short-lived certificates with that key.

Second, when using short-lived certificates, it is important to have a fallback mechanism when a certificate is not replaced in time. A key observation here is that the current Online Certificate Status Protocol (OCSP) [39] solutions provide an excellent fallback mechanism (that can be called upon at such instances). OCSP is already implemented by all CAs and the server load is primarily determined by the request volume, not by the number of certificates with tracked status. (Memory and disk to keep track of certificates are not expected to be bottlenecks for an individual CA.)

Third, with today's high CT compliance, the load of CT logs would be proportional to the rate that new certificates are issued. As a naive implementation of short-lived certificates would result in a huge increase in the issuance rate of new certificates (e.g., Figure 15, discussed later in this section), this could result in a very high load also at CT logs. To address this issue, we introduce the concept of *parent* and *child* certificates. This concept would enable the CAs to submit a *parent* certificate to the CT logs as a means to obtain a special SCT that can be used as inclusion proof for all issued *child* certificates that use the same subject-key mapping and for which the validity period t is a subset of the parent's validity period T (i.e., $t \subseteq T$).

The idea of logging a *parent* on behalf of its *child* certificates is inspired by the use of pre-certificates in current CT systems [51]. Pre-certificates are created and logged prior to certificate issuance in order to obtain an SCT for the certificate,

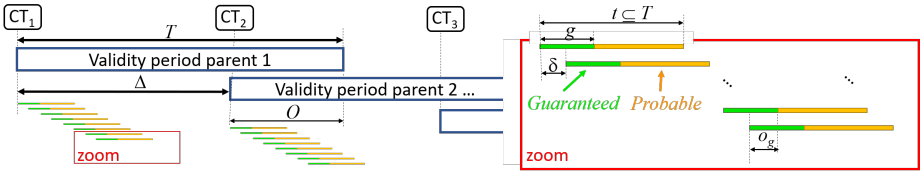


Fig. 14: Parent-child issuance overview. Left-hand side shows *parent* certificates logged in CT. Right-hand side zooms in on a few overlapping *child* certificates.

but include a critical *poison extension* that ensures that it can not be validated by a client. By only logging *parent* certificates, the CAs can help keep the load on CT logs low even when issuing many more *child* certificates. An alternative idea of how to effectively obtain an SCT for a larger set of certificates (in our case a set of child certificates) is to introduce specific log entries into CT logs [36], where several short-lived certificates can be contained in one CT entry. Our solution would work with this approach too.

Motivated by these three key observations, we propose an approach inspired by the general ideas of a three-phased certificate, first presented by Rivest in 1998 [66]. In contrast to regular certificates (as used today), which only have two phases: probable (check revocation status) and expired, Rivest suggested that certificates should have three phases: *guaranteed*, *probable*, *expired*. The idea is to make checking revocation status unnecessary during the first (guaranteed) phase and only check in the second (probable) phase. To combine the first two observations, we extend Rivest's idea to separate the key generation and validation process from the certificate generation process. To do this: (1) *Parent* certificates with a validity period T are issued and submitted to CT logs every Δ days with an overlap of $O{=}T{-}\Delta$ days. (2) *Child* certificates are issued every $\delta \ll \Delta$ days with an overlap of $o_g{=}g{-}\delta$ days in the *guaranteed* phases, where g is the duration of the *guaranteed* phase. (3) Revocations are handled by (i) requiring OCSP checks of *child* certificates after g days and (ii) asking the CA to stop generating/releasing new *child* certificates based on that key. Figure 14 provides an overview of the issuance and logging process.

Note that a client would only need to perform OCSP calls when a child certificate is not replaced with another child certificate within time g or when a certificate actually is compromised. In both cases, the client would perform the OCSP checks as soon as the guaranteed period g has expired. At this time, a client would be informed whether the certificate (and its parent certificate) has been revoked.

Browser discussion: With our solution, the *parent* should not sign the *child*. Instead, the *child* certificate's is expected to use a validity period $t \subseteq T$ that is a subset of its parent's validity period T. Non-CT enabled browsers can treat the certificate independently, while CT-enabled browsers implementing our solution can use the special shared SCTs to validate that the subject-key mapping has been logged in a similar manner as with SCTs based on regular pre-certificates.

Another interesting browser-related aspect is the browser-side usage of OCSP checks. Here, our solution is designed such that OCSP only is used as a fallback mechanism during the *probable* phase. This design choice is motivated by similar reasons (e.g., privacy, performance, etc.) as why Chrome today does not perform OCSP checks [76]. By avoiding the use of OCSP checks for any certificate in the *guaranteed* phase we incentivize CAs and servers to properly manage their certificates so that they always can present a certificate in its *guaranteed* phase. Any performance penalties (which can be sever if enforcing strict OCSP checks), for example, are only endured when a certificate already should have been replaced by a new child certificate. While Chrome currently does not perform OCSP checks, other browsers do. Furthermore, all CAs operate active OCSP servers that provide (mostly) good response rate for status checks of all their issued certificates up to the expiry time of each individual certificate (and beyond) [46]. Implementing such fallback mechanism is therefore expected to be trivial for all browsers. Also, as long as the servers properly maintain their certificates, the browsers should never need to make any OCSP checks.

