Comprehensive Revocation Checking at Scale: the Deployment of CRLite in Mozilla Firefox

Nehal Fooda
University of Maryland
USA

James Larisch
Cloudflare, Inc.
USA

John M. Schanck
Mozilla
USA

Taejoong Chung
Virginia Tech
USA

Dave Levin
University of Maryland
USA

Bruce M. Maggs
Duke University and Emerald Innovations
USA

Christo Wilson
Northeastern University
USA

Abstract

This paper describes the multi-year effort undertaken by Mozilla to incorporate and deploy CRLite, a TLS certificate revocation checking system, in the Firefox browser. With the deprecation of the Online Certificate Status Protocol (OCSP) by Let's Encrypt and others, CRLite is now the only broadly deployed mechanism capable of checking the revocation status of every TLS certificate. The successful seven-year evolution of CRLite from a research prototype to a widely used tool required improvements in the original data structures (now using partitioned Ribbon filters), changes in Certificate Authority revocation practices, and correctly synchronizing with Certificate Transparency logs to eliminate false positives. Using data from Firefox's opt-in telemetry, we report that CRLite achieves an effective revocation coverage of 87.8% while maintaining moderate bandwidth costs, even during real revocation events like the November 2025 Microsoft incident. Through simulations, we also examine the potential impacts of shorter certificate lifetimes and hypothetical mass revocations on CRLite. CRLite demonstrates that low-latency, private, comprehensive certificate revocation checking is possible using moderate bandwidth.

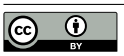
CCS Concepts

• **Networks** → **Application layer protocols**; **Network measurement**; **Security protocols**.

Keywords

TLS, HTTPS, CRL, OCSP, DNS, DNSSEC, Network Measurement, RevDNS, Revocation

ACM Reference Format:
Nehal Fooda, James Larisch, John M. Schanck, Taejoong Chung, Dave Levin, Bruce M. Maggs, and Christo Wilson. 2026. Comprehensive Revocation Checking at Scale: the Deployment of CRLite in Mozilla Firefox. In *ACM SIGCOMM 2026 Conference (SIGCOMM '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3789240.3829108



This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGCOMM '26, Denver, CO, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2467-1/2026/08
https://doi.org/10.1145/3789240.3829108

1 Introduction

The modern Internet depends on a simple promise: when a user connects to a site, the browser should be able to authenticate the endpoint and establish an encrypted channel with the intended party. The web's public key infrastructure (PKI) delivers that promise by issuing and validating cryptographically verifiable chains of trust.

That trust fails in predictable ways. Private keys get stolen, leaked, or mishandled [22, 30, 50]; CAs are sometimes compromised [17]; infrastructure and policy mistakes happen. In the Web PKI, a compromised key can let an attacker impersonate a site without obvious user-visible warning, enabling interception, credential theft, and data exfiltration. Revocation is the mechanism meant to limit this damage: once a certificate should no longer be trusted, relying parties must learn that fact quickly, reliably, and at Internet scale.

In practice, revocation has had a long history of falling short of its stated goal. Measurements show that browsers have often checked revocation incompletely (or not at all) [41]. Even when browsers do distribute revocation information, many deployments rely on curated, partial blocklists that intentionally cover only a small subset of the ecosystem [37]. This is not because revocation is unimportant, but because the traditional mechanisms have forced painful trade-offs: OCSP adds latency and leaks browsing behavior; CRLs can be large and slow to refresh; and soft-fail policies turn routine outages into silent security gaps [21, 41]. Against that backdrop, it is tempting to treat multi-day exposure windows as unavoidable, or to argue that revocation should be abandoned in favor of shorter-lived certificates. From an operator's point of view, that is an attractive simplification; from a security point of view, it is a decision to tolerate compromise for longer than necessary.

CRLite, introduced by Larisch et al. in 2017, proposed a different way to pay the bill [37]. Instead of performing per-connection online checks, CRLite compresses *all* revocation information into a client-side data structure and pushes it to browsers as a background update. Clients then perform revocation checks locally: fast, private, and independent of CA responder availability. The original work suggested that this could be done with an initial download on the

order of 10 MB and daily updates on the order of hundreds of kilobytes.

This paper is an *experience report* on what it took to turn that design into a production, default-on revocation system in Firefox. The core lesson is that the hard part was not proving that a compact filter can represent revocations. *The hard part was operating revocation as an always-on, correctness-first data pipeline at browser scale*: ingesting heterogeneous CA inputs, staying synchronized with a rapidly changing CT ecosystem, meeting tight bandwidth budgets, and never breaking sites due to false positives. Over seven years, Mozilla had to make engineering changes, redesign safety boundaries, and engage directly with the CA ecosystem to improve the quality of the revocation signal itself.

Summary of contributions This paper makes the following contributions:

- We describe how Mozilla operationalized CRLite as a production pipeline that continuously collects certificate and revocation inputs across CAs and refreshes client state on a 12-hour cadence.
- We quantify the practical constraints that dominated deployment, especially bandwidth budgets and the requirement to eliminate false positives, and we document the concrete design changes Mozilla made to meet those constraints.
- We report ecosystem-facing lessons from deploying a system that depends on CA and CT behavior, including where existing PKI metadata and operational practices were not good enough for correctness-first automation.
- Using telemetry from the Firefox desktop fleet, we evaluate CRLite's behavior in the field, including coverage outcomes and the operational gaps that remain under a conservative coverage model.
- Using longitudinal aggregator logs and simulations, we evaluate robustness under large-scale revocation surges and explore forward-looking "what-if" scenarios, including a PKI dominated by shorter-lived certificates.
- Finally, we distill deployment-driven lessons and discuss implications for the future of revocation, including which ecosystem changes would make comprehensive revocation cheaper, safer, and easier to operate at scale.

2 Leading up to CRLite's Deployment

This section provides background on the Web PKI and certificate revocation, introduces CRLite, and summarizes ecosystem changes since the original CRLite paper.

2.1 The Web PKI

The Web PKI enables TLS clients such as web browsers to authenticate TLS servers when establishing encrypted channels. When a browser connects to `example.com` over HTTPS, the server presents an X.509 certificate that binds `example.com` to a public key. Before using that key to establish an encrypted channel, the client must authenticate or validate the certificate. The most important criterion for validity is whether the certificate chain is signed (transitively) by an entity the browser trusts. Other checks include hostname matching, expiry checking, and key usage checking.

In the common case, the server provides the end-entity (or *leaf*) certificate and one or more *intermediate* CA certificates that form a chain from the leaf to a trusted *root* CA certificate in the client's *root store*. The client accepts the certificate only if it can build and verify a signature chain to a root it trusts.[1] Root store policies are set by browsers and operating systems, while many cross-vendor baseline requirements are standardized in the CA/Browser Forum [1].

2.2 Certificate Revocation is a Mess

A certificate can be deemed invalid before its expiry via *revocation*. Revocation is important because it is the only way to alert clients to incorrect or malicious certificates. For instance, if a site's private key is stolen, clients will trust an attacker serving the corresponding certificate until the CA notifies them that the certificate has been revoked.

Unfortunately, all existing revocation mechanisms (before CRLite) had fundamental issues. No solution was *complete* (all revoked certificates covered), *fresh* (updates propagate quickly), *private* (no per-site queries to CAs), *reliable* (no page loads broken by transient CA outages), and *efficient* (low latency and low bandwidth). These issues led some to conclude that "revocation is broken" [32, 34].

We organize these mechanisms along three axes: *push-based* (CRLs, browser blocklists), *online checks* (OCSP), and *expiry-based* (OCSP Stapling, short-lived certificates).

- *CRLs.* A CA publishes a Certificate Revocation List (CRL) containing identifiers of revoked certificates. Clients fetch the issuing CA's signed CRL and check that the target certificate is not listed.
  - **Pros:** Private; the CA does not learn what site a client is visiting.
  - **Cons:** CRLs for high-volume CAs can be large and slow to distribute; clients cache CRLs and can be stale; many browsers fail-open (if they cannot reach the CA, they assume not revoked).
- *OCSP.* The client queries the CA's Online Certificate Status Protocol responder while validating a certificate, asking "Is this certificate revoked" and requiring a "good" response.
  - **Pros:** Low bandwidth per check.
  - **Cons:** Not private; adds latency; can be stale due to caching; and is commonly fail-open in browsers.
- *OCSP Stapling and Must-Staple.* The server fetches an OCSP response and includes it in the handshake. When combined with the OCSP Must-Staple extension, clients can fail-closed (assume revoked) if the staple is missing.
  - **Pros:** Private and typically low-latency for clients; can be fail-closed.
  - **Cons:** Requires server-side support and remains rare in practice [21].
- *Browser blocklists.* Browsers distribute curated lists of high-priority revocations through their own update channels, such as CRLSet [35] and OneCRL [26].
  - **Pros:** Private, fast, and reliable.

