Bloom Filter Enhanced OCSP

Norbert Varga[1,*], Martin Mislovič[1] and Ladislav Hudec[1]

[1]*Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava, Slovakia*

Abstract

Verifying the revocation status of public-key certificates is a challenging task in the Internet of Things (IoT). Certificate revocation lists (CRLs) are too large for constrained devices, and OCSP is costly both in computation and in transmission, so deployments frequently rely on ad-hoc schemes that provide weaker security guarantees. This paper presents and evaluates *Bloom OCSP*, a minimal modification of OCSP that queries entire pre-assigned groups of certificates rather than individual ones and returns the group's revocation state as a bit array compressed with a Bloom filter. The protocol differs from prior work in particular by eliminating false positives on the responder side while remaining on standard PKI protocols, which simplifies both integration and extension. We implemented the protocol as a research prototype and evaluated it directly on an ESP32 microcontroller. Within a single exchange smaller than one standard OCSP response, the prototype reliably transmits up-to-date revocation information for more than 500 certificates; the false-positive elimination scheme guarantees the correctness of every response, and querying by group tells the responder only which group, not which specific certificate, a client is verifying, giving group-level (k-anonymity) protection rather than full unlinkability.

Keywords

Internet of Things, certificate revocation, OCSP, Bloom filter, PKI, ESP32

1. Introduction

The Internet of Things (IoT) is expanding across both households and industry, increasing the need to verify the identity of communicating devices. Certificate revocation is especially problematic in this setting: the computational resources of an IoT device may be so limited [1] that the standard X.509 validation procedures are not performed on it and are replaced by ad-hoc schemes such as pre-shared symmetric keys [2], which do not scale to newer deployments such as large sensor networks or connected vehicles.

The two predominant methods of distributing revocation information, CRLs [3] and OCSP [4, 5], are both poorly suited to constrained devices. A CRL is large and impractical to download over an unreliable connection, whereas OCSP is demanding in both computation and transmission, mainly because each response is signed and verified individually. Prior work indicates, however, that OCSP can be made much more efficient, primarily by allowing a single signed response to cover many certificates and by compressing the transmitted revocation information [2, 6].

The objective of this work is to make revocation checking efficient enough for resource-constrained IoT devices while preserving compatibility with the existing PKI. The contributions of the paper are as follows:

- We propose *Bloom OCSP*. Instead of querying the status of a single certificate, the client queries an entire group and receives the group's revocation state as a bit array compressed with a Bloom filter, so that a single exchange covers far more certificates than standard OCSP. Groups are not chosen by the client. The certification authority assigns each certificate to a group at issuance and records the identifier in a certificate extension, so the client reads it from the certificate it is validating and sends that single number instead of a serial number. The modifications to OCSP are kept minimal: we remain on standard PKI protocols and technologies (HTTP/TCP, ASN.1) so that the protocol integrates into existing infrastructure.

ITAT'26: Information Technologies – Applications and Theory, September 25–29, 2026, Biele Karpaty, Slovakia

*Corresponding author.

norbert.varga@stuba.sk (N. Varga); xmislovic@stuba.sk (M. Mislovič); ladislav.hudec@stuba.sk (L. Hudec)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-09

CEUR-WS.org/Vol-4323/paper38.pdf

- We introduce a *false-positive elimination* step on the responder side. In contrast to the probabilistic nature of a Bloom filter, it guarantees correct responses without a fallback query from the client.
- We implemented the protocol and evaluated it *on a real IoT device* (ESP32), not only in simulation. The ESP32 represents an intermediate class of device: it supports standard protocols and libraries, yet it is much more constrained than a general-purpose computer.
- Because the client queries a group, the query is *partially anonymized*: the responder cannot determine which certificate of the group the client is verifying.

On the ESP32 we show that a single exchange smaller than one OCSP response can transmit revocation information for more than 500 certificates, and that verifying a single certificate requires at least 66 bytes less than with OCSP.

We emphasize that the presented implementation is a *research prototype (proof of concept)*: its purpose is to demonstrate feasibility, to enable the measurement of real metrics on a constrained device, and to provide a basis for further research, rather than to serve as a production system. Several components are deliberately simplified; we state these limitations explicitly in Sec. 6, where they also serve as starting points for future work.

The remainder of the paper is organized as follows. Section 2 reviews related technologies and work. Section 3 describes the design of Bloom OCSP, Section 4 its implementation, and Section 5 its evaluation. Section 6 discusses the limitations and the status of the prototype, and Section 7 concludes the paper and outlines directions for future work.

2. Related Work

Certificate revocation is not exclusively a web problem. It is equally relevant in code signing, in cryptographic tokens and smart cards, in vehicular networks, and in the IoT, that is, in any setting where a long-lived certificate may be compromised before it expires. The web represents a notable case: online checking (OCSP) is progressively being replaced by pre-distributed structures for reasons of privacy, performance, and reliability, whereas constrained, domain-specific settings such as the IoT continue to require fresh online checks, which is the case addressed here. We classify existing work by the data structure used to represent revocation state and, at the end of the section, position Bloom OCSP among these approaches; Table 1 summarizes the comparison.

2.1. Standard mechanisms: CRL and OCSP

A certificate revocation list (CRL) [3] is an authority-signed list of the serial numbers of revoked certificates. A client downloads it once and may use it offline, but the list is large (in a web-PKI measurement the median certificate's CRL was about 51 KB and the largest exceeded 70 MB [7]). An entry may become stale between updates, and successive versions overlap heavily, which is why delta-CRLs transmit only the increments. OCSP [4, 5] instead verifies a single certificate by means of an *online query* to a responder; although a response may in principle cover several certificates, in practice it covers only a few. This reduces the client's memory requirements and improves freshness, though at some cost. Each response must be signed individually, and the client depends on the responder's availability. There is also a privacy concern, since the responder learns which certificates, and therefore which services, the client is using. Both mechanisms are widely deployed today, and the remainder of this section consists of approaches that reduce their overhead by representing revocation state differently.

2.2. Pre-distributed structures in browsers

Major browsers have largely abandoned online OCSP in favor of a *quite different* model: instead of querying each certificate, the client periodically retrieves a single, centrally prepared structure and verifies status *locally*. This eliminates both the privacy leak and the query latency, at the cost of soft-fail behavior (a missing entry is treated as valid). Chrome CRLSet [8] distributes only a curated set of

important revocations, and Firefox OneCRL [9] pushes a list of revoked intermediate CA certificates. CRLite [10] extends this model to all end-entity certificates, compressing every web-PKI revocation into a *cascade of Bloom filters* [11] (approximately 6.6 bits per revocation) and distributing regular updates to clients. Alternating filters of revoked and valid certificates eliminate the false positives that a single Bloom filter would introduce. These designs assume the globally connected browser environment and a complete certificate set. They do not transfer to constrained IoT, where a device requires the fresh status of an individual certificate without retrieving a global structure.