Parameter discussion: We next briefly discuss the best parameter choices in the context of prior research and best practices. For part of this discussion, we refer to the CA/Browser Forum Baseline Requirements (BR). These BR are shaped in a democratic process of CAs and browser vendors, where both the browser vendors and the CAs have a strong interest in security while keeping costs low. Today, the BR has a central role in the governance of CAs [19]. For example, non-compliance has been used as an argument for root removal [57], and the major root programs require CAs to comply with the BR [70].

The guaranteed period determines the worst-case response time to a revocation. The intention is to allow organizations to choose their own guaranteed period based on their individual risk assessment. However, for the CAs to comply with the BR, they must revoke certificates within 24 hours in some serious cases (e.g., key compromise) and within 5 days for less critical cases [24,26]. Therefore, the revocation mechanism would remain a part of the system. Furthermore, using a *guaranteed* period g of 24 hours is expected to provide as good protection as achieved by a conservative client always performing revocation checks and better protection than the current status-quo of not doing revocation checks. Motivated by OCSP responses being cached for 4 days on average [74], others have suggested that similar guarantees as OCSP can be achieved using certificates with a 4-day validity period [76]. Based on these observations, we foresee that a good selection for the *guaranteed* period g may vary between 1-to-4 days.

Currently, the CA/Browser EV Guidelines suggest that EV certificates should be valid for up to a year [7]. Given this and the measured average frequency that different CAs currently issue certificates, we suggest that new keys are generated, CAs perform re-validation checks of such subject-key mappings, and that the *parent* certificates are submitted to CT logs accordingly (i.e., $\Delta < T$ is less than a year). During this period, new *child* certificates (reusing this key) are then generated every δ days. To ensure overlapping *guaranteed* phases and avoid unnecessary OCSP checks, we suggest using $\delta < g$. Finally, we note that the va-

lidity periods of the *child* certificates can be much longer than g, as long as the browsers commit to OCSP lookups during the *probable* phase.

This approach ensures that domains that always maintain an up-to-date certificate in the *guaranteed* phase can provide services to their clients without any performance penalty. We propose that browsers only penalize the domains that do not provide up-to-date certs (i.e., that are in the *probable* phase).

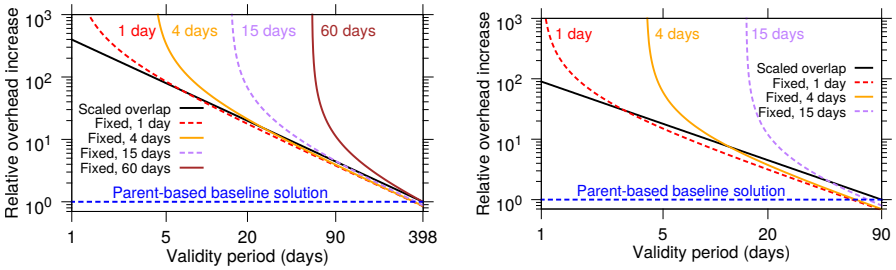
Deployment incentives and challenges: Like past successful changes to the certificate management practices (e.g., CAs becoming CT compliant and 398-day compliant), new solutions must be easy to deploy and/or driven by demand/pressure from users and browser vendors. First, assuming that browsers would demand CA compliance, we believe that our solution easily can be deployed by CAs to meet such expectations. Second, there already is interest in shorter certificate lifetimes. For example, Let's Encrypt publicly expressed interest in shorter lifetimes than the 90-day validity periods used today [8], which was selected to "allow plenty of time for manual renewal if necessary". Third, our solution allows individual CAs to use different parameters and safety margins based on the level of automation that they can provide each customer. Since different safety margins have different security-performance tradeoffs, some CAs are likely to compete based on the level of automation that they can provide. This could drive the demand of good implementations compatible with our framework.

Finally, there are other subtle policy decisions that browsers can do to incentivize CAs and servers to implement and properly maintain up-to-date child certificates. For example, consider again our use of OCSP checks as a potential fallback mechanism during the *probable* phase. While they initially could use a safe-fail policy here (to limit performance implications of slow OCSP responses), it is foreseeable that some may eventually (in the long term) push for strict OCSP checks for any certificate that is not within its (short initial) *guaranteed* phase, regardless of whether it is a child certificate or a regular certificate. This would incentivize servers to both use our *child-parent* approach and to make sure that they always can present an up-to-date child certificate. Ideally all servers would eventually try to maintain up-to-date child certificates and OCSP checks would only be needed when a server fails to do so (of legit or non-legit reasons).

5.3 Data-driven overhead analysis

In this section we examine the overhead associated with different high-level certificate management solutions. For this analysis, we assume that the overhead is proportional to the issuance rate of certificates that require (1) the validation of subject-key mappings and (2) the submission of new certificates to CT logs. Both overheads are important since the subject-key validation process can be both time consuming and costly, and since many CT logs already contain more than a billion certificates and the log sizes are quickly growing [73].

