Privacy-Preserving Revocation Checking with Modified CRLs

Alice and Bob
Computer Science Department
University of Anonymity

Abstract

Certificate Revocation Lists (CRLs) are the most popular means of revocation checking. A CRL is essentially a signed and timestamped list containing information about all revoked certificates issued by a certification authority. One of the shortcomings of CRLs is poor scalability which influences update, bandwidth and storage costs. Other, more efficient revocation techniques leak potentially sensitive information. Information leaks occur since third parties (agents, servers) of dubious trustworthiness discover the identities of the parties posing revocation check queries as well as identities of the queries' targets. An even more important privacy loss results from the third party's ability to tie the source of the revocation check with the query's target. (Since, most likely, the two are about to communicate.) This paper focuses on privacy and efficiency in revocation checking. Its main contribution is a simple modified CRL structure that allows for efficient revocation checking with customizable level of privacy.
Keywords: Anonymity and Privacy, Certificate Revocation.

1 Introduction and Motivation

Public key cryptography allows entities to establish secure communication channels without pre-established shared secrets. While entities can be assured that communication is secure, there is no guarantee of authenticity. Authenticity is obtained by binding a public key to some claimed identity or name which is later verified via digital signatures in conjunction with public key certificates (PKCs). A public key certificate, signed by a recognized certification authority (CA), is used to verify the validity, authenticity and ownership of a public key. As long as the issuing CA is trusted, anyone can verify the CA's certificate signature and bind the included name/identity to the public key. Public key certificates work best in large interconnected open systems, where it is generally infeasible to directly authenticate the owners of all public keys. X.509 [20] is one well-known certificate format widely used in several Internet-related contexts. The peer-based PGP/GPG [2, 5] format represents another popular approach.

Since a certificate is a form of a capability, one of the biggest problems associated with large-scale use of certificates is *revocation*. There are many reasons that can lead to a certificate being revoked prematurely. They include [20]: loss or compromise of a private key, change of affiliation or job function, algorithm compromise, or change in security policy. To cope with revocation, it must be possible to check the status of any certificate at any time.

Revocation techniques can be roughly partitioned into implicit and explicit classes. In the former, each certificate owner possesses a timely proof of non-revocation which it supplies on demand to anyone. Lack of such a proof implicitly signifies revocation. An example of implicit revocation is Micali's Certificate Revocation System (CRS) [14]. Most revocation methods are explicit, i.e., they involve generation, maintenance and distribution of various secure data structures that contain revocation information for a given CA or a given range of certificates.

Certificate Revocation Lists (CRLs) represent the most widely used means of explicit revocation checking. Each certificate issuer periodically generates a signed list of revoked certificates and publishes it at (usually untrusted) public directories or servers. Inclusion of a certificate in the list signifies explicit revocation. Verifiers retrieve and cache the latest CRL and use it during certificate validation. Typically, a revoked certificate is included in a CRL from the time it is revoked until its validity period expires. Since certificate lifetime is typically measured in years, even modest revocation rates can result in very long CRLs. In bandwidth-constrained environments, transferring such CRLs can be expensive. Furthermore, since CRLs are published periodically, another potential concern is that many verifiers may request them around the time of publication. The burst of requests immediately following CRL publication may result in very high network traffic and can cause congestion. Thus, one of the biggest disadvantages of CRLs is the high cost associated with updating and querying the lists and this raises serious scalability concerns.

Other well-known explicit revocation methods include Certificate Revocation Trees (CRTs) [10] and Skip-Lists [6]. Another prominent technique is the On-line Certificate Status Protocol (OCSP) [15] which involves a multitude of validation agents (VAs) which respond to client queries with signed replies indicating current status of a target certificate. However, these explicit revocation methods have an unpleasant side-effect: they divulge too much information. Specifically, a third party (agent, server, responder or distribution point) of dubious trustworthiness knows: (1) the entity requesting the revocation check (source), and (2) the entity whose status is being checked (target). An even more important **loss of privacy** results from the third party tying the source of the revocation checking query to that query's target. This is significant, because revocation status check typically serves as a prelude to actual communication between the two parties.[1]