[1] We say certificate A *signs* certificate B if B contains a signature that verifies under A's public key.

  - **Cons:** Incomplete by design; only a small fraction of revocations are covered [41].
- *Short-lived certificates.* Although not strictly a revocation mechanism, issuing certificates with very short lifetimes (e.g., 6–47 days) [14] reduces the exposure window after key compromise to the point where revocation may be unnecessary.
  - **Pros:** Compromise exposure window shrinks automatically.
  - **Cons:** Requires automated issuance and renewal; does not help for certificates that are compromised shortly after issuance and still have days of validity remaining.

2.3 CRLite's Evolution

Larisch et al. introduced CRLite in 2017 [37] with a new design that promised completeness, freshness, privacy, reliability, and efficiency. In the initial design, *all* revocations (pulled from CRLs and OCSP) were periodically shipped to all browsers in a compressed set data structure. Whenever the browser encountered a certificate, it would locally (and thus privately) check whether the certificate was in the set or not; if so, the browser rejected the connection. Initially, all revocations were stored in a 10MB data structure with 580KB daily updates. All revocation data was distributed from Mozilla's infrastructure rather than CA-hosted endpoints. Conceptually, CRLite is a more efficient encoding of a CRL: it stores each revocation in a few bits rather than the tens of bytes a serial number occupies in a traditional CRL—small enough to ship the complete revocation set to every client.

CRLite's original design. CRLite initially stored revocations in *Bloom filters* [37]. A Bloom filter is a compressed set whose `member` operation can exhibit false positives—it can return either "no" or "maybe". Internally, it stores its set as a bit vector and uses k hash functions to determine which bits to set to one (on `insert`) and which to check (on `member` checks). The false positive rate p [37] is tunable and inversely proportional to its size. CRLite inserted certificates into Bloom filters using a concatenation of their issuer and serial number.

Because of false positives, naively inserting all revocations into a single Bloom filter cannot work because it will report non-revoked certificates as revoked, which is unacceptable. CRLite's fix was to *discover* all false positives by collecting all non-revoked certificates that appeared to be a member of the first Bloom filter, then constructing a second Bloom filter with all those false positives. The algorithm for checking whether a certificate is revoked then became a two-step routine: first check if in the first filter—if no, it is not revoked; if yes, check if in the second filter—if no, it is revoked; if yes, it is not revoked. However, the second filter can exhibit false positives too. But CRLite applies the same fix: collect all *revoked* certificates that show up as not revoked in the second filter, and construct a *third* filter with those certificates. It then continues this construction, with each filter in the *cascade* getting smaller and smaller, until the observed false positive rate is zero.

CRLite's Bloom filter cascade is only possible because of Certificate Transparency (CT) [2]. The crucial step of enumerating all false positives requires knowledge of all possible certificates a client might encounter; if a revoked certificate was not known at CRLite creation time, the cascade may report that the certificate is not revoked, which is catastrophic. Fortunately, since 2018, Google Chrome has required all certificates to be logged in public Certificate Transparency logs, which allows anyone to reliably construct CRLite filters.

Mozilla began constructing and integrating CRLite into Firefox in 2020 [33]. However, Mozilla encountered challenges with bandwidth usage, reliability, and false positives, and the system was not enabled by default in Firefox desktop until 2025. We describe these challenges in §3.

Clubcard: the evolution of CRLite. Since CRLite's initial design, improvements have been made to the underlying data structure. Mozilla's current deployment replaces the original Bloom filter cascade with a structure built from *two-level cascades of ribbon filters* [46]. Ribbon filters are more efficient than Bloom filters: they use a linear-algebraic membership test and achieve lower space overhead at the same false positive rate. Moreover, ribbon filters only need two such filters while still exhibiting no false positives (and the two-level cascade is within roughly 12% of optimal size). Prior work estimated switching to ribbon filters alone could shrink CRLite's initial download by about 40% [46].

Further space efficiency is gained by *partitioning* the universe of certificates into blocks—e.g., by issuer—and encoding each block within its own two-level ribbon filter cascade. The resulting data structure, called a *Clubcard* [46], produces CRLite filters that are on average 54% smaller than the original Bloom filter cascade while preserving the same lookup semantics [46].

2.4 The Web PKI Has Changed Since CRLite

The Web PKI has changed significantly since CRLite was originally designed in 2017. Several ecosystem shifts have changed both the inputs to CRLite and the constraints it must satisfy.

2.4.1 Certificate Transparency became required. In 2017, CT coverage was incomplete: major browsers did not yet require CT for all publicly trusted TLS certificates. Today, CT is broadly enforced by relying parties (Chrome, Apple platforms, and Firefox), and certificates that lack sufficient CT information are rejected by default [16, 28, 43]. As a result, CT logs now provide a much stronger foundation for certificate enumeration than the original CRLite prototype assumed.

2.4.2 Certificate lifetimes are shrinking. The maximum lifetime of publicly trusted TLS certificates has been reduced from years to months. After Apple announced enforcement of a 398-day maximum, the CA/B Forum aligned Baseline Requirements to cap lifetimes at 398 days starting in September 2020 [15, 23]. More recently, the CA/B Forum adopted a schedule that further reduces maximum lifetime to 200 days starting March 2026, 100 days starting March 2027, and 47 days starting March 2029 [24]. In parallel, short-lived certificates are gaining traction. Let's Encrypt issued its first six-day certificate in 2025 and made six-day certificates generally available in 2026 [14, 40].

Shorter lifetimes change the revocation landscape. They reduce the window of exposure after key compromise, but they also increase issuance volume and update churn, raising the bar for any revocation system that aims to remain complete and fresh at scale.

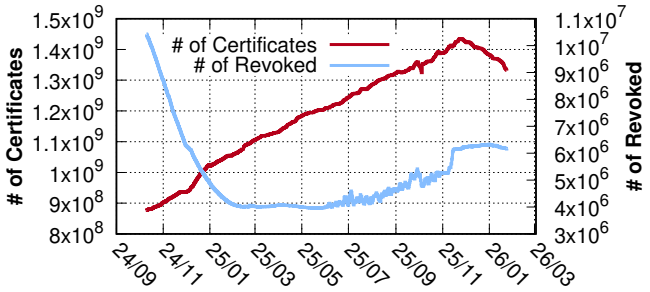


Figure 1: Web PKI growth and revocation volume at Mozilla since 2024.

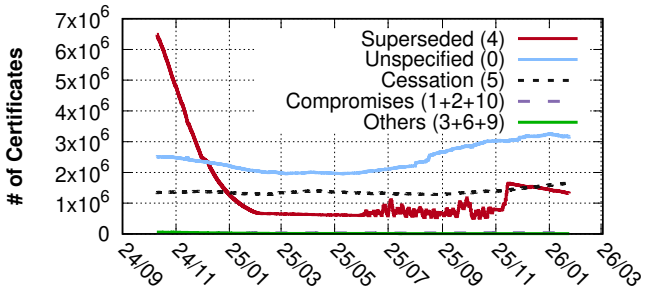


Figure 2: Active revocations by Reason Code over time; the ecosystem is dominated by administrative churn—`Superseded` (the certificate has been re-issued) and `Unspecified` (no reason given)—while security-critical codes such as `KeyCompromise` represent a negligible fraction.

2.4.3 Revocation distribution is shifting back to CRLs. In 2023, the CA/B Forum adopted Ballot SC063v4, making OCSP optional for CAs and requiring CRL publication [20]. Prior to this change, several CAs were only publishing revocation statuses through OCSP. Let's Encrypt began publishing CRLs in 2022, and they turned down their OCSP servers in 2025 citing privacy and operational cost [25, 39].

This shift matters directly for CRLite. A CRLite aggregator needs up-to-date revocation statuses, and regularly querying OCSP responders for millions of certificates is untenable.

Mozilla root store policy has also been updated to require CAs to disclose their CRL distribution points to CCADB. This has made it so that a CRLite aggregator can determine the CRLs that it needs to fetch without scraping CRL distribution points from end-entity certificates.