To illustrate the value of *parent* certificates, we present a simple model that captures the relative increase in the number of certificates (parent or traditional) that must be issued for a set of domains when the validity period is reduced.



(a) Basecase: valid 398 days, 60-day overlap (b) Base case: valid 90 days, 30-day overlap

Fig. 15: Relative increase in the number of subject-key validations and CT submissions of selected management policies compared to two baselines.

Model: Consider the set of certificates $\mathcal{N}$ currently used by a large set of servers. Let T and O denote the average validity time and overlap, respectively. Assuming the system is in steady state, we can then use Little's law to obtain the average rate λ that new certificates must be generated as: $\lambda{=}N/(T{-}O)$, where $T{>}O$ and $N{=}|\mathcal{N}|$. The relative increase in the issuance rate can now be calculated as: $(T_{old}{-}O_{old})/(T_{new}{-}O_{new})$, where the subscripts capture change.

Baseline comparisons: Figure 15 illustrates the effect that certificate lifetimes can have on the issuance rate. For this discussion, we normalize all numbers relative to two basic baselines. In particular, we show the relative increase in the number of subject-key validations and CT submissions of when using a few different example management policies relative to the corresponding overhead when using these two baselines, as a function of the selected validity period when using the different example policies. Figure 15(a) shows the relative (multiplicative) increase when we start with an average validity time of 398 days and an average overlap of 60 days, and then reduce the lifetimes in different manners. Figure 15(b) shows the corresponding statistics when we start with an average validity period of 90 days and an average overlap of 30 days. The first default scenario corresponds to changes relative to the most commonly used certificates and overlaps used by Amazon and the second case corresponds to what subjects using Let's Encrypt usually use. In both cases, we include results for the case when the overlap is scaled proportionally to the validity period, the cases when a fixed overlap (e.g., 1, 4, 15, or 60 days) always is used (regardless of validity period), and for our proposed method. Here, we assume that *parent* certificates are issued with a similar frequency and overlap as in the baseline systems. For this solution, the *validity time* (on the x-axis) corresponds to the frequency that *child* certificates are generated (by CA) and used (by servers). Again, the overhead associated with these actions is very small compared to the overhead of validating a subject-key pairing and submitting *parent* certificates to CT logs.

The first scenario (Figure 15(a)) illustrates that a CA reducing its validity period from 398 days to 90 days using normal means would increase its issuance overhead by more than a factor of four (i.e., $> 300\%$), and that further reductions

Table 1: Increase in overhead for different CAs using short-lived certificates without the proposed technique. We show results for different average inter-issuance intervals $\Delta = T_{new} - O_{new}$, measured in days. The columns to the left are based on the CAs' current median values for T_{old} and O_{old}. The columns to the right are based on the observed distributions of T_{old} and O_{old} for each CR of a CA.

	Based on median values						Based on full distribution					
	60	20	10	5	2	1	60	20	10	5	2	1
Let's Encrypt	1.02	3.05	6.10	12.20	30.50	61.00	1.05	3.14	6.28	12.56	31.39	62.78
DigiCert	2.32	6.95	13.90	27.80	69.50	139.00	1.59	4.76	9.52	19.04	47.60	95.19
Google	0.52	1.55	3.10	6.20	15.50	31.00	0.52	1.57	3.15	6.29	15.73	31.45
cPanel	1.27	3.80	7.60	15.20	38.00	76.00	1.35	4.06	8.12	16.23	40.58	81.16
Sectigo/Comodo	2.68	8.05	16.10	32.20	80.50	161.00	2.72	8.16	16.32	32.63	81.59	163.17
GlobalSign	2.32	6.95	13.90	27.80	69.50	139.00	1.87	5.60	11.21	22.42	56.04	112.08
Go Daddy	2.72	8.15	16.30	32.60	81.50	163.00	2.22	6.65	13.30	26.61	66.51	133.03

to a 5-day validity period with a 1-day overlap would increase overhead by a factor of 84.5 (or 8,350%). In contrast, our parent-based solutions can be used with even shorter *guaranteed* periods without increasing the number of subject-key verifications or CT-log submissions. This clearly demonstrates the effectiveness of our approach. When comparing against the second baseline (Figure 15(b)) of 90-day certificates (e.g., currently used by Let's Encrypt) we still see significant reductions in overhead. For example, with a validity period of 5-days and a 1-day overlap, we would see a factor 15 (or 1,400%) difference and for organizations that would want daily certificates the overhead would be 90x (8,900%) higher than the *parent* certificate approach.

Measurement-based CA comparisons: To put the above changes in perspective we first refer back to the CDFs of the most popular CAs validity periods and current overlaps (Figure 6). With all CAs having a median value well above one of these two base cases, our approach would hence consistently result in substantial improvements in overhead compared to the naïve approaches when using short-lived certificates.