In the society preoccupied with the gradual erosion of electronic privacy, loss of privacy in current revocation checking is an important issue worth considering. Consider, for example, certain countries with less-than-stellar human rights records where mere intent to communicate (indicated by revocation checking) with an *unsanctioned* or *dissident* web-site may be grounds for arrest or worse. In the same vein, sharp increase in popularity (deduced from being a frequent target of revocation checking) of a web-site may lead authorities to conclude that

[1]We assume that communication between verifiers and on-line revocation agents (third parties) is private, i.e., conducted over secure channels protected by tools such as IPSec [8] or SSL/TLS [7].

something *subversive* is going on. Clearly, the problem can also manifest itself in other less sinister settings. For example, many internet service providers already keep detailed statistics and build elaborate profiles based on their clients' communication patterns. Current revocation checking methods – by revealing sources and targets or revocation queries – represent yet another source of easily exploitable (and mis-used) personal information.

Hiding sources of revocation queries can be easily achieved with modern anonymization techniques, such as onion routing, anonymous web browsing or remailers. While this might protect the source of a revocation query, the target of the query remains known to the third party. Furthermore, although anonymization techniques are well-known in the research community, their overall penetration remains fairly low. Also, in order to take advantage of an existing anonymization infrastructure, one either needs to place some trust in unfamiliar existing entities (e.g., remailers, re-webbers or onion routers) or make the effort to create/configure some of these entities.

Contributions: In this work, we propose modifications to the well-known CRL structure. These modifications enable efficient and privacy-preserving revocation checking. Specifically, we provide a mechanism for verifiers to query untrusted online entities regarding the revocation status of a certificate without revealing to the latter information about the actual certificate of interest. Privacy requirements can be tuned to achieve verifier-specific level of privacy. This is achieved without the need for verifiers to retrieve the entire CRL.

Organization: The remainder of the paper is organized as follows: Section 2 discusses some popular revocation techniques. We discuss relevant prior work in section 3. Section 4 describes the details of our technique. We discuss some practical implications of the proposed modifications in section 5. We analyze the scheme in section 6 and present the various overhead costs. We outline some future directions and conclude in section 7.

2 Certificate Revocation Techniques

In this section, we briefly overview some popular certificate revocation techniques and associated data structures. In the following, we refer to the entity validating certificates (answering certificate status queries) as a Validation Authority (VA). A distinct entity – Revocation Authority (RA) – is assumed responsible for actually revoking certificates, i.e., generating signed data structures, such as CRLs.

Certificate Revocation Lists (CRLs): CRLs are the most commonly used means of checking the revocation status of public key certificates. A CRL is a signed list of certificates that have been revoked before their scheduled expiration date. Each entry in the CRL indicates the serial number of the certificate associated with the revoked public key. The CRL entry may also contain other relevant information, such as the time, and reason for the revocation. CRLs are usually distributed in one of two ways: In the "pull" model, RA distributes the CRLs via a public directory. The clients/queriers download the CRL from these public databases as needed. In the "push" model, the RA directly sends the CRL to the clients at regular intervals. If CRLs are distributed using a pull model, they should be issued at regular intervals even if there are no changes, to prevent new CRLs being maliciously replaced by old CRLs. Since a CRL can get quite long, a RA may instead post a signed Δ-CRL which contains only the new entries consisting of the list of certificates revoked since the last CRL was issued. This requires end-users maintain (and update) secure local images of the CRL. [12] lists some of the other proposed ways to improve the operational efficiency of the CRLs such as Segmented CRLs and CRL distribution points.

Online Certificate Status Protocol (OCSP): this protocol [15] avoids the generation and distribution of long CRLs and provides more timely revocation information. To validate a certificate in OCSP, a client sends a certificate status request to a VA. The latter sends back a signed response indicating the status (revoked, valid or unknown) of the specified certificate. Note that, in this case, the VA is

- the CA who issued the certificate in question, or
- a Trusted Responder whose public key is trusted by the client, or
- a CA Designated Responder (Authorized Responder) who is authorized to issue OCSP responses for that CA.

In other words, the VA is an on-line authority trusted by both the client and the CA. (A VA can also serve multiple CAs.)