2.4.4 Mass revocation events occur with increasing frequency. Revocation systems are put to the test during mass revocation events—incidents where millions of certificates are near-simultaneously revoked due to systemic vulnerabilities, operational errors, or CA distrust. We outline major revocation and misissuance-related events in recent history:

- **Vulnerability-driven spikes.** In 2014, the Heartbleed crisis forced the revocation of over 100,000 certificates within 30 days [22]. Traditional revocation delivery mechanisms buckled: some CRLs grew by over 200x, and OCSP responder saturation created a denial of service on revocation infrastructure [22].
- **Operational errors.** In 2016, an operational logic error at GlobalSign unintentionally advertised "revoked" OCSP status for roughly 2.5 million certificates [42].
- **Issuance errors.** In 2020, a software bug in Let's Encrypt's Boulder software necessitated the revocation of around 3 million certificates—roughly 2.6% of their active set—within a 65-hour window [38].
- **CA distrust.** While not directly related to mass revocation, the phased distrust of Symantec (2017–2018) affected millions of certificates across several brands [29]. More recently, in 2024, Entrust's failure to adhere to mandatory 24-hour revocation timelines for mis-issued certificates led to a community-wide decision to distrust their roots [30].

3 CRLite Finally Deployed

While Mozilla ultimately deployed and enabled CRLite in Firefox Desktop (2025), they overcame significant challenges to get there. This section describes CRLite's two major challenges—cost and correctness—and the concrete engineering and ecosystem changes Mozilla made to satisfy them.

3.1 Too Big, Too Expensive

The primary barrier to enabling CRLite broadly was cost, driven by bandwidth. Specifically, how many bytes of CRLite data Mozilla would need to ship to each client or user per month. Around 2020, early sizing suggested roughly 52 MB per user per month. After a major surge of revocations from Google Trust Services in late 2022 and early 2023, the estimate increased to roughly 156 MB per user per month. These numbers were materially larger than those the original 2017 CRLite paper used due to the explosive growth of the Web PKI.

The Challenge: Web PKI growth and bursty revocations. The Web PKI grew tremendously in the past decade. Figure 1 illustrates this trajectory. By mid-2024, Mozilla reported observing [46] over 900M distinct unexpired certificates in transparency logs[2], a roughly 30× increase over the original dataset used in the 2017 CRLite paper. The size of CRLite grew accordingly. The 2017 paper estimated that revocation information could be distributed in a 10 MB filter cascade and 580 KB daily updates. But in 2024, the average full filter size was roughly 18 MB (1.8× the initial estimate) and daily delta updates averaged 800 KB (1.4× the initial estimate). Revocations are also bursty, so a single CA incident can dominate update volume for weeks.

To handle the explosion in volume, Mozilla overhauled three parts of the CRLite pipeline by replacing the multi-level Bloom filter cascades with a more compact data structure, simplifying the incremental update mechanism, and taking measures to reduce the number of relevant revocations in the ecosystem.

[2]Mozilla counts unexpired certificates that chain to a root in Mozilla's root store and considers two certificates distinct if they have a different issuer or a different serial number.

Strategy 1: Optimizing the data structure. The research prototype used a multi-level Bloom filter cascade to encode revocation statuses. Mozilla replaced this with a "two-level cascade of Ribbon filters" that had been described by Mike Hamburg at RWC 2022 [31]. They then showed that one could improve compression efficiency by 1) partitioning the set of revocation statuses, 2) encoding each block of the partition in a separate two-level cascade, and 3) stitching the results together. The resulting design, called *Clubcard* [46], produces CRLite filters that are roughly 54% smaller than the original proposal while preserving the same lookup semantics.

Strategy 2: Compressing delta updates. The original CRLite paper proposed shipping periodic full structures and compact delta updates in between. In the paper, delta updates were binary diffs of the Bloom filters in the cascade (about 580 KB on average in 2017). Mozilla's first deployment did not implement compressed deltas due to the complexity of the proposal. Instead, Firefox shipped a full filter every 10 days and used "stash files" between full updates [6]. Stash files were uncompressed lists of revoked serial numbers grouped by issuer, which offered little benefit over CRLs. Moreover, because a stash only listed revocations, it could not provide a true "non-revoked" signal. A CRLite filter outputs "revoked", "not-revoked", or "unknown", where the last case is used for certificates that were not known to the aggregator. A stash file does not distinguish between "not revoked" and "unknown". As a result, a client using a few-day-old filter could be fully up to date on stash downloads and still not be confident about the validity of a certificate issued after the filter build time. Mozilla's current CRLite deployment uses the clubcard data structure for delta updates, which provides good compression and a reliable signal of non-revocation.

Strategy 3: Filtering out irrelevant revocations. Mozilla cannot control how many certificates are issued or revoked, but they can control which revocation entries it enters into CRLite. They take three steps here. First, Mozilla discards revocation entries that cannot be matched to a certificate observed in Certificate Transparency (currently 1.6M certificates). The original CRLite paper counted all serial numbers in all CRLs, but in practice, unmatched entries are likely expired or internal certificates, and keeping them increases client cost for no benefit.

Second, Mozilla has encouraged CAs to not needlessly revoke certificates. Figure 2 details the cumulative volume of revocations collected by Mozilla, categorized by *reason code* since September 2024; it shows that the vast majority of revocations are driven by *administrative churn* rather than *security incidents*. Notice the sharp decline in `Superseded` revocations (red line in Figure 2). Mozilla's analysis revealed that a single major CA (also a hosting provider) was responsible for approximately 67% of all revocations in the Web PKI at the time. These revocations were triggered automatically for "parked" domains that had been inactive for 90 days—a policy that generated massive churn with zero security benefit. Mozilla presented this data at a CA/Browser Forum meeting, raising the disproportionate bandwidth cost imposed on the ecosystem. Following this engagement, the CA agreed to stop revoking certificates solely for inactivity, resulting in the immediate and sustained reduction in revocation volume visible in the plot.

Finally, Mozilla is pushing to make reason codes a more reliable and information-dense signal. Notice that the dataset remains dominated by `Unspecified` (reason code 0) revocations. In contrast, actual security compromises (codes 1, 2, and 10)—the primary threat model for revocation checking—remain a negligible fraction of the total volume. Ideally, CRLite should only include these security-critical revocations, but is forced to include all revocations. To address this, Mozilla started requiring CAs to (almost) always include a reason code [49] and is advocating for a dedicated "administrative" reason code that the aggregator can explicitly ignore. Mozilla also publishes CRLite filters that only include revocations with high-risk revocation reason codes, such as key compromise, to minimize size. These smaller filters are currently used by default on Firefox for Android.

3.2 False Positives

The second deployment constraint was correctness. During early deployments, CRLite sometimes classified non-revoked certificates as revoked, forcing Mozilla to halt rollouts. These failures were not due to the probabilistic nature of the data structure. They were due to confusion about which certificates had been seen by the CRLite aggregator.

Coverage metadata. A filter *covers* a certificate if the aggregator included a revocation status for that certificate in the filter. The data structure used in CRLite always gives a correct result for a covered certificate, but it can err on other certificates. Thus, a CRLite filter must be published with *coverage metadata* that perfectly describes the set of covered certificates. This metadata must allow the client to answer definitively, and for any certificate, *"Does this filter cover this certificate?"*

Issuance based coverage. Section IV.D of the 2017 paper suggests that clients can avoid false positives by checking

- that the `notBefore` date of the certificate is before some cutoff, and
- that the certificate chains to a given set of roots.

The coverage metadata is thus a timestamp and a list of root certificates. This turned out to be insufficient. Recall that the CRLite aggregator discovers certificates in Certificate Transparency logs, and that the `notBefore` date is merely an issuance claim—it is not proof that the certificate was logged in CT. Since the certificates in a log are not ordered by `notBefore` date, it is possible that a new certificate with a `notBefore` date before the cutoff will be added to the log after the cutoff has been chosen. In this scenario, the client believes that this certificate is covered. Since the aggregator never saw the certificate, it is absent from the filter, and the data structure may give an erroneous result.

Transparency based coverage. Upon discovering false positives in their CRLite implementation with issuance-based coverage, Mozilla switched to a notion of Signed Certificate Timestamp (SCT)-based coverage [3]. When a pre-certificate is submitted to a transparency log, the log returns an SCT, which may be embedded in the final certificate as evidence of logging. In the revised system, the client avoids false positive by checking

- that the certificate is presented with an SCT timestamp before some cutoff, and
- that the certificate chains to a given set of roots.

The coverage metadata is still a timestamp and a list of root certificates.