We next quantify the improvements for individual CAs. Table 1 shows the relative overhead increases that the top-7 CAs (including the five for which we observed key re-usage) would see when changing to use different example certificate update intervals (listed in the second row and measured in days). This corresponds to $\Delta_{new} = T_{new} - O_{new}$. Here, we calculate the increases in two ways: (1) The columns to the left are based on the CAs' current median values for T_{old} and O_{old}. (2) The columns to the right are based on the actual distributions of T_{old} and O_{old} values, as observed for CRs associated with each CA. Note that the increase is substantial when we get down to update intervals of less than a week. For example, with an update interval of 5 days (e.g., a 7-day certificate with a 2-day overlap) all CAs except Google would need to submit 12-33 times as many certificates to CT logs as they do now, and if updating certificates on a daily basis (potentially still with bigger overlap) the overhead increase would be 61-163 times current loads. The lower overheads for Google are

due to them already using substantially shorter update intervals than the other CAs (e.g., median of (70-39) days compared to (90-29) days for Let's Encrypt).

In comparison, using our approach a CA could easily use the same certificate update interval for their parent certificates as they do now (i.e., $\Delta = T_{old} - O_{old}$), or perhaps more likely even increase it. If they increase the update interval for parent certificates, the improvements would be even greater with our approach than suggested here. For example, we expect updates of parent certificates to be significantly less frequent than the 41 days used by Google on average at the moment. By creating and submitting new using parent certificates less frequently, CAs could hence easily reduce the number of CT submissions and subject-to-key checks they perform at the same time as the lifetime of their child certificates can be reduced substantially.

Finally, there are many validity-overlap pairings that result in the same update intervals. The best overlap is expected to be both website dependent and depend on how strictly browsers would enforce OCSP checks (suggested as a fallback mechanism during the *probable* phase). Differences are also expected between CAs. Again, the relatively bigger overlaps used by Google compared to Let's Encrypt was a contributing factor to their shorter update intervals.

6 Related work

Wide-area certificate scanning: Fast internet-scale certificate measurements using systems and tools such as ZMap [35] and Censys [33] have enabled researchers to quickly scan large IP address spaces to collect and analyze large volumes of certificates. Researchers have studied the certificates collected using such tools (e.g., Rapid7 and Censys) and the certificates found in public CT logs [42, 50, 69] to characterize the certificate landscape [78], analyze how well CAs construct certificates [47], study the popularity of cryptographic libraries [61], label devices [11], and a wide range of other purposes. Others have considered the effect of location [79] or discussed how to best adapt the scanning solutions for the much larger IPv6 address space [60].

Certificate management: Complicated issuance and certificate management processes are believed to have slowed down the original HTTPS deployment [20]. Let's Encrypt addressed many of these issues through the introduction of Certbot and other automated processes [9,75]. However, there are still many issues yet to address, including frequent errors in the CAs' issuance processes [47]. Kumar et al. [47] developed a certificate linter (ZLint), quantified the compliance of the CA/Browser Forum's baseline requirements and RFC 5280 [22]. While there has been a drastic reduction in the fraction of certificates with errors, errors are still frequent [47]. Acer et al. [10] used client-side reports from within Chrome to analyze the main causes of certificate errors, and found that almost all date errors are caused by expired certificates.

Others have proposed extensions to the ACME protocol. For example, Borghol et al. [23] presents a related mitigation technique to better protect against domain takeover attacks for trust-based domain-validation services. Their solution

introduces an additional issuance challenge (for trusted re-issuance) that easily can be solved by domains that are currently in possession of the private key associated with a trusted certificate that has been previously issued for the domain.

Certificate replacements: Most papers on certificate replacement consider the reissuing and revoking of certificates during mass-revocation events related to Heartbleed [34,82] or the case when invalid certificates are replaced by other invalid certificates [29]. These studies suggested that the top sites were quicker at revoking certificates and addressing the Heartbleed vulnerabilities than less popular sites [34] and that sites that did not do this immediately were very slow to do so [34,82]. None of these works considered replacement relationships under normal circumstances, the primary focus in this paper. Mirian [58] finds that popular websites are more likely to be proactive in their certificate renewal than less popular websites. In parallel work, Omolola et al. [64] evaluated how reactive administrators utilizing automation for reissuing certificates were in the event of the Let's Encrypt mass-revocation event (Apr. 2020). They found that 28% successfully reissued their certificates manually within a week—around three times better than the result a week after the Heartbleed bug. They focus on Let's Encrypt certificates found in CT logs and do not consider key reusage or when replacements occur on the servers.

Revocation problems: Browsers have traditionally performed revocation checks using the Online Certificate Status Protocol (OCSP) [39] or Certificate Revocation Lists (CRLs) [22]. However, due to several security, privacy, and performance issues many browser vendors today do not utilize these protocols [28,55].

One area of broad research interest is better ways to revoke certificates. While the goal is the same as OCSP and CRLs, the paths taken differ substantially between solution approaches [30,32,49,72]. Proposals include more efficient push-based protocols and compact forms to convey which certificates have been revoked [49,72] and the caching/sharing of revocation statuses [32]. Others have considered if the world is ready for OCSP Must-Staple and hard-fail policies [30].