2.3. Academic and compact structures

Hash trees. Kocher's revocation tree [12] proves a certificate's status from the signed Merkle-tree root and the hashes along the path to it, reducing the proof from $O(n)$ to $O(\log n)$. However, a single proof covers only one certificate and the cost remains higher than that of OCSP, so the approach is largely of academic interest.

Bitmaps. The compressed CCSP protocol [6] returns a *bitmap* for an entire group of certificates in a single OCSP-like response rather than the status of one certificate, so that one signature covers the whole group and the responder no longer learns which specific certificate is being checked (the client requests only the group). As with our protocol, this yields partial query anonymization. It is the closest predecessor of our group-query model. CCSP is, however, web-oriented, and its bitmap must be *decompressed* before use, which is disadvantageous on constrained devices. A Bloom filter, being a probabilistic structure, can be queried directly, without decompression.

2.4. Revocation for the Internet of Things

The most closely related work is TinyOCSP [2], which transports OCSP over CoAP and CBOR and removes redundant fields, reducing the single-validation message overhead by at least 73%. Its authors also outline a Bloom-compressed CRL, but that component remains at the design-and-analysis stage (formulas and simulation) and was neither implemented nor measured on a constrained device, unlike the remainder of the protocol. This is exactly the contribution we make: we implement and measure the Bloom compression on the microcontroller itself, and we eliminate false positives at the responder rather than tolerating them.

For intermittently connected constrained networks, V'CER [13] constructs a *validation tree* of valid certificates in which only the root is signed: individual certificates require no signature of their own, and the single root signature attests to many of them. A certificate proves its validity through its inclusion in the tree, and nodes exchange these proofs among themselves, which greatly reduces the need to contact the authority, an advantage in the IoT, where connectivity is unreliable. A deterministic alternative to Bloom filters is provided by cryptographic accumulators such as EVOKE [14]. The entire revocation state is condensed into a single short accumulator value, against which a device proves, by means of a witness, that its certificate belongs to the set of valid ones, *without false positives*. Its elliptic-curve accumulator is lightweight relative to RSA- or Merkle-tree-based alternatives, though it still requires public-key operations rather than the plain hashing used here. It was evaluated on commodity hardware and a Raspberry Pi, not on an ESP32-class microcontroller.

The difficulty of verification on such hardware results from its constraints. RFC 7228 [1] classifies constrained devices by memory. Class 0 ($<$ 10 KiB RAM) is too limited even for simplified protocols. Class 1 (~10 KiB RAM, 100 KiB Flash) cannot support full TLS/HTTP but can run lightweight protocols such as CoAP [15]. Only class 2 (~50/250 KiB) approaches commodity devices. The lightweight TLS libraries MbedTLS [16] and wolfSSL [17] were developed for classes 1 and 2. Establishing a secure connection on such a device is itself demanding. A recent TLS 1.3 client for the ESP32 [18] shows this, with a handshake of 0.9–1.5 s and a binary of approximately 1.2 MB including embedded certificates. That work evaluates the handshake and certificate-chain authentication but does not consider revocation, which is commonly left out of constrained TLS deployments and is the gap addressed here. A recent survey reaches the same conclusion, recommending query-based schemes for memory-constrained

devices and calling constant per-check overhead at minimal client storage an open problem [19]. With respect to privacy, our design does not provide the guarantees of dedicated revocation-privacy schemes; it offers only an inexpensive, partial anonymization (the responder does not learn which certificate of the group is being queried).

2.5. Summary and positioning

The development proceeds from large lists, through hash trees and bitmaps, to Bloom filters, that is, toward compact, group-oriented representations of revocation state. For the constrained IoT environment, the most promising are group bitmaps (CCSP) and Bloom filters (CRLite). Each has a shortcoming, however. They are either not query-based and require a global structure, or they handle false positives in a manner unsuitable for a lightweight client. Bloom OCSP combines a group-compressed map with the OCSP query model and is distinguished from prior work in four respects (cf. Table 1).

Unlike CRLs and pre-distributed structures such as CRLite, it retains the query to a responder but queries entire groups, so no global structure must be distributed or maintained and the response remains current. It handles false positives neither with a cascade (CRLite) nor with a client-side fallback query (TinyOCSP). Instead, the responder, which knows the state of the entire group, tightens a single filter. Unlike TinyOCSP, it remains on standard HTTP/TCP and ASN.1 and was validated on a real microcontroller, including the compression that TinyOCSP only designed and simulated. The Bloom array can also be queried without the decompression required by the CCSP bitmap. Finally, unlike accumulators (EVOKE), it avoids elliptic-curve cryptography, which is appropriate for the ESP32 class, at the cost of a probabilistic structure that the false-positive elimination step renders exact.

This leads to the hypothesis that the compressed-bitmap approach of CCSP can be transferred to the IoT and further reduced using a Bloom filter, yielding a protocol that transmits information about many more certificates per exchange without additional overhead. Transmitting information about many certificates at once is also beneficial for multi-hop networks, where a single query can cover the status of every certificate in the immediate neighborhood.

Table 1
Comparison of protocols and schemes for certificate revocation.

Property	OCSP	**Bloom OCSP**	TinyOCSP	CCSP	CRLite	EVOKE
Status retrieval model	query	query [g]	query	query	pull	push
Network stack	TCP/HTTP	TCP/HTTP	UDP/CoAP	TCP/HTTP	HTTPS	DLT/P2P
Revocation compression	no	yes [a]	yes [b]	yes [c]	yes [e]	yes [f]
False-positive handling	—	responder	client	—	cascade	—
Query anonymization	no	partial	no	partial	yes	yes
Standard PKI protocols	yes	yes	no	yes	yes	no
Verified on IoT device	yes	yes	yes [d]	no	no	partial [h]

[a] no false positives [b] Bloom CRL with false positives [c] bitmap (CRL) [d] compression only designed and simulated [e] filter cascade
[f] cryptographic accumulator [g] group query [h] on commodity devices and a Raspberry Pi, not on a microcontroller

3. Protocol Design: Bloom OCSP

Bloom OCSP transmits revocation information for several certificates in a single modified OCSP response, compressed with a Bloom filter. Unlike TinyOCSP [2], we do not transport the protocol over CoAP but retain standard HTTP/TCP and ASN.1. We give up part of the performance in exchange for compatibility with a large portion of the existing PKI and the ability to reuse substantial parts of existing SSL libraries; the modifications to OCSP are therefore kept as minimal as possible.