Unfortunately, SCT-based coverage was also insufficient. The coverage metadata did not specify the set of logs that the aggregator observed. Hence a certificate could pass the coverage check with an SCT from a log that was not monitored by the CRLite aggregator at all. The choice of cutoff also failed to account for each log's Maximum Merge Delay (MMD). An SCT is evidence of logging, but not proof of logging. It is a promise that a log will include a certificate within the next MMD period (typically 24 hours). This creates a dangerous window in which a certificate can carry a valid timestamp yet not be visible in the corresponding log. Once again, this choice of metadata led clients to apply CRLite to certificates that were not covered, which led to false positives [4, 5].

A workable solution. To address all the above problems, Mozilla switched to a notion of coverage based on per-log timestamp intervals [7]. The coverage metadata includes a hashmap that associates CT log identifiers to coverage intervals. The coverage intervals are given as [`start`, `end`] where

- `start` is the earliest timestamp that the aggregator observed in the log plus the MMD of the log, and
- `end` is the latest timestamp that the aggregator observed in the log minus the MMD.

An interval, rather than a single timestamp, is used so that the aggregator can consume logs in reverse order, which prioritizes new certificates that are less likely to be expired. This is important because some CT log operators impose severe rate limits that make it difficult to ingest log entries as quickly as they are created.

With this metadata, the client can avoid false positives by checking

- that the certificate was presented with an SCT timestamp in the appropriate coverage interval,
- that the certificate chains to a given set of roots.

This eliminates false positives so long as the monitored CT logs adhere to their promise to log certificates within the maximum merge delay. In a future iteration of the system, Mozilla plans to define coverage based on log index intervals rather than timestamp intervals. The log index is not available in SCTs from RFC 6962 logs, but it is encoded in the `leaf index` extension of SCTs from static CT logs [47]. Switching to log index intervals will eliminate the risk of false positives due to MMD violations. The only remaining source of false positives will then be from logs that provide an inconsistent view to the aggregator.

4 CRLite in Production

In this section, we describe the design, implementation, and deployment efforts Mozilla has embarked on to deploy Clubcard-powered CRLite [45, 46] in practice. As of the time of this writing, Mozilla Firefox's desktop and mobile clients use CRLite to check the revocation status of all certificates on the web. Desktop clients are capable of detecting all revocations, whereas mobile clients can only detect a smaller number of high priority revocations, as described above.

4.1 Aggregator design

Mozilla's CRLite aggregator runs on a single Google Compute Engine n2-highmem-16 machine. Its role is to ingest certificates from CT logs, fetch CRLs from CAs, and publish CRLite filters.

CT log ingestion. A continuously running `ct-fetch` program reads entries from CT logs. This program periodically updates the list of monitored logs from a Firefox remote settings collection. Firefox engineers can control which CT logs are monitored by CRLite, but by default any log that appears in Google's v3 log list [27] is monitored.

The `ct-fetch` program has one worker thread per CT log. That worker reads log entries and extracts

(1) The hour of the `notAfter` date of the certificate.

(2) The issuer of the certificate.

(3) The serial number of the certificate.

For each log entry, a tuple containing the serial number, the CT log identifier, and the log entry timestamp is stored in Redis in a set that is keyed by the expiry hour of the certificate concatenated with the SHA256 hash of the issuer certificate's subject public key info. Log workers queue writes to the Redis cache on a single thread. The Redis cache is also used to store coverage metadata including, for each CT log, the log identifier, its MMD, the range of entries that the aggregator downloaded, and the smallest and largest timestamps that the aggregator observed. Log workers periodically queue updates to the metadata after reading a (configurable) number of log entries. If the `ct-fetch` process is interrupted, the metadata is used to resume downloading the log from the last checkpoint.

Periodic commits to disk. Once per hour, certificate data and coverage metadata is moved from the Redis cache to persistent storage in a process that we call a *commit*. The process is carefully designed to ensure that the set of certificates on disk matches the set of certificates claimed by the coverage metadata.

At the beginning of the process, the aggregator reads coverage metadata from the Redis cache and writes it to disk. The aggregator then iterates over the (expiry hour, issuer SPKI hash) labeled bins of (serial, log, timestamp) tuples in the Redis cache. It uses the on-disk coverage metadata to filter out tuples that correspond to certificates that are not covered. The serial numbers from the remaining tuples, those corresponding to covered certificates, are written out to files and removed from the Redis cache; there is one file for each (expiry hour, issuer SPKI hash) bin. Finally the aggregator removes the files for any (expiry hour, issuer SPKI hash) bins that have expired.

The commit process runs in parallel with the `ct-fetch` process, and does not stall CT log ingestion. The on-disk copy of the coverage metadata that is written during a commit can also be used to restart the `ct-fetch` process in the event that the data in the Redis cache is lost.

Filter generation. Once every 12 hours the aggregator fetches the full list of CRLs disclosed by CAs from a report in the Common CA Database (CCADB)—the central repository used by root store operators to track Certificate Authority (CA) identities, root store inclusion, and revocation endpoint locations. The CRLs are downloaded to disk. The CRLs persist on disk between runs, so if a CRL fails to download during one run the most recently downloaded

CRL will be used to determine revocation statuses. The aggregator then generates several Clubcards (CRLite data structures powered by Ribbon filters and partitioned).

(1) A full snapshot: a Clubcard that encodes $R \subseteq U$ where U is the set of non-expired and covered certificates and R is the subset of those certificates that appear in CRLs.

(2) A delta update: a Clubcard that encodes $R' \subseteq U$ where U is the set of non-expired and covered certificates and R' is the subset of those certificates that have recently appeared in CRLs.

The set R is stored on disk and made available to the next run of the filter generation process, and the previous run's R is used to determine R'.

The aggregator can apply additional restrictions on the set R. For instance, the current deployment generates a full snapshot and delta update in which R consists only of high priority revocations—those with the `keyCompromise`, `cessationOfOperation`, and `privilegeWithdrawn` reason codes.

Filter publication. The Clubcards generated in the filter generation step are published through a Firefox remote settings collection. Each record in the collection has a `channel` field. This channel allows Firefox clients to switch between different sets of Clubcards. At present there is one channel which publishes Clubcards that encode all revocations, and one channel that publishes Clubcards that encode only high priority revocations. Desktop users are subscribed to the all revocations channel by default. Mobile users are subscribed to the high-priority channel by default.

Updates to the remote settings collection are pushed to Firefox clients immediately. Users who are not online during an update poll the remote settings server shortly after the browser is started.

In the current configuration, Clubcards are published every 12 hours. At all times, a channel contains one full snapshot and a series of delta updates. Every 45 days the collection of delta updates is cleared, and a new full snapshot is published.

In principle, there is no reason to publish full snapshots. This is because certificates in the WebPKI have bounded validity periods. Once all of the certificates that were encoded in a Clubcard have expired, it can be discarded. So a full snapshot is only needed until the sequence of delta updates is long enough to contain data on all non-expired certificates. Beyond that point, one can always remove the oldest clubcard from the collection when a new delta update is published, and keep a constant sized collection.

Publishing a full snapshot every 45 days, as Mozilla currently does, incurs greater bandwidth costs than necessary. However it ensures that clients never need to perform more than 90 Clubcard queries per revocation check.

In a future revision of Mozilla's CRLite instantiation, when the maximum validity period for a certificate in the WebPKI is on the order of 45 days, they will dispense with full snapshots and publish only delta updates.

5 Deployment

Mozilla's deployment of CRLite was not a singular event, but a six-year iterative process of refining data structures, redefining safety boundaries, and carefully ramping up enforcement. Transforming a research prototype into a default browser mechanism required solving the twin challenges of *cost* and *correctness* detailed in §3. This section details the evolution of CRLite through five distinct phases of deployment, summarized in Table 1.

5.1 Phase I: Infrastructure and Early Stumbles (2019–2020)

Mozilla began operating a production CRLite aggregator and publishing filters in mid-2019. Client-side support initially landed in Firefox 70 (October 2019). The initial deployment strategy focused on the bandwidth constraints described in §3. In Firefox 78 (June 2020), we introduced "stash files"—uncompressed lists of revocations distributed between full filter updates. This allowed us to maintain freshness via small, frequent updates without forcing clients to redownload the entire filter daily.

However, the initial definition of "coverage"—the logic determining which certificates can be safely checked against the filter—proved insufficient. As analyzed in §3 (*Root Cause 1*), early implementations relied on certificate issuance dates (NotBefore). We discovered that because Certificate Transparency (CT) logs are not strictly ordered by issuance date, a certificate could be issued *before* a filter's cutoff but logged *after* the aggregator had built the filter. This discrepancy created the consistency gap described in §3, where valid certificates were incorrectly deemed revoked because they were missing from the filter despite meeting the issuance cutoff.