Short-lived certificates: Other solutions to the above problem include the use of short-lived certificates [62,71,76], proxy certificates [28,77,80], and the use of different delegation schemes [15,18,21,27,43,52,53]. Conceptually, the idea to use shorter validity periods is simple. Unlike our work, previous works on short-lived certificates did not reuse keys. As we previously noted, just shortening the validity period would result in a big overhead for CAs and CT logs. An interesting alternative way to obtains SCTs for the child certificates may be to combine our idea with that of utilizing special log entries for a collection of short-lived certificates [36]. However, such hybrid scheme may require some extra care in how to best ensure that child certificates are not leaked ahead of time and would still benefit from key reusage and the rest of our proposal.

Both proxy certificates and delegation schemes typically are designed to allow a third party to serve content on behalf of a domain owner without giving them access to the private key of the domain owner. Chuat et al. [28] present

a nice survey and high-level comparison of the above approaches, in which they also make a case for the use of short-lived proxy certificates. While proxy certificates [28] and delegated credentials [15,43] (and similar approaches) help reduce the number of servers that keep long-lived certificates, they do not address the actual problem of speeding up revocations when revocations are needed.

7 Conclusion

This paper first presents a novel server-side characterization of the CR relationships in the wild, including the reuse of public keys. Second, it proposes and demonstrates a simple way to combine parent-child certificate relationships and three-phase certificate handling to reduce the reliance of revocation checks.

Our data-driven CR analysis captures management biases, including the influence that the services offered by different CAs may have on the timing of replacements, safety margins, certificate violations (e.g., early/late usage), and whether the public key is reused. The results highlight a lack of industry standards for replacement policies [38]. Interestingly, the top-CAs using shorter validity periods often also use more common (default) overlaps and their customers achieve more consistent/predictable lifetime characteristics. Having said that, we observe the smallest fraction of gaps and early/late usage for the more expensive (and longer-lived) EV certificates. Another interesting observation is that the three CAs (Sectigo, GlobalSign, Go Daddy) with highest key reuse (>65%) all achieved substantially less gaps when reusing keys than when not reusing keys. While they do not have as high key reuse, Let's Encrypt nicely demonstrates how key-reuse chains can help customers achieve good key utilization.

Finally, motivated by the effectiveness and potential of some of the observed automation solutions and trends, we present a new way to address an important revocation problem currently leaving web users highly vulnerable to man-in-the middle attacks of compromised keys. Our solution takes some current trends to the extreme and combines the use of short-lived three-phase certificates, the introduction of the concept of parent-child certificate relationships, and some simple management rules. The solution addresses the important revocation problem without needing to increase the frequency of subject-key validations and CT log submissions.

Interesting future work includes the collection and analysis of more fine-grained datasets, comparisons with alternative data sources (e.g., CT logs) to obtain a more complete picture of the certificate replacement landscape, performing additional analyses to understand other characteristics (such as relationship of current characteristics to domain popularity), and the implementation and testing of the proposed solution.

Acknowledgment: This work was supported by the Swedish Research Council (VR) and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

1. Marking HTTP As Non-Secure, https://www.chromium.org/Home/chromium-security/marking-http-as-non-secure
2. Project Sonar, https://www.rapid7.com/research/project-sonar/
3. SSL Certificates - Rapid7 Open Data, https://opendata.rapid7.com/sonar.ssl/
4. Scanning All The Things (2013), https://blog.rapid7.com/2013/09/26/internet-wide-probing-rapid7-sonar/
5. Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates (2020), https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.7.0.pdf
6. Chrome Root Program (2020), https://www.chromium.org/Home/chromium-security/root-ca-policy
7. Guidelines For The Issuance And Management Of Extended Validation Certificates (2020), https://cabforum.org/wp-content/uploads/CA-Browser-Forum-EV-Guidelines-v1.7.2.pdf
8. Aas, J.: Why ninety-day lifetimes for certificates? (2015), https://letsencrypt.org/2015/11/09/why-90-days.html
9. Aas, J., et al.: Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web. In: Proc. ACM CCS (2019). https://doi.org/10.1145/3319535.3363192
10. Acer, M.E., Stark, E., Felt, A.P., Fahl, S., Bhargava, R., Dev, B., Braithwaite, M., Sleevi, R., Tabriz, P.: Where the Wild Warnings Are: Root Causes of Chrome HTTPS Certificate Errors. In: Proc. ACM CCS (2017). https://doi.org/10.1145/3133956.3134007
11. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y.: Understanding the Mirai Botnet. In: Proc. USENIX Security (2017)
12. Apple: About upcoming limits on trusted certificates (Mar 2020), https://support.apple.com/en-us/HT211025, last accessed: Jan. 2021
13. Apple: Apple's Certificate Transparency policy (2020), https://support.apple.com/en-us/HT205280, last accessed: Jan. 2020
14. Bano, S., Richter, P., Javed, M., Sundaresan, S., Durumeric, Z., Murdoch, S.J., Mortier, R., Paxson, V.: Scanning the Internet for Liveness. SIGCOMM Comput. Commun. Rev. (2), 2–9 (5 2018). https://doi.org/10.1145/3213232.3213234
15. Barnes, R., Iyengar, S., Sullivan, N., Rescorla, E.: Delegated Credentials for TLS. Internet Draft (June 2020), https://datatracker.ietf.org/doc/draft-ietf-tls-subcerts/09/
16. Barnes, R.: Deprecating Non-Secure HTTP (2015), https://blog.mozilla.org/security/2015/04/30/deprecating-non-secure-http/
17. Barnes, R., Hoffman-Andrews, J., McCarney, D., Kasten, J.: Automatic Certificate Management Environment (ACME). RFC 8555 (3 2019). https://doi.org/10.17487/RFC8555
18. Basin, D., Cremers, C., Kim, T.H.J., Perrig, A., Sasse, R., Szalachowski, P.: Design, Analysis, and Implementation of ARPKI: An Attack-Resilient Public-Key Infrastructure. IEEE Trans. on Dependable and Secure Computing **15**(3), 393–408 (2016). https://doi.org/10.1109/TDSC.2016.2601610
19. Berkowsky, J.A., Hayajneh, T.: Security Issues with Certificate Authorities. In: Proc. IEEE UEMCON (2017). https://doi.org/10.1109/UEMCON.2017.8249081