Verification proceeds in several steps (Fig. 1): the client reads the group identifier from the certificate, checks its cache, sends a query for the entire group if necessary, receives the Bloom-filter response, and determines the status of the certificate from the bit array.

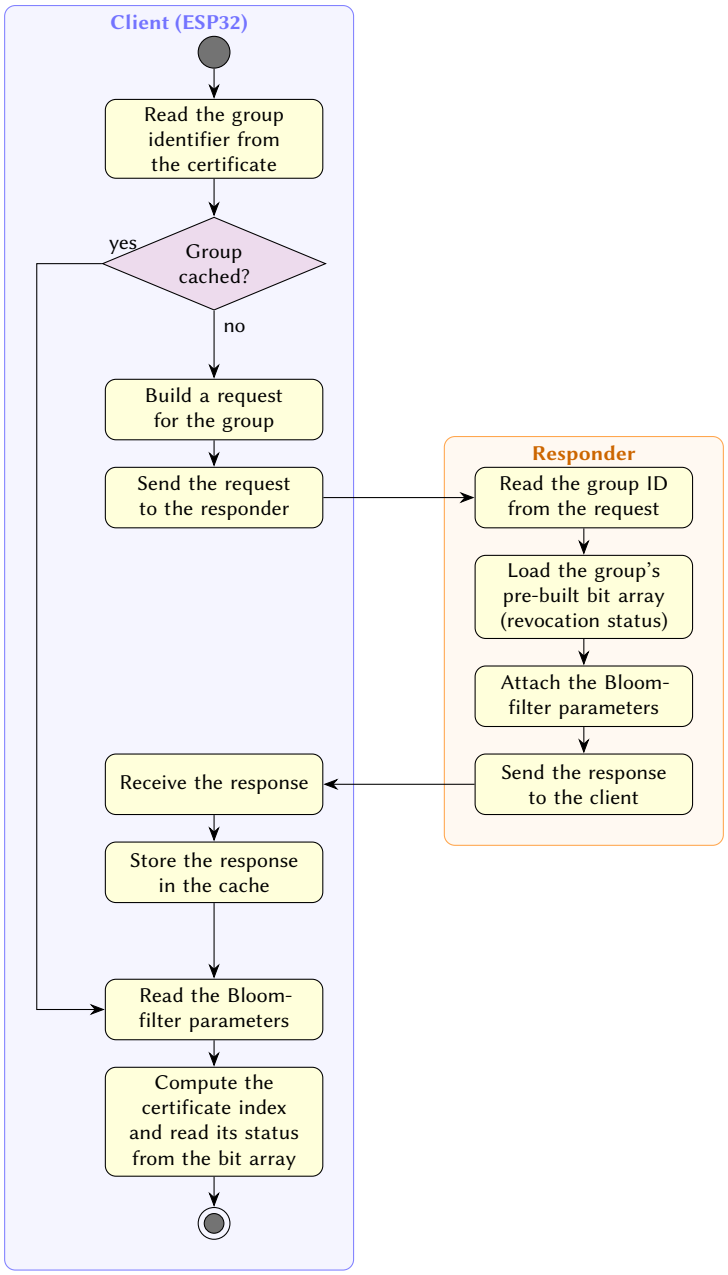


Figure 1: Activity diagram of the certificate-verification flow in Bloom OCSP, adapted from the master's thesis [20]. If the group's revocation data is already cached, the client skips the query and reads the status directly from the bit array; otherwise it queries the responder, which returns the Bloom-compressed bit array for the entire group. Because false positives are eliminated at the responder, the client never issues a fallback query.

3.1. Request and response

In the request, the `serialNumber` field of `CertID` is replaced by the group identifier `certGroupId`. Since ASN.1 represents both as integers, this constitutes a minimal change to existing libraries.

The principal changes are in the response, where the `SingleResponse` sequence is replaced by a new `BloomResponse` sequence that conveys the entire Bloom-compressed bit array instead of the status of a single certificate:

```
BloomResponse ::= SEQUENCE {
  bloomResponseVersion   OBJECT IDENTIFIER,
  timestamp              GeneralizedTime,
  bitfield               BIT STRING,
  certGroupIdentifier    INTEGER,
  numHashes              INTEGER
}
```

The `bloomResponseVersion` field (an OID) specifies the protocol version and allows future changes, such as replacement of the hash function, while `timestamp` replaces the original `thisUpdate`. The current design carries no `nextUpdate` equivalent, so the validity of a cached entry is left to a client-side policy (a fixed interval in our tests); a future revision should add an explicit validity field (Sec. 6). The `certGroupIdentifier` field binds the response to a specific group, and `bitfield` contains the Bloom-compressed array itself. The number of hash functions is selected by the responder according to the group size and the target false-positive rate; because this rate changes during false-positive elimination (Sec. 5), it is transmitted to the client in the `numHashes` field, without which the client would not know how many hash functions to apply when querying the filter. Since this count must be an integer, it is rounded before use, which shifts the achieved false-positive rate slightly from the optimum of (1), a deviation the elimination step then absorbs. Relative to the original design, the sequence of explicit hash algorithms was removed, since both parties use the agreed method (MD5), which reduces the transmitted data.

3.2. Certificate extension

To enable group queries, a certificate must indicate the group to which it belongs. We convey this in a non-critical X.509 extension containing a single field, a numeric group identifier:

```
GroupIDCertExtension DEFINITIONS EXPLICIT TAGS ::= BEGIN
  id-pe-groupID OBJECT IDENTIFIER ::= { 1.3.6.1.4.1.99999.1 }
  GroupID ::= INTEGER
END
```

As an extension, it preserves all original certificate data, so a client without Bloom OCSP support can still verify the certificate using standard OCSP. (The OID used here is for research purposes only; a production deployment would use a properly assigned identifier.)

The client also needs the responder's network address, but this requires no new field. As in ordinary OCSP, that address is read from the certificate's Authority Information Access extension. A client that supports Bloom OCSP and finds the group identifier appends a dedicated path (`/bloom`) to the address to reach the group-query endpoint. A client without such support uses the ordinary OCSP endpoint.