5.2 Phase II: The Coverage Crisis (2020–2022)

To address these gaps, we fundamentally changed how clients determined filter applicability. In Firefox 83 (November 2020), we switched to a Signed Certificate Timestamp (SCT)-based definition of coverage, reasoning that the SCT timestamp is a more reliable indicator of log ingestion than the certificate's internal dates.

Confident in this new heuristic, we enabled CRLite in Firefox 84 (December 2020) on the Nightly channel. This rollout was short-lived. Users immediately reported false positives caused by the *Maximum Merge Delay* (MMD) of CT logs—precisely the "Merge Delay Trap" identified in §3. We realized that an SCT is merely a promise to log within a specific window (typically 24 hours). A certificate could possess a valid SCT timestamp t but not yet be visible to the aggregator at time $t + \delta$. Consequently, the client believed the certificate was covered (and thus revoked, as it was absent from the filter), while the aggregator had simply not yet seen it.

We reverted CRLite to a telemetry-only mode in Firefox 86 (January 2021) and spent the next year re-engineering the coverage logic. The solution, deployed in Firefox 98 (March 2022), was to implement the *Explicit Coverage Intervals* described in §3. This explicitly accounted for the MMD of each log, ensuring the client only consulted CRLite if the certificate fell within a timestamp range the aggregator had definitively ingested.

5.3 Phase III: The Hybrid "Confirm Revocations" Mode (2022–2024)

With a stable definition of coverage, we began a cautious rollout to regain confidence. In Firefox 100 (May 2022), CRLite was set to "Confirm Revocations" mode.

Date	Milestone
Oct 2019	Client support added (Firefox 70)
Jan 2021	Reverted to telemetry-only (Coverage redesign)
May 2022	Hybrid "Confirm Revocations" mode (Firefox 100)
Oct 2024	Clubcard data structure support (Firefox 132)
Aug 2025	**Full Enforcement Mode on Desktop** (Firefox 142)
Nov 2025	Enforcement Mode on Mobile (High-Priority Channel)

Table 1: Key milestones in the multi-year deployment timeline of CRLite.

In this configuration, CRLite acted as an authoritative source for *validity* but not for *revocation*.

- If the filter returned "not revoked," the browser proceeded immediately, eliminating the privacy and latency costs of an OCSP query.
- If the filter returned "revoked," the browser treated it as a hint and performed a standard OCSP query to confirm.

This hybrid mode allowed us to validate the filter's accuracy at scale without risking false positives (the risk of false negatives was considered sufficiently small because it was already much lower than the failure rate of OCSP for revoked certificates). During this period, we also improved cryptographic agility, replacing the MurmurHash3 algorithm in April 2022 to strengthen collision resistance.

5.4 Phase IV: Clubcards and Enforcement (2024–2025)

As the Web PKI grew (see Figure 1), the original Bloom filter data structures became unwieldy, exacerbating the cost challenges from §3. To support the sheer volume of certificates, we implemented *Strategy 2* (§3) by integrating support for "Clubcard" data structures in Firefox 132 (October 2024), followed by Clubcard-based delta updates in Firefox 133 (November 2024).

This era also introduced *Filter Channels* (Firefox 128.2, September 2024), utilizing the *Semantic Filtering* strategy (§3) to ship different datasets to different device classes. Desktop clients continued to receive the full dataset, while mobile clients were subscribed to a "High-Priority" channel containing only high-risk revocations (e.g., key compromises).

The final push to enforcement revealed subtle edge cases. In March 2025, we discovered and fixed false positives caused by the improper handling of CT log entries involving precertificate signing certificates. Following this fix, we enabled **Enforcement Mode**—where CRLite is the sole authority and OCSP is disabled—in Nightly builds of Firefox 135 (February 2025).

5.5 Phase V: General Availability (2025)

The widespread deployment followed a staged safety release:

(1) *Early Beta (April 2025, Firefox 137):* CRLite enforcement was enabled for beta users to catch regression bugs.

(2) *Desktop Release (April 2025, Firefox 137):* Filter downloads were enabled by default. However, Release builds remained in a "Double Check" configuration: treating CRLite "revoked" responses as authoritative only if confirmed by OCSP, but deferring to OCSP if results differed.

(3) *Full Enforcement (August 2025, Firefox 142):* We switched Release builds to full enforcement mode. For covered certificates, Firefox 142+ no longer sends OCSP queries, relying entirely on the local CRLite store.

Post-deployment monitoring revealed two final classes of logic errors. In September 2025, we fixed a false negative bug where non-covered revoked certificates were improperly passed to the filter generation process. Our program for constructing delta updates would mistakenly treat these certificates as if their revoked status had already been published even though clients saw their status as not-covered. In October 2025, we corrected an MMD offset calculation error that caused false positives on some newly issued certificates. Additionally, in Firefox 144 (October 2025), we reordered the validation pipeline to perform CT signature checks *before* revocation checking, ensuring that invalid SCTs could not trigger false revocation states.

Finally, in Firefox 145 (November 2025), we enabled the high-priority filter channel by default on Mobile platforms, bringing enforcement mode to the mobile fleet for the first time.

6 Evaluation

We evaluate CRLite along four axes that matter for a production revocation system. First, *real-world efficacy*: for every certificate validation attempt made by a Firefox user in the wild, what are the chances Firefox covers that certificate. Second, *operational robustness*: how the update pipeline behaves under large revocation surges observed in production. Third, *future scaling*: how dissemination and storage costs change under plausible PKI shifts such as shorter certificate lifetimes. Fourth, *client-side cost*: whether the per-validation CPU and memory overhead is small enough for a browser environment. Our analysis combines (i) opt-in Firefox desktop telemetry over a 90-day window ending in January 2026, (ii) longitudinal logs from the CRLite aggregator (snapshot and delta sizes), (iii) simulations over Certificate Transparency (CT) log data aggregated from Cloudflare-monitored CT logs, and (iv) microbenchmarks over the current production filter set.

6.1 Coverage and gaps in production

Real-world efficacy is measured through an analysis of 90 days of telemetry ending in January 2026. Every certificate validation attempt is classified into one of three categories: *covered*, *not covered*, or *no filter*.

- *Effective Coverage (87.63%):* This metric represents the overall security posture of the deployed system. While the aggregator's coverage for known certificates is higher, the total effective coverage across all verification attempts—including those made by clients that have not yet synchronized a filter—averages 87.63% (Figure 3). The complement of effective coverage is the sum of two gaps in coverage: the *bootstrap gap* and the *aggregator gap* (Figure 4). These gaps are not simply implementation bugs; they are the practical costs of operating a correctness-first, conservative coverage model.

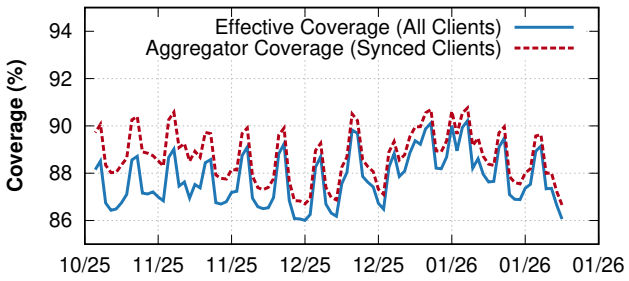


Figure 3: Effective CRLite coverage rate over a 90-day observation period. The effective coverage is stable, with an attempt-weighted mean of 87.63% (and an unweighted daily mean of 87.80%).

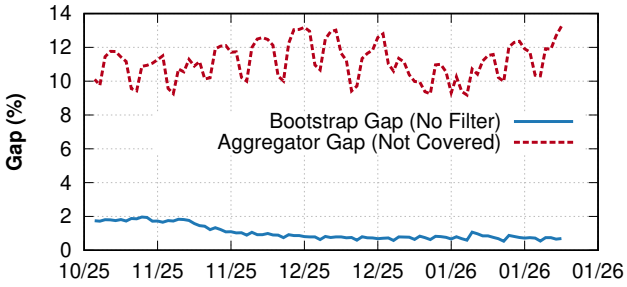


Figure 4: Analysis of security gaps in CRLite coverage. The aggregator gap (certificates issued between updates) remains the dominant factor, while the bootstrap gap (clients without filters) is minimal.