Certificate Revocation Trees (CRTs): this technique was proposed by Kocher [10] as an improvement for OCSP [10]. Since the VA is a global service, it must be sufficiently replicated in order to handle the load of all validation queries. This means the VA's signature key must be replicated across many servers which is either insecure or expensive. (VA servers typically use tamper-resistance to protect their signing keys). Kocher's idea is a single highly secure RA which periodically posts a signed CRL-like data structure to many insecure VA servers. Users then query these insecure VA servers. The data structure proposed by Kocher is basically a Merkle Hash Tree (MHT) [13] where the leaves represent currently revoked certificates sorted by serial number (lowest serial number is the left-most leaf and the highest serial number is the right-most leaf). The root of the hash tree is signed by the RA. A client wishing to validate a certificate issues a query to the nearest (untrusted) VA server which, in turn, produces a short proof that the target certificate is (or is not) on the CRT. If n certificates are revoked, the length of the proof is $O(\log n)$ (In contrast, the proof size in plain OCSP is $O(1)$).

Skip-lists and 2-3 trees: One problem with CRTs is that, each time a certificate is revoked, the whole tree must be recomputed and distributed in its entirety to all VA servers. A data structure allowing for dynamic updates would solve the problem since a secure RA would only need to send small updates to the data structure along with a signature on the new root of the structure. Both 2-3 trees proposed by Naor and Nissim [17] and skip-lists proposed by Goodrich, et al. [6] are natural and efficient for this purpose. Additional data structures were proposed in [1]. When a total of n certificates are already revoked and k new certificates must be revoked during the current time period, the size of the update message to the VA servers is $O(k \log n)$ (as opposed to $O(n)$ with CRT's). The proof of certificate's validity is of size $O(\log n)$, same as with a CRT.

3 Related Work

The most closely related prior work is the recent paper [18] which proposes privacy-preserving extensions to Certificate Revocation Trees. In this work, instead of specifying the actual target certificate, a revocation query includes a range of certificates serial numbers. (The range includes the target certificate.) In this paper, we use a similar approach to achieve privacy with CRLs. As observed in [18], CAs typically assign consecutive serial numbers to consecutively issued certificates and groups of related certificates (e.g., issued to the same company) have consecutive serial numbers. Thus, information leaks can occur in cases when the query range subsumes or contains a large block of related consecutive certificate serial numbers. To mitigate this issue, [18] proposes sorting leaf nodes along *permuted* serial numbers. One example of a sufficiently random permutation involves using a block cipher (such as AES) with a fixed publicly known key.

Another result which considered privacy in revocation checking was the work by Kikuchi [9]. This work originally identified the problem and proposed a fairly heavy-weight (inefficient) cryptographic technique specific to CRLs. The solution relies on cryptographic accumulators which are quite expensive.

A related research topic is Private Information Retrieval (PIR) [3, 11]. PIR refers to a set of cryptographic techniques and protocols that – in a client-server setting – aim to obscure the actual target(s) of database queries from malicious servers. Although PIR techniques could be applicable in our context, they tend to be relatively inefficient owing to either (or both) many communication rounds/messages or expensive cryptographic operations. As will be seen in subsequent sections, PIR techniques would amount to overkill in the context of privacy-preserving revocation checking.

4 Privacy-Preserving CRL Querying

We start by noting that a CRL trivially meets the privacy requirement, since in the typical usage scenario, a verifier simply downloads and stores the entire CRL. This allows it to locally perform any number of revocation checks against any certificate issued by a particular CA. However, downloading and caching a large CRL in its entirety is neither bandwidth nor storage efficient for the client. Although delta-CRLs tend to be smaller and more bandwidth-efficient, they still impose high storage overhead on the client. In addition, a client is required to periodically download the base-CRL as well as all subsequent delta-CRLs in order to keep the CRL cache recent and accurate. In this section, we propose a new CRL model and outline a mechanism which overcomes the aforementioned drawbacks, while allowing efficient and private revocation checking.

In our model, a trusted RA periodically generates CRLs and publishes them at untrusted online VA-s. A client interested in validating a certificate queries a VA to obtain authentic revocation information corresponding to that certificate. Although this model bears similarities to the OCSP method, unlike OCSP, the VA-s in our model can be third party responders who are not trusted by either clients or RA-s.