20. Bernhard, M., Sharman, J., Acemyan, C.Z., Kortum, P., Wallach, D.S., Halderman, J.A.: On the Usability of HTTPS Deployment. In: Proc. CHI (2019). https://doi.org/10.1145/3290605.3300540
21. Bhargavan, K., Boureanu, I., Fouque, P.A., Onete, C., Richard, B.: Content Delivery over TLS: A Cryptographic Analysis of Keyless SSL. In: Proc. IEEE Euro S&P (2017). https://doi.org/10.1109/EuroSP.2017.52
22. Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., Cooper, D.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, RFC Editor (5 2008). https://doi.org/10.17487/RFC5280
23. Borgolte, K., Fiebig, T., Hao, S., Kruegel, C., Vigna, G.: Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates. In: Proc. NDSS (2018). https://doi.org/10.14722/ndss.2018.23327
24. CA/Browser Forum: Ballot SC6 - Revocation timeline extension (2018), https://cabforum.org/2018/09/14/ballot-sc6-revocation-timeline-extension
25. CA/Browser Forum: Ballot SC 31: Browser Alignment (2020), https://cabforum.org/2020/07/16/ballot-sc31-browser-alignment/
26. CA/Browser Forum: Baseline requirements documents (2021), https://cabforum.org/baseline-requirements-documents/
27. Cangialosi, F., Chung, T., Choffnes, D., Levin, D., Maggs, B.M., Mislove, A., Wilson, C.: Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem. In: Proc. ACM CCS (2016). https://doi.org/10.1145/2976749.2978301
28. Chuat, L., Abdou, A., Sasse, R., Sprenger, C., Basin, D., Perrig, A.: SoK: Delegation and Revocation, the Missing Links in the Web's Chain of Trust. In: Proc. IEEE Euro S&P (2020). https://doi.org/10.1109/EuroSP48549.2020.00046
29. Chung, T., Liu, Y., Choffnes, D., Levin, D., Maggs, B.M., Mislove, A., Wilson, C.: Measuring and Applying Invalid SSL Certificates: The Silent Majority. In: Proc. IMC (2016). https://doi.org/10.1145/2987443.2987454
30. Chung, T., Lok, J., Chandrasekaran, B., Choffnes, D., Levin, D., Maggs, B.M., Mislove, A., Rula, J., Sullivan, N., Wilson, C.: Is the Web Ready for OCSP Must-Staple? In: Proc. IMC (2018). https://doi.org/10.1145/3278532.3278543
31. Cimpanu, C.: Apple strong-arms entire CA industry into one-year certificate lifespans (June 2020), https://www.zdnet.com/article/apple-strong-arms-entire-ca-industry-into-one-year-certificate-lifespans/, last accessed: Jan. 2021
32. Dickinson, L., Smith, T., Seamons, K.: Leveraging Locality of Reference for Certificate Revocation. In: Proc. ACSAC (2019). https://doi.org/10.1145/3359789.3359819
33. Durumeric, Z., Adrian, D., Mirian, A., Bailey, M., Halderman, J.A.: A Search Engine Backed by Internet-Wide Scanning. In: Proc. ACM CCS (2015). https://doi.org/10.1145/2810103.2813703
34. Durumeric, Z., Li, F., Kasten, J., Amann, J., Beekman, J., Payer, M., Weaver, N., Adrian, D., Paxson, V., Bailey, M., Halderman, J.A.: The Matter of Heartbleed. In: Proc. IMC (2014). https://doi.org/10.1145/2663716.2663755
35. Durumeric, Z., Wustrow, E., Halderman, J.A.: ZMap: Fast Internet-wide Scanning and Its Security Applications. In: Proc. USENIX Security (2013)
36. Eskandarian, S., Messeri, E., Bonneau, J., Boneh, D.: Certificate Transparency with Privacy. vol. 2017, pp. 329–344 (2017). https://doi.org/10.1515/popets-2017-0052
37. Felt, A.P., Barnes, R., King, A., Palmer, C., Bentzel, C., Tabriz, P.: Measuring HTTPS Adoption on the Web. In: Proc. USENIX Security (2017)