- *Bootstrap gap (1.05%):* The "no filter" state occurs when a client does not have a usable local filter, typically due to a new installation, profile reset, or an extended offline period. The low incidence rate indicates that the dissemination mechanism reaches nearly the entire active desktop fleet in the steady state. From an engineering standpoint, this is also a sanity check: if the bootstrap were large, improvements in filter quality would not matter because the fleet would not be receiving them.
- *Aggregator gap (11.32%):* These are validations where the client had a filter, but the certificate was not within that filter's coverage. There are several concrete causes:
  - *Update window:* the certificate was issued after the client's most recent filter update.
  - *Merge-delay exclusion:* the certificate was issued near the edge of what the aggregator can prove it has fully observed, due to CT log merge-delay behavior.
  - *Ingest lag:* the aggregator had not yet ingested the relevant CT log entries when the filter was generated.

  The first cause is inherent to any periodic update mechanism: shorter update intervals reduce this gap but increase aggregation and distribution work. The latter two causes are consequences of correctness-first coverage semantics. They exist because Firefox must never apply CRLite to certificates that the aggregator might not have fully observed (§3). As CT evolves from RFC 6962 logs toward static CT logs, we expect merge-delay driven uncertainty to shrink in importance. Unlike RFC 6962 logs, which rely on a 24-hour Maximum Merge Delay (MMD) to integrate certificates, static CT logs eliminate MMD by batching certificates upon submission and providing leaf index extensions within SCTs. As discussed in §3, this allows for index-based reasoning, enabling tighter and more definitive coverage definitions that eliminate the risk of false positives from MMD violations. However, the update-window component remains unless the system increases update frequency.

6.2 Robustness under a real mass-revocation campaign

To stress-test the production pipeline under real inputs, we analyze the 2025 Microsoft mass-revocation campaign [8]. After a policy non-compliance discovery involving more than 77 million certificates, Microsoft revoked certificates in batches weekly over approximately six months (May to November 2025) rather than in a single day, to avoid extreme CRL growth. This strategy prioritized revocations based on telemetry identifying whether a certificate was actually "in use," while utilizing the natural expiration of the remaining population—over 90% of which was reported as no longer in use.

Figure 5 shows the production impact on CRLite. Two curves exhibit a permanent "step-up" as the campaign progresses: the number of active revocations and the total filter size. In contrast, the dissemination cost that matters most to end users, the delta update size, remains a series of transient spikes that stay well below 500 KB. The peak delta update size is approximately 400 KB (August 2025), even during a campaign that increased active revocations from roughly 4.0 M to 6.1 M.

The deployment takeaway is not that revocation volume can spike. That has been true for decades. The takeaway is that Clubcard plus delta updates shifts the operational burden away from persistent, large client downloads and toward short-lived dissemination spikes, which are far easier to absorb in a push-based update channel than issuer-specific CRLs.

6.3 Future Scalability: The "What-If" Analysis

To understand the limits of CRLite, we simulate the PKI under extreme stress. We evaluate how the system performs when two major future trends collide: a massive revocation event (e.g., Heartbleed) occurring in a world dominated by short-lived certificates. To do this, we use certificates aggregated from Cloudflare-monitored CT logs from July 3rd to November 3rd 2025; for simplicity we use the `notBefore` date for determining when each certificate is covered by CRLite. Note the data double-counts cross-signed certificates (0.78% of total) due to how the CT entries were stored in the database.

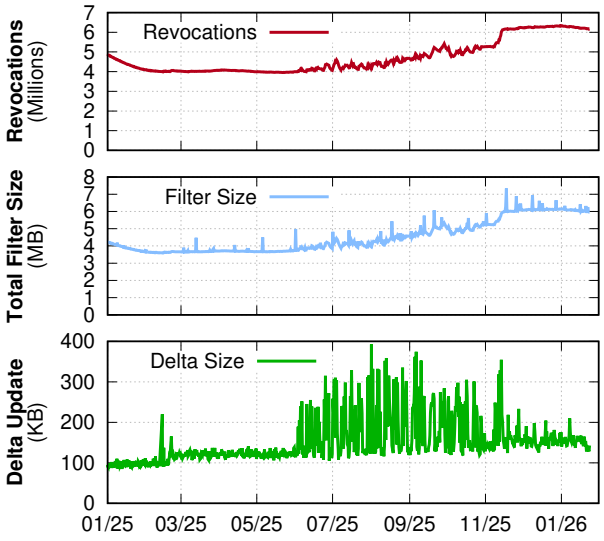


Figure 5: CRLite production metrics during the 2025 Microsoft mass-revocation event. While the revocation count (top) and total filter size (middle) underwent a permanent "step-up" in scale as certificates were processed, the delta update bandwidth (bottom) remained a series of transient, sub-megabyte spikes.

6.3.1 Experimental Setup and Definitions. The Clubcard architecture is subjected to a "Stress Test" scenario combining two variables:

- *Mass Revocation:* A simulated event where 50% of all valid certificates across all CAs are revoked over a 5-day period, as in the case of a widespread software vulnerability in a common library like OpenSSL. It is modeled after post-Heartbleed decay rates [50].
- *Short-lived Certificates:* We model a high-churn environment where all certificates have 6-day lifetimes *Short-lived*, consistent with industry trends [14], compared against a *Baseline* of the real-world distribution (using original lifetimes as recorded in CT logs).

6.3.2 Impact on Dissemination Bandwidth (Deltas). First, we analyze the daily bandwidth cost for active clients (Delta Updates). Figure 6 compares the delta update sizes across both certificate lifetime scenarios under the stress test.

As shown in Figure 6, both the Baseline (Real-World) and Short-lived (6-day) scenarios experience a significant bandwidth spike, reaching approximately (≈117 MB) during the height of the 50% mass-revocation event. However, the 6-day Short-lived scenario demonstrates a steep decay curve compared to the Baseline. Because these certificates expire in less than a week, the revoked entries are purged from the aggregator's valid set and the resulting Clubcard filter almost immediately following the conclusion of the 3-day revocation surge. Ideally, for mobile deployment, these peaks suggest that mass revocations will require auxiliary mechanisms (e.g., throttled rollout) regardless of the compression algorithm.

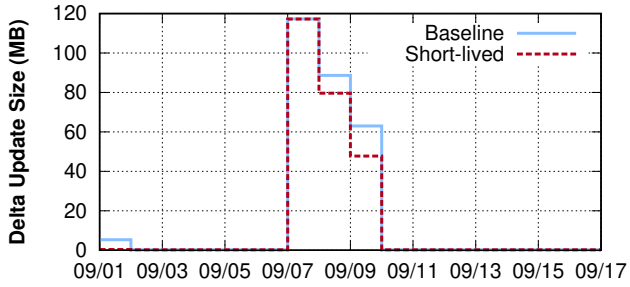


Figure 6: Delta Update Sizes during a 50% mass revocation event. The Short-lived (6-day) scenario returns to the bandwidth baseline faster than the Baseline, as revoked entries are pruned upon expiration.

6.3.3 Impact on Storage and Bootstrap Cost (Full Size). While Deltas affect daily traffic, the *Total Filter Size* determines the storage footprint on disk and the "Bootstrap Cost" for new or returning users. Figure 7 illustrates the most critical divergence between the two lifetime scenarios.

Under Baseline conditions, the filter undergoes a "spike and stay" pattern. As shown in Figure 7, the filter grows to accommodate the 50% surge and remains bloated at approximately (≈134 MB). This persistent inflation represents a significant burden for any new user attempting to bootstrap a filter during the recovery phase. In contrast, the Short-lived (6-day) scenario reveals the decisive advantage of pairing the Clubcard architecture with high-churn certificate ecosystems. The filter exhibits a clear "spike and drop" behavior. Because certificates expire within six days, revoked entries are automatically pruned from the filter shortly after the surge concludes. This "Self-Cleaning" property ensures that the system reclaims its original lightweight footprint (≈0.4 MB) within a week of the crisis.

6.3.4 Offline client break-even analysis. An operational question that only appears at scale is how to resynchronize clients that have missed many updates. In the current client behavior, an offline client retrieves the full sequence of missing delta updates. At some point, the cumulative deltas exceed the cost of downloading a fresh snapshot.

Using production logs and the issuer + expiry partitioning lower bound, we estimate the break-even threshold. Let δ be the mean size of a single 12-hour delta update, and let F be the size of a fresh full filter snapshot. From the current production filter set we use:

- *Delta update size (δ):* approximately 200 KB (current production size of one 12-hour update).
- *Full filter size (F):* approximately 6 MB (current production snapshot).