Group assignment need not be random. Aligning groups with logical deployment units, such as a subnet, a batch of devices from a single manufacturer, or a fleet with a common role, organizes revocation by deployment and allows an entire subnet or device group to be revoked in a single operation, by one modification to the corresponding group. This is especially useful in the IoT, where devices are commonly deployed and decommissioned in batches.

3.3. False positives

A false positive would cause a valid certificate to be evaluated as revoked, disrupting communication. False positives are an inherent property of Bloom filters, so they must be addressed. Rather than employing a fallback deterministic query on the client side, we perform *active false-positive elimination on the responder side*: since the responder knows the correct state of all certificates in the group, after constructing the bit array it verifies whether a false positive has occurred and, in case of a conflict, tightens the target false-positive rate and recomputes the array. The client therefore always receives a *single* correct filter without the need for a further query. This construction runs *offline*: the responder,

which is not resource-constrained, builds a filter per group and refreshes it periodically as the revocation state changes, so a query is answered by returning the group's current, pre-built filter rather than by computing it on the request path.

A stricter false-positive rate entails a larger bit array. For n revoked certificates in a group and a target false-positive rate p, the array size m (in bits) and the number of hash functions k are given by [21]:

$$m = -\frac{n \ln p}{(\ln 2)^2} \qquad k = \ln 2 \cdot \frac{m}{n}. \tag{1}$$

Both expressions are the classical optimum of the filter [11, 21]. With k independent, uniform hash functions over m bits, a given bit stays zero with probability $\approx e^{-kn/m}$. An element that was never inserted is therefore reported as a match with probability $p \approx (1-e^{-kn/m})^k$. Minimizing that probability over k yields the second relation, $k = (m/n)\ln 2$, which describes a filter that is half full. Substituting it back reduces the false-positive rate to $p = 2^{-k}$, and solving that for m gives the first relation. Each inserted certificate therefore costs $m/n \approx 1.44 \log_2(1/p)$ bits. Since p enters only through its logarithm, the array grows only *logarithmically* as the false-positive rate is tightened, and each halving of p adds approximately 1.44 bits per certificate. False-positive elimination is therefore inexpensive: the responder, which is not resource-constrained, can tighten p by several orders of magnitude at the cost of only a small increase in array size. Two further advantages follow: the client remains simple, and partial anonymization is preserved, since it still queries only the entire group. The distinction from other Bloom-based approaches (the cascade in CRLite, the tolerance of false positives in TinyOCSP) was discussed in Sec. 2.

3.4. Security considerations

A full security analysis is beyond the scope of this prototype, but we outline an informal threat model. Bloom OCSP inherits the response structure of OCSP: the `BloomResponse` is carried inside the signed `tbsResponseData`. The responder's signature therefore covers the whole structure, including `numHashes` and `certGroupIdentifier`, and a network attacker cannot tamper with the array, the hash count, or the group binding without invalidating the signature. (Our prototype does not yet verify this signature on the client; see Sec. 6.)

A *malicious or compromised responder* can misreport revocation state, but this is already true of standard OCSP and the group model does not worsen it: the worst it can do is mark a valid certificate as revoked, a denial of service that a signed lie would also achieve in OCSP. One group-specific gap is that, since the client sends only a group identifier, it cannot by itself detect that the responder answered for the wrong group. Binding the certificate to its group through the extension and disciplined group management mitigate this.

Two attacks are specific to the compression. First, a *client-driven denial of service*. Since false-positive elimination is the costly step, an attacker might try to trigger it repeatedly with crafted queries. This does not work in our design. Filters are built offline and refreshed periodically, and a query only returns the stored filter for a group (Sec. 3), so a request for a very large or non-existent group costs at most a lookup. Group sizes are bounded when the filters are built, keeping even that periodic construction cost bounded. Second, an *algorithmic-complexity attack* on the unkeyed MD5 hashing (Sec. 4): an adversary able to influence issued serial numbers could choose them to maximize collisions and inflate the array. A keyed hash with a per-group secret seed (e.g. SipHash) removes this ability at negligible cost. We flag both as targets for hardening in future work.

Finally, the anonymity is limited and depends on the grouping policy. A group query yields k-anonymity *within* the group. When groups are aligned with deployment units (Sec. 3) for convenient batch revocation, the group identifier may reveal nearly as much as a serial number, and client IP and timing weaken it further. Conversely, assigning groups *independently* of the deployment topology (so that a group mixes unrelated devices) raises the privacy the scheme offers, at the cost of losing the

batch-revocation convenience, a trade-off left to the group-assignment policy. In either case it is an inexpensive by-product, not a substitute for a dedicated revocation-privacy scheme.

4. Implementation

The aim of the implementation is to demonstrate the feasibility of the protocol and to obtain real *transmission* metrics from the device itself rather than from simulation; it is a prototype rather than a finished product, and certain components are deliberately simplified (Sec. 6 lists these and their consequences). It comprises a client on the ESP32 device, written in C, and a responder written in Python, as outlined in Fig. 2. The complete prototype and its design are documented in the underlying master's thesis [20], and the proof-of-concept code is publicly available [22].

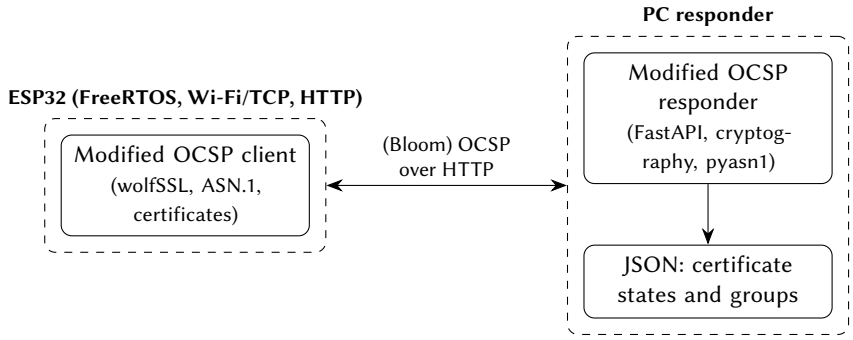


Figure 2: System architecture: the constrained client (ESP32) communicates with the responder on a PC using modified OCSP messages over HTTP.

4.1. Client (ESP32)

We selected the ESP32 microcontroller (dual-core, 2 MB flash, minimum free heap 261,040 B) as a representative IoT device: it cannot run a full operating system but provides Wi-Fi and support for cryptographic libraries. The choice is deliberate, as a *moderately constrained* device. It is not a general-purpose computer such as a Raspberry Pi, on which several related solutions were evaluated, nor is it the most constrained sensor, which is incapable of standard protocols. In terms of RAM, the ESP32 lies at the upper bound of RFC 7228 class 2 [1], or marginally above it. A basic difference nonetheless separates it from general-purpose devices such as the Raspberry Pi. It operates without an operating system and with orders of magnitude less memory, so it remains far more constrained. This intermediate position is where our work is most useful. It targets a compromise among standard PKI protocols, existing libraries, and resource-constrained hardware that nonetheless remains open to further development. Development was carried out in the ESP-IDF environment [23], which provides low-level access to the hardware.