38. Fu, P., Li, Z., Xiong, G., Cao, Z., Kang, C.: SSL/TLS Security Exploration Through X.509 Certificate's Life Cycle Measurement. In: Proc. IEEE. ISCC (2018). https://doi.org/10.1109/ISCC.2018.8538533
39. Galperin, S., Adams, D.C., Myers, M., Ankney, R., Malpani, A.N.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560, RFC Editor (6 1999). https://doi.org/10.17487/RFC2560
40. Google: Certificate Lifetimes (2020), https://chromium.googlesource.com/chromium/src/+/master/net/docs/certificate_lifetimes.md, last accessed: Jan. 2021
41. Google: CRLSets (The Chromium Projects) (2020), https://dev.chromium.org/Home/chromium-security/crlsets
42. Gustafsson, J., Overier, G., Arlitt, M., Carlsson, N.: A First Look at the CT Landscape: Certificate Transparency Logs in Practice. In: Proc. PAM (2017). https://doi.org/10.1007/978-3-319-54328-4_7
43. Guzman, A., Nekritz, K., Iyengar, S.: Delegated credentials: Improving the security of TLS certificates. Facebook blog (Nov 2019), https://engineering.fb.com/2019/11/01/security/delegated-credentials/
44. Josefsson, S., Leonard, S.: Textual Encodings of PKIX, PKCS, and CMS Structures. RFC 7468 (4 2015). https://doi.org/10.17487/RFC7468
45. Korzhitskii, N., Carlsson, N.: Characterizing the Root Landscape of Certificate Transparency Logs. In: Proc. IFIP Networking (2020)
46. Korzhitskii, N., Carlsson, N.: Revocation Statuses on the Internet. In: Proc. PAM (2021). https://doi.org/978-3-030-72582-2_11
47. Kumar, D., Wang, Z., Hyder, M., Dickinson, J., Beck, G., Adrian, D., Mason, J., Durumeric, Z., Halderman, J.A., Bailey, M.: Tracking Certificate Misissuance in the Wild. In: Proc. IEEE S&P (2018). https://doi.org/10.1109/SP.2018.00015
48. Langley, A., et al.: The QUIC Transport Protocol: Design and Internet-Scale Deployment. In: Proc. ACM SIGCOMM (2017). https://doi.org/10.1145/3098822.3098842
49. Larisch, J., Choffnes, D., Levin, D., Maggs, B.M., Mislove, A., Wilson, C.: CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. In: Proc. IEEE S&P (2017). https://doi.org/10.1109/SP.2017.17
50. Laurie, B.: Certificate transparency. Communications of the ACM **57**(10), 40–46 (2014). https://doi.org/10.1145/2659897
51. Laurie, B., Langley, A., Kasper, E.: Certificate Transparency. RFC 6962 (2013). https://doi.org/10.17487/RFC6962
52. Lesniewski-Laas, C., Kaashoek, M.F.: SSL splitting: Securely serving data from untrusted caches. Computer Networks **48**(5), 763–779 (2005). https://doi.org/10.1016/j.comnet.2005.01.006
53. Liang, J., Jiang, J., Duan, H., Li, K., Wan, T., Wu, J.: When HTTPS meets CDN: A case of authentication in delegated service. In: Proc. IEEE S&P (2014). https://doi.org/10.1109/SP.2014.12
54. Linn, J.: Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures. RFC 1421 (2 1993). https://doi.org/10.17487/RFC1421
55. Liu, Y., Tome, W., Zhang, L., Choffnes, D., Levin, D., Maggs, B., Mislove, A., Schulman, A., Wilson, C.: An End-to-End Measurement of Certificate Revocation in the Web's PKI. In: Proc. IMC (2015). https://doi.org/10.1145/2815675.2815685
56. Ma, Z., Austgen, J., Mason, J., Durumeric, Z., Bailey, M.: Tracing your roots: exploring the TLS trust anchor ecosystem. In: Proc. IMC (2021). https://doi.org/10.1145/3487552.3487813