The break-even point in update cycles is $T = F/\delta$:

$$T = \frac{6\ \text{MB}}{200\ \text{KB}} = 30\ \text{cycles}. \quad (1)$$

Each cycle is 12 hours, so T corresponds to approximately 15 days of inactivity. This suggests a simple client policy: if a client has been offline for more than about two weeks, it should fetch a fresh snapshot rather than the full delta chain. This minimizes bandwidth

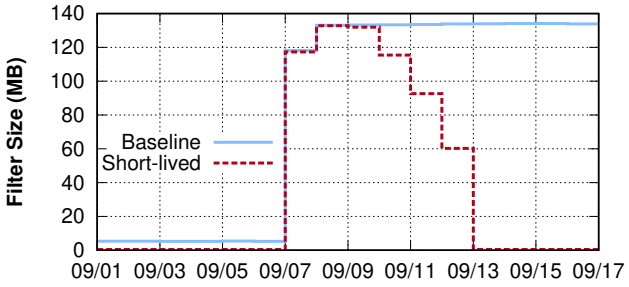


Figure 7: Total Filter Sizes during a 50% mass revocation event. The Baseline (Real-World) distribution suffers from persistent bloat ("spike and stay"), while the Short-lived (6-day) scenario demonstrates a rapid "spike and drop" recovery .

and reduces time-to-synchronization after long offline periods. A production implementation should also consider variance (not just the mean) and wire-format compression effects, but the order-of-magnitude threshold is stable under reasonable assumptions.

6.4 Client-side resource usage

The Clubcard query path was benchmarked using an extension to `rust-query-crlite` [44], run against the live filters Firefox downloads from Mozilla's remote settings infrastructure.

Query latency. Each Clubcard filter query takes approximately 235–270 ns (Table 2), with full filters at the higher end and delta filters at the lower end of that range. Under Mozilla's current publication schedule—one full snapshot every 45 days and delta updates every 12 hours— a client holds at most one full filter plus 89 deltas, requiring 90 filter queries per revocation check. The total query time is therefore approximately 21 µs. Two SHA-256 calls—one to hash the issuer's SPKI and one to form the lookup key `issuer_spki_hash || serial`—add a small constant overhead independent of the number of delta updates.

Query time scales with the number of filters held, so update cadence affects per-check cost. Delta size scales approximately proportionally with the update window [46] —halving the update interval roughly halves each delta—so daily client bandwidth is roughly invariant to cadence. The per-check query time, however, scales with the number of filters: moving from 12-hour to 6-hour updates, while keeping the 45-day snapshot interval fixed, doubles the maximum number of Clubcards a client holds (from 90 to 180), and thus roughly doubles the per-check query time.

Storage. The current production full snapshot is approximately 6 MB; each delta update is approximately 200 KB. At the end of a 45-day publication cycle a client stores roughly 22 MB of filter data. Query latency and storage are essentially identical whether filters are held in memory or read from disk.

For comparison, an OCSP round-trip typically takes tens to hundreds of milliseconds [19]—three to four orders of magnitude slower than CRLite's local lookup. These CPU and memory requirements were low enough that they did not factor into Mozilla's decision-making about whether or not to deploy the system. The binding constraints were on the aggregator and distribution side—bandwidth budgets and false-positive correctness—not client performance.

Component	Count	Size	Query (ns)
Full filter	1	6 MB	270
Delta update	up to 89	~200 KB ea.	236 ea.
Total per validation	—	~22 MB	~21,000

Table 2: CRLite client-side cost per certificate validation. Query latencies are medians over 100 samples of 200,000 calls per filter on the current production filter set.

7 Lessons Learned

In this section, we summarize lessons learned in developing and deploying the most comprehensive certificate revocation-checking system to date.

False positives are unacceptable; design for zero tolerance. The multi-year delay between initial deployment (2019) and full enforcement (2025) was driven primarily by false positive incidents. The original CRLite paper proposed a data structure that eliminated false positives, but only when the client and aggregator were in perfect agreement about which certificates were known to the aggregator, i.e. which certificates were covered. As described in §3, it took multiple design iterations to arrive at the correct notion of coverage. Then, even with the correct definition, subtle bugs in its implementation led to user-observable breakage. The lesson: in browser-scale deployments, "mostly correct" is operationally unacceptable. Every corner case will be exercised by a large enough user base.

Comprehensive revocation is feasible, but required a holistic approach. Our deployment demonstrates that, contrary to decades of assertions otherwise, it is indeed feasible to disseminate all revocations to a large population of users. Achieving this goal, however, required a combination of developments in theory, engineering, and policy. Looking back, the system would not have been successful without *all* of these developments. The broader lesson here is that the original paper naively assumed it could solve the revocation problem "from the outside" using technical means only. But a practical, deployed solution required ecosystem wide changes. Mozilla, and others, needed to engage with the Web PKI (specifically CAs) as a whole and reason about real-world constraints.

8 Looking Ahead

In this section we describe recent and future changes in the PKI and how they might affect CRLite. We also discuss potential changes to the PKI.

8.1 Static CT

As described in §3, the Certificate Transparency ecosystem is moving towards a new API called Static CT [47]. Static CT not only allows for simpler implementations of CT logs, but also rectifies many of the challenges with CRLite filter generation today. The

first is that Static CT logs have a zero or near-zero merge delay. This will reduce or eliminate CRLite's need to add a MMD "buffer period" to each end of the coverage metadata. Second, entries in static CT logs are strictly ordered by sequence number. This will allow coverage for each log to be defined based on sequence number, rather than inclusion time.

8.2 Post-quantum certificates and online validation

The Web PKI community is working on transitioning TLS certificates to cryptographic signature algorithms that are resistant to quantum computers. Unfortunately, the main contenders for post-quantum signature algorithms exhibit large signature (2.4kB for ML-DSA-44) and public key (1.3kB) sizes. For reference, an ECDSA-P256 key is just 64 bytes. Since a single TLS certificate chain regularly contains six signatures and two keys, stuffing signatures that large into the current X.509 certificates increases TLS handshake latency and can even break some connections [48].

In response, alternative designs to traditional certificates have been proposed. One of these designs, Merkle Tree Certificates (MTCs) [18], proposes having each CA periodically push post-quantum signed Merkle Trees to all browsers containing information about all their issued certificates. Certificates are then represented as Merkle Tree proofs—so to validate a certificate, browsers simply check whether the certificate is included in the corresponding CA's Merkle Tree.

Ultimately MTCs do not fundamentally change CRLite's viability. If MTCs become mainstream, CRLite could still be used to distribute revocation information. Rather than containing serial numbers, CRLite would contain MTC indexes (scoped to each CA). In fact, CRLite and MTCs synergize well: both require the browsers to download periodic updates to correctly perform certificate validation. The adoption of MTCs (along with CRLite) would signify a shift in certificate validation to a more "online" process.

8.3 Certificate lifetimes may get shorter still

Engineers from the Chrome security team have suggested doing away with revocation checking altogether [9, 36], in favor of moving to short-lived certificates. This shifts the security model from an explicit "kill switch" (revocation) to an implicit one (expiration). The key question is the vulnerability window: CRLite operates on a 12-hour update cycle, whereas the CA/B Forum schedule trends toward multi-week lifetimes, and more aggressive proposals argue for 24-48 hour lifetimes [9]. Even the trend toward shorter lifetimes is bounded: the current CA/B schedule caps maximum lifetime at 47 days by 2029 [24], and 24–48 h proposals remain proposals rather than mandates.

Short lifetimes simplify client logic, but they also move failure modes onto website operators. Even 47-day lifetimes have drawn operational-risk pushback from CAs and large subscribers, who cite tighter renewal margins and the risk of automation failures. Today, Let's Encrypt certificates are typically valid for 90 days, and common ACME deployments renew roughly 30 days early, leaving real slack for outages, rate limits, configuration mistakes, and transient CA failures. With 24-48 hour certificates, that slack collapses to hours, and ordinary automation failures turn into user-visible site outages. In practice, the realistic end state is not "expiration only", but "shorter lifetimes plus fast client-side revocation", where shorter validity reduces background persistence and CRLite preserves a rapid incident-response path.

8.4 Aggregator trust and scale

CRLite's aggregator runs as a single instance (§4), which raises a reasonable concern about single-point-of-trust and single-point-of-failure. Two properties mitigate this. First, CRLite construction is publicly auditable: anyone with access to Certificate Transparency logs and the published set of CRL distribution points can independently rebuild the filter and verify the version Mozilla ships. Second, the aggregator's work scales with the number of CRL distribution points (currently a few hundred) rather than the number of clients (hundreds of millions), so partitioning by issuer is a straightforward scaling path if pressure ever emerges.

9 Ethics Statement

All of the telemetry data collected for CRLite and analyzed in this study was collected in accordance with Mozilla's Data Privacy Principles [12] and the Mozilla Privacy Policy [13]. The Firefox Privacy Notice is available at [10]. None of the data includes any personally identifiable information (PII), e.g., IP addresses or Mozilla account names, nor does the data contain URLs or any identifying features of certificates. All collected data falls in Mozilla Data Collection Categories 1 and 2 [11].