For the handling of certificates, ASN.1, and DER encoding, we used the wolfSSL library [17], which we compiled manually in order to access its internal functions for low-level ASN.1 processing (for example `getSequence()`) and to build our own response decoder upon them. Both the verified certificate and the issuer certificate are embedded in the device. Wi-Fi connectivity and the transmission of HTTP requests are handled by the `esp_wifi` and `esp_http_client` libraries; the implementation is blocking.

4.2. Responder (Python)

Since the work focuses on the client, the responder does not implement the full functionality of an OCSP responder but only generates valid responses. Its endpoints are implemented with FastAPI, the HTTP server with uvicorn, OCSP messages with the `cryptography` library, and ASN.1 with `pyasn1`. For ease of testing, certificate states and groups are stored in `.json` files (the implications of this simplification are discussed in Sec. 6).

4.3. Bloom filter

To ensure identical behavior on the client and the server, we developed our own implementation of the Bloom filter. It derives the number of hash functions k and the array size m from the number of inserted elements n and the target false-positive rate p according to relation (1) in Sec. 3. The individual hash functions are simulated by prepending the function index to the byte representation of the serial number and applying MD5. MD5 was selected for its low computational cost; collision resistance is not required to protect group membership, which is public and stated directly in the certificate. The unkeyed hash is, however, predictable: an adversary able to influence the serial numbers a certification authority issues could choose them to maximize collisions, inflating the bit array and forcing repeated recomputations at the responder (an algorithmic-complexity attack, Sec. 3.4). A keyed hash with a per-group secret seed, such as SipHash (comparably cheap on the ESP32), would remove this ability; we leave its integration to future work.

5. Evaluation

The evaluation addresses the *transmission* characteristics of the protocol: message sizes, the fill of the bit array, and cache effectiveness. The prototype does not measure memory or energy consumption (Sec. 6). Because false-positive elimination is non-deterministic, every group configuration was measured in five independent runs; for groups of 200 certificates and more the standard deviation stays within about a tenth of the mean.

5.1. Bit-array size and revocation rate

Relation (1) indicates that the array size grows with the number of encoded (revoked) certificates. The reference for efficiency is a simple bitmap, which uses one bit per certificate. Fig. 3 shows the theoretical growth for a group of 50 certificates at a base false-positive rate $p = 0.1$.

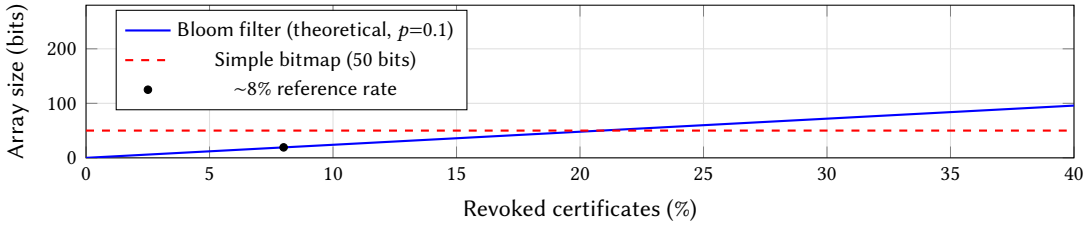
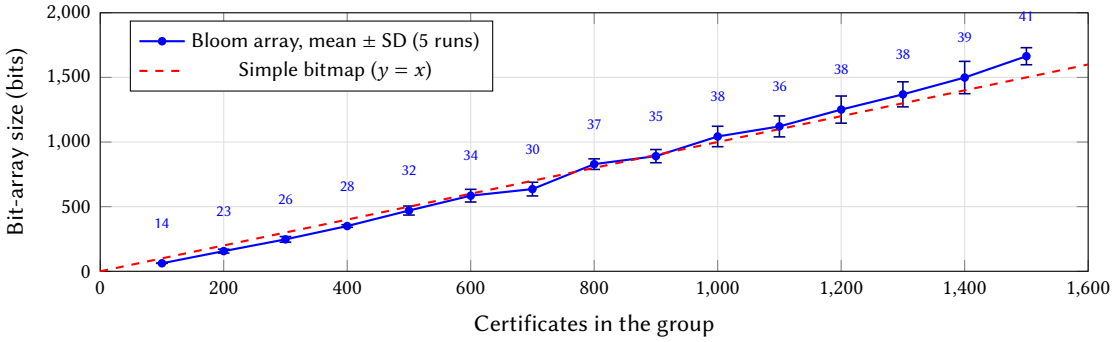
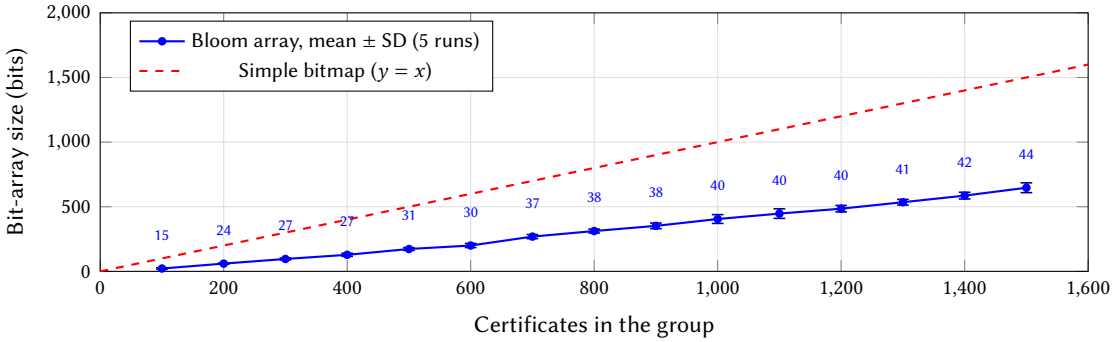


Figure 3: Theoretical size of the Bloom array for a group of 50 certificates, before false-positive elimination. At the pessimistic reference rate of ~8% [7] and $p = 0.1$, the Bloom array is much smaller than a simple bitmap; the two are equal at approximately 21% fill. False-positive elimination narrows this margin (Fig. 4a).