57. Markham, G.: Mailing list: Mozilla dev.sec.policy: Procert decision (2017), https://groups.google.com/g/mozilla.dev.security.policy/c/Ymrpsm7s5_I
58. Mirian, A., Thompson, C., Savage, S., Voelker, G.M., Felt, A.P.: HTTPS Adoption in the Longtail. Tech. rep., Google and UC San Diego (2018), https://research.google/pubs/pub49037/
59. Mozilla: OneCRL (CA/Revocation Checking in Firefox) (2020), https://wiki.mozilla.org/CA:RevocationPlan#OneCRL
60. Murdock, A., Li, F., Bramsen, P., Durumeric, Z., Paxson, V.: Target Generation for Internet-Wide IPv6 Scanning. In: Proc. IMC (2017). https://doi.org/10.1145/3131365.3131405
61. Nemec, M., Klinec, D., Svenda, P., Sekan, P., Matyas, V.: Measuring Popularity of Cryptographic Libraries in Internet-Wide Scans. In: Proc. ACSAC (2017). https://doi.org/10.1145/3134600.3134612
62. Nir, Y., Fossati, T., Sheffer, Y., Eckert, T.: Considerations for using short term certificates. Technical report/Internet-Draft (March 2018), https://datatracker.ietf.org/doc/draft-nir-saag-star/01/
63. O'Brien, D.: Certificate Transparency Enforcement in Chrome and CT Day in London (2018), https://groups.google.com/a/chromium.org/d/msg/ct-policy/Qqr59r6yn1A/2t0bWblZBgAJ, last accessed: Jan. 2020
64. Omolola, O., Roberts, R., Ashiq, I., Chung, T., Levin, D., Mislove, A.: Measurement and Analysis of Automated Certificate Reissuance. In: Proc. PAM (2021). https://doi.org/10.1007/978-3-030-72582-2_10
65. O'Brien, D., Sleevi, R., Whalley, A.: Chrome's Plan to Distrust Symantec Certificates. Google Security Blog (Sept 2017), https://security.googleblog.com/2017/09/chromes-plan-to-distrust-symantec.html
66. Rivest, R.L.: Can we eliminate certificate revocation lists? In: Proc. FC (1998). https://doi.org/10.1007/BFb0055482
67. Sander, R.: What is a Code Signing Certificate? How does it work? IEEE Computer Society (2021), https://www.computer.org/publications/tech-news/trends/what-is-a-code-signing-certificate
68. Schechter, E.: A secure web is here to stay (2018), https://security.googleblog.com/2018/02/a-secure-web-is-here-to-stay.html
69. Scheitle, Q., Gasser, O., Nolte, T., Amann, J., Brent, L., Carle, G., Holz, R., Schmidt, T.C., Wählisch, M.: The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. In: Proc. IMC (2018). https://doi.org/10.1145/3278532.3278562
70. Serrano, N., Hadan, H., Camp, L.J.: A Complete Study of P.K.I. (PKI's Known Incidents)). In: Proc. TPRC47 (2019). https://doi.org/10.2139/ssrn.3425554
71. Sheffer, Y., Lopez, D., de Dios, O.G., Perales, A.P., Fossati, T.: Support for Short-Term, Automatically Renewed (STAR) Certificates in the Automated Certificate Management Environment (ACME). RFC 8739 (Mar 2020). https://doi.org/10.17487/RFC8739
72. Smith, T., Dickinson, L., Seamons, K.: Let's Revoke: Scalable Global Certificate Revocation. In: Proc. NDSS (2020). https://doi.org/10.14722/ndss.2020.24084
73. Spotter, C.: Certificate Transparency Log Growth (2021), https://sslmate.com/labs/ct_growth/, last accessed: Jan. 2021
74. Stark, E., Huang, L.S., Israni, D., Jackson, C., Boneh, D.: The Case for Prefetching and Prevalidating TLS Server Certificates. In: Proc. NDSS (2012)
75. Tiefenau, C., von Zezschwitz, E., Häring, M., Krombholz, K., Smith, M.: A Usability Evaluation of Let's Encrypt and Certbot: Usable Security Done Right. In: Proc. ACM CCS (2019). https://doi.org/10.1145/3319535.3363220

76. Topalovic, E., Saeta, B., Huang, L.S., Jackson, C., Boneh, D.: Towards Short-Lived Certificates. In: Proc. IEEE W2SP (2012)
77. Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820 (2004). https://doi.org/10.17487/RFC3820
78. VanderSloot, B., Amann, J., Bernhard, M., Durumeric, Z., Bailey, M., Halderman, J.A.: Towards a Complete View of the Certificate Ecosystem. In: Proc. IMC (2016). https://doi.org/10.1145/2987443.2987462
79. Wan, G., Izhikevich, L., Adrian, D., Yoshioka, K., Holz, R., Rossow, C., Durumeric, Z.: On the Origin of Scanning: The Impact of Location on Internet-Wide Scans. In: Proc. IMC (2020). https://doi.org/10.1145/3419394.3424214
80. Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S., Siebenlist, F.: X.509 Proxy Certificates for Dynamic Delegation. In: Proc. Annual PKI R&D workshop (2004)
81. Wilson, B.: Reducing TLS Certificate Lifespans to 398 Days. Mozilla Security Blog (July 2020), https://blog.mozilla.org/security/2020/07/09/reducing-tls-certificate-lifespans-to-398-days/, last accessed: Jan. 2021
82. Zhang, L., Choffnes, D., Levin, D., Dumitracs, T., Mislove, A., Schulman, A., Wilson, C.: Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed. In: Proc. IMC (2014). https://doi.org/10.1145/2663716.2663758