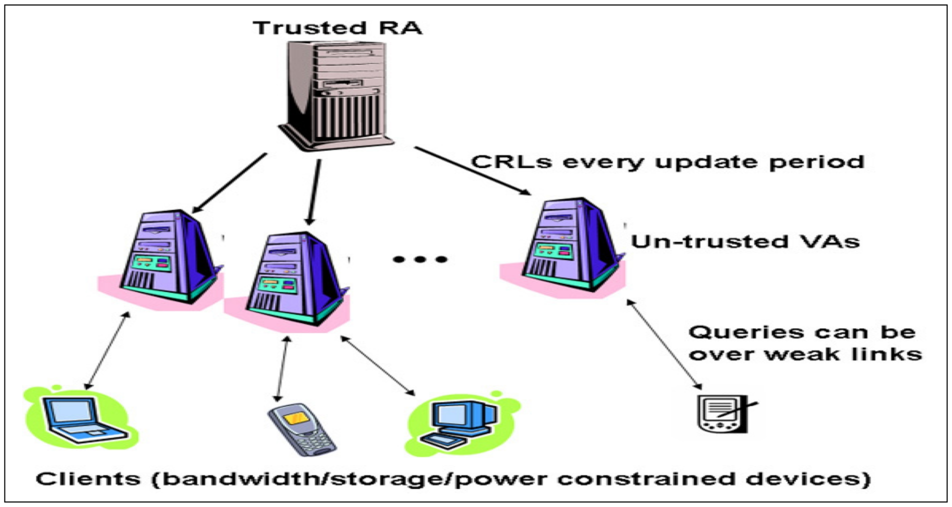
Trusted RA
CRLs every update period
Un-trusted VAs
Queries can be over weak links
Clients (bandwidth/storage/power constrained devices)

Figure 1: System Overview

Since VA-s are untrusted, it is essential to ensure the integrity and authenticity of the revocation information of a certificate with respect to the RA responsible for that certificate. To achieve this, the RA individually signs each CRL entry. In other words, information regarding each revoked certificate is separately signed by the RA and placed onto the CRL. This differs from the usual CRL structure where all revoked certificates are included in the CRL which is then collectively signed once by the RA. Our modificaton clearly introduces some additional burden for the RA who is now required to sign each entry. However, we claim that this modification makes certificate revocation checking very efficient for clients; at the same time, it is not prohibitively expensive for a typical RA which is a computationally-powerful machine. (We analyze actual overheads in section 6.) This relatively minor modification obviates the need for the client to download the entire CRL.

4.1 (Non-Privacy-Preserving) Revocation Checking

When a VA receives a revocation status query from a client, it checks to see if there is a CRL entry corresponding to that certificate. If it exists, the VA sends back a **Revoked** reply along with the CRL entry signed by the RA, as the proof. However, if the certificate in question is not revoked, then simply sending a **Not Revoked** reply does not provide integrity and authenticity guarantees with respect to the RA[2]. To achieve authenticity of **Not Revoked** replies, we suggest chaining of CRL entries, as described below.

Signature Chaining: To provide authenticity of query replies, the RA securely links the signatures of the revoked certificates to form a so-called *signature chain*. To construct a signature chain, the RA generates the individual signature for each revoked certificate in the following way:

Definition 1

$$Sign(i) = h(h(i)||h(IPC(i)))_{SK}$$

where i is the (permuted) serial number of a revoked certificate, $h()$ is a suitable cryptographic hash function, $||$ denotes concatenation, IPC denotes the immediate predecessor certificate and SK is the private signing key of the RA.

The immediate predecessor is a revoked certificate with the highest serial number which is less than the serial number of the current revoked certificate. In other words, the RA sorts all revoked certificates in ascending order by serial number and computes the signature of each revoked certificate by including the hash of the immediate predecessor, thereby explicitly chaining (linking) all signatures.

With signatures chained in the above fashion, when a client queries the status of an unrevoked certificate, the VA composes an **Not Revoked** reply by returning the two boundary certificates $CERT_a$ and $CERT_b$ that subsume the non-existent serial number along with the signature of $CERT_b$.

4.2 Privacy Preserving Revocation Checking

Our CRL modification allows a client to query the VA for revocation status of a certificate and obtain a concise and authentic proof of either revocation or validity. However, this introduces privacy concerns since clients now query

[2]Recall that VAs are untrusted in our model, and a malicious or lazy VA might not properly execute the query and send incorrect reply to the client.

the VA by a specific certificate serial number, thus revealing the identity of the target entity. To address this problem we employ the same technique as in [18]. Instead of querying by a specific certificate serial number i, we propose querying by a randomly selected range of serial numbers (j, k) with $j \leq \ i \leq \ k$. This effectively hides the certificate of interest. The only information divulged to the VA is that the target certificate lies in the interval $[j, k]$. This translates into the probability of correctly guessing i as: $P_i = \frac{1}{k-j+1}$. Each number in the range is equally likely to be the serial number of interest and the VA has no means, other than guessing, of determining the target certificate. Furthermore, the VA has no way of telling whether the actual query target is a revoked or a valid certificate.