The measured data confirm this approximately linear growth. The theoretical break-even point lies at approximately 21% fill (Fig. 3); in a practical implementation, in which the array is rounded to whole bytes and the response also carries a header, the boundary is somewhat lower, approximately 16–18% for a group of 50 certificates (and approximately 20% for 20 certificates). The principal precondition for efficiency is that revoked certificates constitute only a small fraction of the group, which is reasonable in the IoT, where device certificates are typically long-lived and static. Precise revocation rates for the IoT are not well documented. As a deliberately pessimistic reference we adopt ~8%, the fraction of *fresh* web-PKI certificates found revoked in a large-scale measurement [7], a value inflated by the Heartbleed mass-revocation, whereas *actively served* certificates there stayed below 1%. A later measurement of 48 million certificates reports 2.18% revoked, and 0.35–1.5% once one mass-revocation event is excluded [24]. Realistic rates for long-lived IoT certificates are therefore likely far lower, where the Bloom encoding is even more advantageous (Fig. 4b). At this fill the theoretical Bloom array is well below the bitmap; the next subsection shows how false-positive elimination narrows this gap in practice.



(a) ~8% revocation



(b) ~3% revocation

Figure 4: Size of the Bloom array after false-positive elimination versus the number of certificates in the group. Each point is the mean of five runs, rounded to whole numbers, the bars span one standard deviation, small enough here to be barely visible, and the value beside the point is the mean recomputation count. The line $y = x$ is a simple bitmap (one bit per certificate). (a) At the ~8% reference rate the array tracks the bitmap and crosses above it for the larger groups; (b) at ~3% it stays well below across the entire range, confirming that the size advantage grows as the revocation rate falls.

5.2. False-positive elimination and scalability

False-positive elimination enlarges the array, since each conflict tightens the false-positive rate. Substituting into (1) gives the rate at which a Bloom array and a simple bitmap are equal in size at 8% fill: $p \approx 0.25\%$. The implemented scheme starts at $p = 0.1$ and multiplies it by 0.9 per recomputation, so it can recompute at most about 35 times ($0.1 \cdot 0.9^{35} \approx 0.25\%$) before reaching this break-even size. What drives the array toward the bitmap as the group grows is false-positive elimination rather than the number of revoked certificates. At a fixed revocation rate, more revoked certificates enlarge the Bloom array and the bitmap in equal proportion and so do not change their ratio. The target false-positive rate per check is likewise independent of group size, but a larger group means more certificates are tested against the filter, so the probability that at least one yields a false positive grows. Larger groups therefore require more recomputations, which lower p and inflate the array beyond its linear baseline (Fig. 4a). At the 8% reference rate the array tracks the bitmap closely and crosses above it for the larger groups. Because hash collisions are non-deterministic, no fixed group size cleanly separates the two, but over the five-run means the array stayed below the bitmap for groups of up to roughly 700 certificates and began to cross above it in the 700–1000 range. At this revocation rate a group size below about 700 certificates is therefore a reasonable operating point. Beyond it the simple bitmap is the better choice. More broadly, the Bloom array is clearly smaller than a bitmap only at lower revocation rates, where fewer certificates are inserted. At 8% the advantage is modest, and the bitmap is best read as a reference bound. Figure 4b repeats the measurement at ~3% revocation, where the array indeed stays well below the bitmap across the entire range, confirming that the advantage grows as the revocation rate falls. The number of groups is unbounded, so the protocol still scales to any number of certificates,

and selecting the data structure per revocation rate (Sec. 7) would keep the encoding at or below the bitmap throughout. The gap between the two regimes is substantial. At ~3% (Fig. 4b) the array never exceeds roughly 45% of the equivalent bitmap: even for a group of 1500 certificates it occupies about 650 bits against the 1500 bits of a bitmap, while the false-positive elimination still converges in at most a few dozen recomputations. At ~8% (Fig. 4a), by contrast, the array is already at parity with the bitmap in the 700–1000 range and lies consistently above it for the largest groups, because the larger number of inserted (revoked) certificates both enlarges the base array and provokes more elimination rounds. The Bloom encoding is therefore preferable in the low-revocation regime characteristic of long-lived IoT device certificates, where its footprint is a small fraction of the bitmap; as the revocation rate rises toward the web-PKI reference, the two converge and the per-group choice of representation (Sec. 7) becomes the deciding factor.

5.3. Size comparison

We first compared message sizes in the least favorable case, a group containing a single certificate, in which Bloom-filter compression cannot yet take effect. We compared the total exchange (request and response) of Bloom OCSP with that of standard OCSP, whose response for a revoked certificate was 460 B. Even in this case, Bloom OCSP is at least 66 bytes smaller, owing to a more compact message structure and the removal of redundant fields (for instance, the explicit hash-algorithm sequence, Sec. 3). A further notable contribution arises from replacing the certificate serial number with the group identifier: in OCSP the serial number may be encoded in as many as 20 octets [3] (issuers typically use long random serial numbers), whereas the compact numeric group identifier is much shorter, in both the request and the response. With the grouping of certificates, the saving increases further: compression enables a message smaller than a single OCSP response to convey information about more than 500 certificates. Fig. 5 compares how many certificates each approach can transmit within the volume of a single standard OCSP response at an 8% revocation rate. The Bloom OCSP and bitmap capacities were measured on our implementation; the TinyOCSP value is an estimate derived from the minimal message encoding reported in [2], since we did not reimplement that protocol.

For a single message, TinyOCSP [2] is smaller than our solution owing to its minimal overhead (CoAP and removed fields). Many of its modifications could, however, be combined with Bloom OCSP to obtain a protocol that is both minimal in size *and* compresses the revocation information with a Bloom filter. Table 1 (Sec. 2) provides the detailed qualitative comparison.

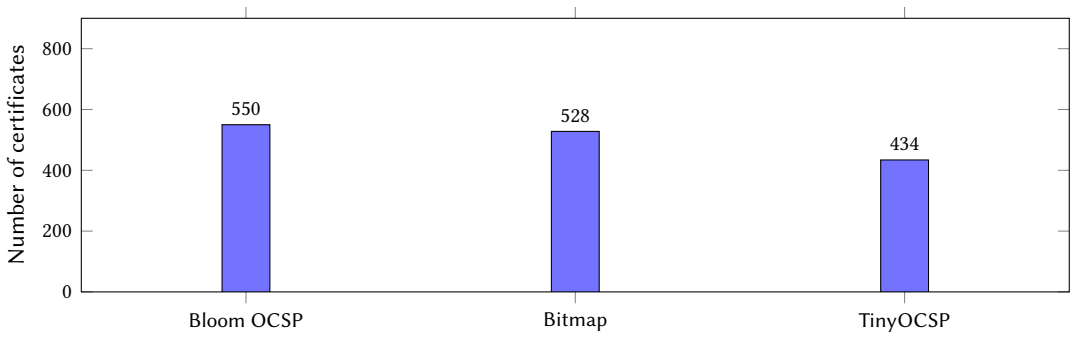


Figure 5: Number of certificates whose revocation information can be transmitted within the volume of a single standard OCSP response (at 8% revocation). Bloom OCSP with false-positive elimination transmits the largest number of certificates.