Acknowledgments

We thank the anonymous reviewers for their helpful feedback. This work was supported in part by NSF CNS-2339378 and a gift from the Mozilla Corporation.

References

[1] CA/Browser Forum. https://cabforum.org/.
[2] Certificate Transparency. https://certificate.transparency.dev/.
[3] Only run CRLite on certificates with a CT SCT available, December 2019. https://bugzilla.mozilla.org/show_bug.cgi?id=1605273.
[4] Evaluate CRLite for certs older than $merge_delay, December 2020. https://bugzilla.mozilla.org/show_bug.cgi?id=1683525.
[5] SEC_ERROR_REVOKED_CERTIFICATE on hetzner.com due to security.pki.crlite_mode=2 (Nightly default), September 2020. https://bugzilla.mozilla.org/show_bug.cgi?id=1667829.
[6] utilize crlite stashes (incremental diffs in crlite state), May 2020. https://bugzilla.mozilla.org/show_bug.cgi?id=1638139.
[7] PSM should use the newly defined CRLite filter coverage metadata, December 2021. https://bugzilla.mozilla.org/show_bug.cgi?id=1747320.
[8] Microsoft PKI Services: Failure to Revoke in 5 Days, May 2025. https://bugzilla.mozilla.org/show_bug.cgi?id=1965612.
[9] Revocation ain't no thang., September 2025. https://dadrian.io/blog/posts/revocation-aint-no-thang/.
[10] Firefox Privacy Notice, January 2026. https://www.mozilla.org/en-US/privacy/firefox/?utm_source=firefox-browser&utm_medium=firefox-desktop&utm_campaign=about-dialog.
[11] Mozilla Data Collection Categories, January 2026. https://wiki.mozilla.org/Data_Collection#Data_Collection_Categories.
[12] Mozilla Data Privacy Principles, January 2026. https://www.mozilla.org/en-US/privacy/principles/.
[13] Mozilla Privacy Policy, January 2026. https://www.mozilla.org/en-US/privacy/.
[14] Josh Aas. We Issued Our First Six Day Cert. https://letsencrypt.org/2025/02/20/first-short-lived-cert-issued, 2025.
[15] Apple. About upcoming limits on trusted certificates. https://support.apple.com/en-us/102028, 2020.
[16] Apple Inc. Certificate transparency requirements. https://support.apple.com/en-us/103214, 2024.

[17] Charles Arthur. Diginotar ssl certificate hack amounts to cyberwar, says expert. http://www.theguardian.com/technology/2011/sep/05/diginotar-certificate-hack-cyberwar.

[18] David Benjamin, Devon O'Brien, Bas Westerbaan, Luke Valenta, and Filippo Valsorda. Merkle Tree Certificates. Internet-Draft draft-davidben-tls-merkle-tree-certs-10, Internet Engineering Task Force, January 2026. Work in Progress.

[19] Protick Bhowmick, Dave Levin, and Taejoong Chung. Reliable and Decentralized Certificate Revocation via DNS: The Case for RevDNS. In *ACM SIGCOMM*, September 2025.

[20] CA/Browser Forum. Ballot sc063v4: Make ocsp optional, require crls, and incentivize automation. https://cabforum.org/2023/07/14/ballot-sc063v4-make-ocsp-optional-require-crls-and-incentivize-automation/, 2023.

[21] Taejoong Chung, Jay Lok, Balakrishnan Chandrasekaran, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, John Rula, Nick Sullivan, and Christo Wilson. Is the Web Ready for OCSP Must-Staple? In *ACM Internet Measurement Conference (IMC)*, 2018.

[22] Zakir Durumeric, James Kasten, David Adrian, J. Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, and Vern Paxson. The Matter Of Heartbleed. In *ACM Internet Measurement Conference (IMC)*, 2014.

[23] CAB Forum. Ballot SC031: Browser Alignment. https://cabforum.org/2020/07/16/ballot-sc031-browser-alignment/, 2020.

[24] CAB Forum. Ballot SC081v3: Introduce Schedule of Reducing Validity and Data Reuse Periods. https://cabforum.org/2025/04/11/ballot-sc081v3-introduce-schedule-of-reducing-validity-and-data-reuse-periods/, 2025.

[25] Aaron Gable. A New Life for Certificate Revocation Lists, September 2022. https://letsencrypt.org/2022/09/07/new-life-for-crls.

[26] Mark Goodwin. Revoking Intermediate Certificates: Introducing OneCRL. https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/, March 2015.

[27] Google Chrome CT Team. Chrome Certificate Transparency v3 Log List. https://www.gstatic.com/ct/log_list/v3/log_list.json, 2026.

[28] Google Chrome Security Team. Certificate transparency. https://chromium.googlesource.com/chromium/src/+/main/net/docs/certificate-transparency.md, 2024.

[29] Google Security Blog. Chrome's plan to distrust symantec certificates. https://security.googleblog.com/2017/09/chromes-plan-to-distrust-symantec.html, 2017.

[30] Google Security Blog. Sustaining digital certificate security - entrust certificate distrust. https://security.googleblog.com/2024/06/sustaining-digital-certificate-security.html, 2024.

[31] Mike Hamburg. Improved CRL compression with structured linear functions. Real World Crypto (RWC), 2022.

[32] Scott Helme. Revocation is broken, July 2017. https://scotthelme.co.uk/revocation-is-broken/.

[33] J.C. Jones. Introducing CRLite: All of the Web PKI's revocations, compressed. https://blog.mozilla.org/security/2020/01/09/crlite-part-1-all-web-pki-revocations-compressed/.

[34] Adam Langley. Revocation doesn't work, March 2011. https://www.imperialviolet.org/2011/03/18/revocation.html.

[35] Adam Langley. Revocation Checking And Chrome's CRL, February 2012. https://www.imperialviolet.org/2012/02/05/crlsets.html.

[36] Adam Langley. Revocation Still Doesn't Work, April 2014. `https://www.imperialviolet.org/2014/04/29/revocationagain.html`.

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

[38] Let's Encrypt. 2020.02.29 caa rechecking bug. Technical report, 2020.

[39] Let's Encrypt. Ending ocsp support. https://letsencrypt.org/2024/12/05/ending-ocsp/, 2024.

[40] Let's Encrypt. 6-day and ip address certificates are generally available. https://letsencrypt.org/2026/01/15/6day-and-ip-general-availability, 2026.

[41] Yabing Liu, Will Tome, Liang Zhang, David Choffnes, Dave Levin, Bruce Maggs, Alan Mislove, Aaron Schulman, and Christo Wilson. An End-to-end Measurement Of Certificate Revocation In The Web's PKI. In *ACM Internet Measurement Conference (IMC)*, 2015.

[42] Vincent Lynch. GlobalSign Certificate Problems Accidentally Block Sites. Hashed Out by The SSL Store, October 2016. Industry analysis of the 2016 GlobalSign revocation error and its impact on OCSP and CDN caching.

[43] Mozilla. Firefox 135.0 release notes. https://www.firefox.com/en-US/firefox/135.0/releasenotes/, 2025.

[44] John Schanck. rust-query-crlite microbenchmark extension. https://github.com/jschanck/crlite/tree/microbench, 2026.

[45] John M Schanck. CRLite: Fast, private, and comprehensive certificate revocation checking in Firefox. https://hacks.mozilla.org/2025/08/crlite-fast-private-and-comprehensive-certificate-revocation-checking-in-firefox/.

[46] John M Schanck. Clubcards for the WebPKI: smaller certificate revocation tests in theory and practice. In *IEEE Symposium on Security and Privacy*. IEEE, 2025.

[47] Filippo Valsorda. The Static Certificate Transparency API. https://github.com/C2SP/C2SP/blob/main/static-ct-api.md.

[48] Bas Westerbaan. The state of the post-quantum Internet. Cloudflare, 2024. https://blog.cloudflare.com/pq-2024/.

[49] Kathleen Wilson. Revocation Reason Codes for TLS Server Certificates, May 2022. https://blog.mozilla.org/security/2022/05/16/revocation-reason-codes-for-tls-server-certificates/.

[50] Liang Zhang, David Choffnes, Tudor Dumitras, Dave Levin, Alan Mislove, Aaron Schulman, and Christo Wilson. Analysis Of SSL Certificate Reissues And Revocations In The Wake Of Heartbleed. In *ACM Internet Measurement Conference (IMC)*, 2014.