Let n be the total number of certificates and m be the number of revoked certificates. Then, assuming uniform distribution of revoked certificate serial numbers over the entire serial number range, m/n is the fraction of revoked certificates and the very same fraction would be revoked within any $[j, k]$ range.

Clearly, perfect privacy is not attainable with range queries. The highest possible privacy is $1/n$ which corresponds to querying the full certificate serial number range.[3] The lowest privacy level corresponds to querying by a specific serial number, i.e., setting $j = k = i$. The optimal query range is determined by the source of the query, i.e., the client. Several factors must be taken into account: (1) desired level of privacy e.g., if $k - j + 1 = 1000$, the probability of correctly guessing i is P_i= 0.001, (2) additional bandwidth and storage overhead stemming from returning a set of CRL entries as a reply.

Once the range size (r) is determined, the client proceeds to set the actual range boundaries: j and k. To do so, it first generates a b-bit random number X where $b = log(r)$ or the bit-length of r. X determines the position, within the range, of the actual target certificate serial number. This step is necessary to randomize/vary the placement of the target. Next, the boundaries are set as: $j = i - X$ and $k = j + r - 1$. The client poses a query to the VA asking for the revocation status of all certificates within $[j, k]$. Since each entry corresponding to a revoked certificate in the $[j, k]$ range is individually signed by the RA, it seems that the VA would need to send all signatures to the verifier. This is clearly inefficient for the verifier. However, we can modify the signature generation process as follows:

> Sort all revoked certificates in ascending order of permuted serial numbers to obtain $\{CERT_0, CERT_1, CERT_2, \ldots, CERT_m, CERT_{m+1}\}$. The two "dummy" boundary values: $CERT_0$ and $CERT_{m+1}$ represent $-\infty$ and $+\infty$, respecively, i.e., they denote values outside the allowed range. Now, compute the signature of each revoked certificate by signing the **running hash** of all certificates in the chain from the first entry to the current certificate:
>
> **Definition 2**
>
> $$Sign(CERT_i) = h(CERT_i || h(CERT_{i-1} || \ldots || h(-\infty))_{SK}$$

[3]This is equivalent to obtaining an entire CRL.

where $h()$ is a suitable cryptographic hash function, $||$ represents concatenation and SK is the private (signing) key of the RA.

The resulting CRL is stored at the VA-s as before.

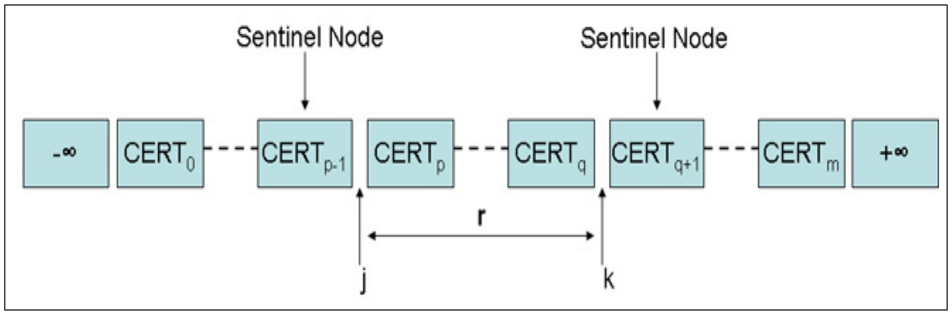

Figure 2: Revocation Query Reply

To assert revocation status of all the certificates in the range $[j, k]$, a VA releases the revoked certificates $[CERT_p, \ldots, CERT_q]$ in the actual range $[j, k]$, two sentinel nodes $CERT_{p-1}$ and $CERT_{q+1}$ just outside the range to prove completeness of the reply, running hash for $CERT_{p-2}$, and a **single** signature – $Sign(CERT_{q+1})$. Since all signatures are computed on running hashes, it can be easily seen that $Sign(CERT_{q+1})$ provides a proof of authenticity and completeness for the entire range.