5.4. Cache effectiveness

Grouping certificates also increases the probability of a cache hit. In a test comprising 100 verifications over 100 certificates divided into 10 groups (entry expiry of 1 minute), Bloom OCSP achieved a mean of 29 hits, compared with 4 for standard OCSP (Fig. 6), which corresponds to an increase in the hit probability from 1:100 to 1:10.

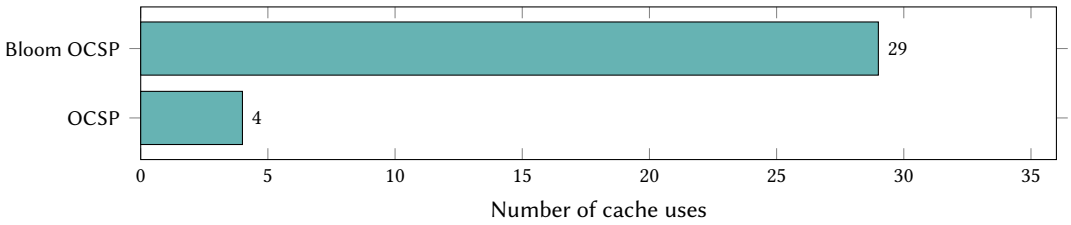


Figure 6: Mean number of cache hits over three runs.

6. Limitations and Prototype Status

The presented solution is a research prototype that demonstrates feasibility and provides real transmission metrics; it requires further development before production deployment. The principal limitations and open issues, several of which also indicate directions for future work (Sec. 7), are as follows:

- **Responder.** It does not implement the full functionality of an OCSP responder, only the valid responses required to test the client. Certificate states and groups are stored in `.json` files; a production deployment would require a reliable ACID database, otherwise the group stated in a certificate may become inconsistent with the responder's records.
- **Client.** The implementation is blocking (synchronous) and not optimized for energy consumption. The wolfSSL library was integrated manually through its internal functions rather than through a stable public interface, which reduces portability and maintainability.
- **Hashing and false-positive elimination.** MD5 was selected for its low overhead; the choice of hash functions, seeds, and constants in the elimination scheme is not optimized and limits the maximum advantageous group size (experimentally 700–1000 certificates).
- **Certificate grouping.** The strategy for forming and assigning groups (size, composition, `GroupID` allocation) is handled statically in the prototype, although it is the principal determinant of both efficiency and the degree of anonymization; its systematic design remains open.
- **Fixed data structure.** The prototype always transmits a Bloom filter, irrespective of the revocation rate in the group. At the extremes this is inefficient: at 0% or 100% revoked certificates a single binary value would suffice, and at a very high revocation rate a plain bitmap would be more economical. The selection of the data structure according to the revocation rate was not addressed (a possible direction, Sec. 7).
- **Security and privacy.** We give only an informal threat model (Sec. 3.4), not a formal analysis. The client does not yet verify the response signature, timestamp, or issuer hashes; a hardened version must add these checks and a keyed hash. The anonymization offered is only partial, and the extension OID used is intended solely for research purposes.
- **Measurement scope.** We measured message sizes and cache hits but not client-side processing latency or compute time (the cost of parsing ASN.1 and computing k MD5 hashes versus verifying an OCSP signature), which is important for the IoT and would be readily obtainable on the ESP32. Memory and energy consumption were likewise not quantified; the preliminarily observed reduction in energy could not be reliably captured with the available equipment.
- **Statistical validation.** The size and scalability curves (Figs. 4a and 4b) are means of five runs per configuration and the cache test is a mean of three, which shows the run-to-run spread but

does not establish narrow confidence intervals. Boundary figures such as the 700–1000-certificate range therefore remain indicative.

7. Conclusion and Future Work

We designed, implemented, and evaluated Bloom OCSP, a lightweight protocol for the more efficient transmission of revocation information in the IoT. Bit arrays compressed with a Bloom filter enable a single exchange to convey the status of tens to hundreds of certificates. The protocol was validated, as a *research prototype*, *on an ESP32 device*, confirming its feasibility and applicability on constrained hardware. The verification of a single certificate required at least 66 bytes less than with OCSP, and a message smaller than one OCSP response conveyed information about more than 500 certificates. Because false positives are removed at the responder, responses remain correct, while the group-based query model limits the responder to the granularity of a group rather than the individual certificate in question, a partial, group-level anonymity. The hypotheses that compression increases the number of certificates transmitted per exchange and that group queries partially anonymize the request are therefore confirmed at the prototype level. The protocol is not a finished, production-ready system: several components are deliberately simplified and require further development (Sec. 6). Even in this state, the protocol indicates potential applications, for example in connected vehicles, drone swarms, and sensor networks.

The principal opportunity lies in making the data structure flexible. The protocol need not be restricted to a single Bloom filter; in ASN.1 it could be expressed as a `CHOICE` that selects the most economical representation for the group's revocation rate. At 0% or 100% revoked certificates, a single binary value suffices. A Bloom filter is used where it is advantageous, and by inverting its meaning (encoding valid rather than revoked certificates) it can be used economically even at very high revocation rates. Over the remaining intermediate range, in which the Bloom filter is least advantageous, a plain bitmap would be used. The protocol would then cover the entire range of revocation rates with the smallest representation at each point.

A second direction is the domain-specific optimization of the Bloom filter itself by means of common representatives. Because the group identifier already determines the set of certificates to be verified, knowledge that has so far been used only to eliminate false positives at the responder, the same knowledge could be used to construct common identifiers for several certificates. This is an established technique for optimizing Bloom filters (IP-address prefixes, or pairs of k-mers in bioinformatics). Suitably chosen and encoded common identifiers would increase the tolerable false-positive rate and further reduce the total size of the revocation information, while also narrowing the range of revocation rates over which a plain bitmap remains necessary; this could increase the maximum advantageous group size from hundreds to thousands of certificates.