5 Practical Considerations

We now discuss some practical implications of the proposed modifications.

5.1 CRL Generation and Size

With the technique described above, the RA signs revoked certificates separately. This increases the overall size of the CRL as well as the computation load on the RA. However, we claim that this overhead is acceptable, for the following reasons:

- Although it might seem that computational overhead of signing each revoked certificate might be significant for the RA, we show that this is not the case in practice. Experiments show that it takes 6.82ms to generate one RSA signature on a "weakling" P3-977MHZ linux PC. Assuming that the update interval for the CRL is one week and the CRL contains $100,000$ certificates, this translates to 11.3 minutes of compute time for every CRL update period. Clearly, the same task would take much less (one or two orders of magnitude) time on more powerful platforms readily available today. For example, the Sun Fire T2000 server can compute 12,850 1024-bit RSA signatures and 18,720 1024-bit DSA signatures in one second with 1 UltraSPARC T1 processor and 32 GB memory running Solaris 10[19]. Going back to our example scenario, if the RA were to be deployed on a T2000 server, it could recompute $100,000$ signatures in just under 10 seconds.

– The proposed technique requires storing a separate signature per CRL entry. This increases the size of the CRL and, consequently, increases the cost of RA-VA communication. However, it is reasonable to assume that this communication is conducted over fast communication links, in contrast to VA-Client communication which can take place over low-bandwidth channels. Furthermore, RA-VA communication occurs relatively infrequently (once per update period), whereas the frequency of clients-VA communication is signficicantly higher. We present sample metrics for communication costs in the next section.

5.2 Freshness Considerations

In order to provide freshness and to prevent malicious VA-s from replaying old CRL-s, the RA must re-sign revoked certificates every update period, even if there are no changes to the CRL. We can reduce the time to re-sign the revoked signatures in the event of no changes to the CRL by using the Condensed RSA signature scheme proposed in [16].

Condensed-RSA Signature Scheme: Given t different messages $\{m_1, ..., m_t\}$ and their corresponding signatures $\{\sigma_1, ..., \sigma_t\}$ generated by the same signer, a Condensed-RSA signature is computed as the product of all t individual signatures:

$$\sigma_{1,t} = \prod_{i=1}^{t} \sigma_i \pmod{n}$$

The resulting aggregated (or condensed) signature $\sigma_{1,t}$ is of the same size as a single standard RSA signature. Verifying an aggregated signature requires the verifier to multiply the hashes of all t messages and checking that:

$$(\sigma_{1,t})^e \equiv \prod_{i=1}^{t} h(m_i) \pmod{n}$$

Condensed-RSA for Re-signing: It is possible to use condensed-RSA for re-signing if the RA signs the revoked certificate data and the timestamp separately and then aggregates the two by multiplying the resultant signatures as explained above. If there are no changes to the CRL, the RA can re-sign all the revoked certificates by: (1) re-using the signatures for the revoked certificate data; (2) creating a single new signature on the new timestamp; and (3) aggregating the signature of the new timestamp with the signatures of all the revoked certificates to re-sign all the revoked certificates.

If we assume, once again, that the CRL contains $100,000$ entries. As mentioned above, it woud take 11.3 minutes to re-sign all these certificates with the new timestamp on a P3-977MhZ Linux machine. However, if we use condensed-RSA, re-signing of the same $100,000$ entries with the new timestamp, using our method can be done in just 4.4 seconds on the same hardware. However, this optimization is less useful if there are changes to the CRL. Because signatures are chained explicitly, any change to the CRL (insertion or deletion of an entry from the chain) causes all the signatures from that point until the end of the chain to be re-computed.

5.3 Forcing Uniform Distribution

In a worst-case scenario, all certificates in the range specified in the query are revoked and corresponding r certificates need to be returned to the client. The simplest solution to this is to force all revoked certificate serial numbers to be uniformly distributed over the entire serial number range. However, this is unrealistic, since, in practice, certificate issuers assign serial numbers to certificates consecutively over well-defined subranges. Each subrange can be used to indicate a different product or class of products, e.g., VeriSign supports the following classes: Standard, Commerce and Premium [4]. Requiring uniform non-sequential certificate distribution would create a maintenance nightmare for both issuers and certificate-holders. Furthermore, gathering and analysis of statistical data would become problematic.

We use a simple extension to the range query technique that addresses the problem. Instead of sorting according to serial numbers, we sort the revoked certificates along *permuted* serial numbers before creating the signature chain. One obvious choice of suitable permutation that ensures uniformity is a block cipher, e.g., DES, with a known and fixed key. Note that the brute-force resistance of the block cipher is not important here. The only issue of concern is the block cipher's quality – for a fixed key – as a pseudo-random permutation (PRP). Ideally, the space of all possible serial numbers would match the set of all possible block cipher outputs. For example, DES-ECB mode outputs 64-bit blocks which matches the size of certificate serial numbers for many X.509v3 CAs. However, the underlying permutation can be resolved with any good block cipher, such as Blowfish or AES. We further observe that a cryptographic hash function is not a good choice for the kind of a PRP we require. Unlike a PRP, a hash function "reduces" its input and collisions are expected, however difficult they might be to compute. Whereas, a PRP resolved with a block cipher such as DES-ECB with a fixed key, guarantees no collisions.

The primary advantage of this extension is that certificate issuers can continue issuing sequentially-numbered certificates over well-defined subranges. As long as an appropriate PRP is used, we can assure uniform distribution of the revoked certificates in the signature chain.

5.4 Query Range Generation

In section 4.2, we outlined a mechanism for positioning the target certificate within the query range by generating a b-bit random number $b = log(r)$ or the bit-length of r, where r is the range of the query. This step is necessary to randomize/vary the placement of the target. We observe that, if a client poses repeated queries against the same target certificate, varying the query range and its boundaries is not advisable, for obvious reasons. In this case, narrowing the overlap of all queries' ranges gradually erodes privacy and might eventually yield a single target certificate. We now discuss one possible solution to eliminate such inference attacks.

Instead of generating a random number (offset) to determine the position of the target certificate within the range, the client generates a b-bit number using a *keyed hash function* h' which takes as input the target certificate serial number i and a secret key sk chosen by the client. $h'(i, sk) = X$ where $|X| = b$. Now, the client sets the range boundaries as described earlier, i.e., $j = i - X$ and $k = j + r - 1$. The client then poses a revocation query to the VA with the range $[j, k]$. The use of the secret key sk to compute range boundaries ensures that, for a given target certificate, the same range is consistently used for repeated queries.

6 Analysis

We now briefly analyze the various overhead factors introduced by our privacy-preserving revocation checking techniques.

Client Overhead: With traditional CRLs, clients are required to store entire lists containing all revoked certificates locally. As such lists grow, this can result in significant storage overhead. On the other hand, with our technique, clients do not incur any storage costs. A client queries the on-line VA for the revocation status of a certificate and does not need to store/cache anything locally.

For traditional CRL, the client is required to periodically contact a CRL distribution point (or directory server) to obtain the most recent CRL. If Δ-CRLs are used, the client needs to periodically download updates in order to keep her copy of the CRL recent and complete. Thus, communication overhead using traditional CRLs and delta-CRLs depends upon the frequency of CRL updates. On the other hand, with our proposed technique, a client incurs one round of communication with the VA for each revocation status check of a certificate.

RA Overhead: In our method, the RA is required to separately sign each entry in the list, whereas, in a traditional CRL, the entire CRL is signed once in its entirety, for each update. This increases both the size of the CRL as well as the computation costs for the RA. However, as discussed in the previous section, we assume that the frequency of querying is much greater than the frequency of

CRL updates; thus, we focus our scheme on minimizing costs for the interaction between the VA and its clients.

Communication Costs: We now compare communication costs of the traditional CRL with our modified version. Table 1 summarizes the notation and parameter values assumed for the evaluation. **Update cost** measures RA-to-VA communication and **query cost** measures VA-to-Client communication.

n	# of Certificates $n = 2 * 10^6$
p	Fraction of the certificates revoked $p = 0.1$
t	update period t=weekly
c	# of clients $c = 10^5$
q'	# of queries posed by each client $q' = 100$
q	Total # of queries in an update period $q = c * q' = 10^7$
r	Size of the range query
l_{sig}	Length of a signature $l_{sig} = 1024$ bits
l_{entry}	Length of a CRL entry $l_{entry} = 160$ bits
l_{hash}	Length of a hash digest $l_{hash} = 160$ bits

Table 1: Notation

Traditional CRL: The CRL update cost is $n * p * l_{entry} + l_{sig}$ for each update period since the RA sends the entire CRL to the VAs. We are assuming that there are c clients and each client poses q' certificate validation checks on the cached CRL in a update period. The CRL weekly query cost is $c * (p * n * l_{entry} + l_{sig})$ since for every query the VA sends the whole CRL to the client.