The remaining directions are incremental. The insights from Bloom OCSP could be combined with the removal of redundant fields from TinyOCSP [2] to obtain a protocol that benefits from both. A more detailed investigation of the false-positive elimination scheme (alternative hash functions and seeds, and the step by which the target rate is tightened, fixed here at a conservative 0.9 whose sensitivity we did not study) could reduce the recomputation cost at the responder. The preliminarily observed reduction in energy consumption, which the available measurement equipment could not reliably capture, also warrants confirmation. Finally, a comparison with deterministic alternatives based on cryptographic accumulators [14], which avoid false positives at the cost of heavier cryptography, and the addition of stronger privacy guarantees where the partial anonymization of a group query is insufficient, merit investigation.

A final direction concerns dissemination. Because Bloom OCSP retains the signed OCSP response, each group response is a compact, self-contained object that any device can verify independently and that resolves revocation queries for the entire group. In IoT networks with intermittent connectivity it could be propagated device-to-device, so that a single fetch from the responder serves a whole neighborhood rather than a single device, in the spirit of the peer dissemination that EVOKE [14] uses

for accumulator updates. Such opportunistic sharing would extend the caching benefit of Sec. 5 to multi-hop and offline settings; we have neither implemented nor evaluated it, and leave it for future work.

Acknowledgments

This research was supported by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 17R05-04-V01-00006 and by project KEGA No. 004ZU-4/2024 “Improving the quality of education in the field of cyber security” and by the Youth grant of the STU No. 1350 “A service for efficient acquisition of revocation data in IoT”.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT-4o in order to: grammar and spelling check, improve the writing style, and assist with the visual editing and formatting of the figures and layout. The tool was used solely for these language and visual refinements, and not for generating the research content, methods, or results. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

[1] C. Bormann, M. Ersue, A. Keränen, Terminology for Constrained-Node Networks, Technical Report RFC 7228, Internet Engineering Task Force, 2014. doi:10.17487/RFC7228.

[2] J. Höglund, M. Furuhed, S. Raza, Lightweight certificate revocation for low-power IoT with end-to-end security, Journal of Information Security and Applications 73 (2023) 103424. doi:10.1016/j.jisa.2023.103424.

[3] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, D. Cooper, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Technical Report RFC 5280, Internet Engineering Task Force, 2008. doi:10.17487/RFC5280.

[4] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, Technical Report RFC 6960, Internet Engineering Task Force, 2013. doi:10.17487/RFC6960.

[5] S. Galperin, M. Myers, R. Ankney, C. Adams, A. Malpani, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, Technical Report RFC 2560, Internet Engineering Task Force, 1999. doi:10.17487/RFC2560.

[6] A. A. Chariton, E. Degkleri, P. Papadopoulos, P. Ilia, E. P. Markatos, CCSP: A compressed certificate status protocol, in: IEEE INFOCOM 2017 – IEEE Conference on Computer Communications, 2017, pp. 1–9. doi:10.1109/INFOCOM.2017.8057065.

[7] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. Maggs, A. Mislove, A. Schulman, C. Wilson, An end-to-end measurement of certificate revocation in the web’s PKI, in: Proceedings of the 2015 Internet Measurement Conference (IMC ’15), 2015, pp. 183–196. doi:10.1145/2815675.2815685.

[8] A. Langley, Revocation checking and chrome’s CRL, 2012. URL: https://www.imperialviolet.org/2012/02/05/crlsets.html.

[9] M. Goodwin, Revoking intermediate certificates: Introducing OneCRL, 2015. URL: https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl.

[10] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, C. Wilson, CRLite: A scalable system for pushing all TLS revocations to all browsers, in: 2017 IEEE Symposium on Security and Privacy (SP), 2017, pp. 539–556. doi:10.1109/SP.2017.17.

[11] B. H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (1970) 422–426. doi:10.1145/362686.362692.

[12] P. C. Kocher, On certificate revocation and validation, in: Financial Cryptography (FC 1998), volume 1465 of *Lecture Notes in Computer Science*, Springer, 1998, pp. 172–177. doi:10.1007/BFb0055481.

[13] D. Koisser, P. Jauernig, G. Tsudik, A.-R. Sadeghi, V'CER: Efficient certificate validation in constrained networks, in: 31st USENIX Security Symposium (USENIX Security 22), USENIX Association, 2022. URL: https://www.usenix.org/conference/usenixsecurity22/presentation/koisser.

[14] C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, EVOKE: Efficient revocation of verifiable credentials in IoT networks, in: 33rd USENIX Security Symposium (USENIX Security 24), USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/mazzocca.

[15] C. Bormann, A. P. Castellani, Z. Shelby, CoAP: An application protocol for billions of tiny internet nodes, IEEE Internet Computing 16 (2012) 62–67. doi:10.1109/MIC.2012.29.

[16] Mbed-TLS Project, Mbed-TLS/mbedtls, 2025. URL: https://github.com/Mbed-TLS/mbedtls.

[17] wolfSSL Inc., About wolfSSL, 2025. URL: https://www.wolfssl.com/about/.

[18] J. Pajkos, E. Kupcová, M. Pleva, M. Drutarovský, ESP32 microcontroller based lightweight TLS 1.3 client for IoT applications, in: 2025 35th International Conference Radioelektronika (RADIOELEKTRONIKA), 2025. doi:10.1109/RADIOELEKTRONIKA65656.2025.11008381.

[19] H. Kwon, D. Kim, H. Kim, C. Hahn, J. Hur, Certificate revocation in the TLS ecosystem: A survey, ACM Computing Surveys 58 (2026) 178:1–178:36. doi:10.1145/3785653.

[20] M. Mislovič, Revocation of a public key certificate in the Internet of Things, Diploma Thesis, Faculty of Informatics and Information Technologies of the Slovak University of Technology, 2026. URL: https://opac.crzp.sk/?fn=detailBiblioFormChildI87P8&sid=2C85915F14684159D94BF3C70391&seo=CRZP-detail-kniha.

[21] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis, 2nd ed., Cambridge University Press, 2017.

[22] M. Mislovič, Bloom OCSP: Proof-of-concept implementation, Source code, 2026. URL: https://github.com/MartinMis/Bloom-OCSP.

[23] Espressif Systems, ESP-IDF getting started, 2025. URL: https://idf.espressif.com/.

[24] N. Korzhitskii, N. Carlsson, Revocation statuses on the Internet, in: Passive and Active Measurement (PAM 2021), volume 12671 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 175–191. doi:10.1007/978-3-030-72582-2_11.