Modified Technique: To update the CRL, the RA sends $n * p * (l_{entry} + l_{sig})$ bits to the VA. To answer a user's query, the VA returns $l_{entry} + l_{sig}$ for a **Revoked** reply and $2l_{entry} + l_{sig}$ for a **Not Revoked** reply if queried by a specific serial number (i.e., for non-privacy preserving querying). Therefore, the VA sends upto $q * (l_{entry} + l_{sig})$ bits each update period to answer q queries. If we employ range queries to enable privacy-preserving querying, then the communication cost depends on the number of revoked certificates in the range. In general, if we assume uniforn distribution of revoked certificates, then the communication cost for user queries for a range of certificates is given by $q * (((r * p) + 2) * l_{entry} + l_{sig})$ bits for each update period. Note that $(r * p) + 2$ denotes the total number of revoked certificates in the specified range plus two sentinel nodes.

The following table shows the estimated communication costs (in bits) according to traditional as well as modified CRL schemes. As shown in the table, the modified CRL query costs are orders of magnitude lower than traditional

CRL costs. Although the cost of updating a CRL is higher in our scheme, the advantage of our scheme is that it allows the clients to query and obtain portions of the CRL securely thus making our scheme flexible, efficient (for the clients) and privacy preserving.

	Traditional CRL	Modified CRL no privacy	Modified CRL r=100
Update Cost (RA-to-VA)	$3.2 * 10^7$	$2.36 * 10^8$	$2.36 * 10^8$
Query Cost (RA-to-Client)	$3.2 * 10^{12}$	$1.18 * 10^{10}$	$2.94 * 10^{10}$

Table 2: Communication Cost comparison

7 Future Directions and Conclusions

An outstanding issue is assessing loss of privacy in the presence of repeated queries. If we assume that multiple clients, at about the same time, are all interested in a particular target certificate (e.g., because of a breaking news article) and the adversary (third party or VA) is aware of the potential target, co-relating multiple range queries does not seem difficult since all the range queries in question would have at least one certificate in common. As part of our future work, we want to study this problem in greater detail to make such inferences more difficult.

In conclusion, we described a very simple (yet novel) approach for privacy-preserving revocation checking using CRLs. We proposed minor modifications to the well-known CRL structure. These modifications enable efficient (for the clients) and privacy-preserving revocation checking. We provided a mechanism for verifiers to query untrusted online entities regarding the revocation status of a certificate without having to retrieve the entire CRL. Each client, depending on the desired level of privacy, can determine a revocation query range that best suits its needs. This results in a trade-off between privacy and bandwidth overhead. In the worst case, the overhead can be significant if the desired privacy level is high as is the number of revoked certificates. However, if only a small fraction of all certificates are revoked, our approach is reasonably efficient.

References

[1] W. Aiello, S. Lodha, and R. Ostrovsky. Fast digital identity revocation. In Hugo Krawczyk, editor, *Proceedings of Crypto'98*, number 1462 in LNCS. IACR, Springer Verlag, 1998.

[2] The OpenPGP Alliance. Openpgp: Open pretty good privacy, `http://www.openpgp.org/`.

[3] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylog communication. In *Proceedings of Eurocrypt'99*, LNCS. IACR, Springer Verlag, 1999.

[4] Verisign Corporation. Compare all ssl certificates from verisign, inc. `http://www.verisign.com/products-services/security-services/ssl/buy-ssl-certificates/compare/index.html`.

[5] Inc. Free Software Foundation. Gnu privacy guard, `http://www.gnupg.org/`.

[6] M. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proceedings of DARPA DISCEX II*, 2001.

[7] OpenSSL User Group. The openssl project web page, `http://www.openssl.org`.

[8] S. Kent and R. Atkinson. Security architecture for the internet protocol. Internet Request for Comments: RFC 2401, November 1998. Network Working Group.

[9] H. Kikuchi. Privacy-preserving revocation check in pki. In *2nd US-Japan Workshop on Critical Information Infrastructure Protection*, pages 480–494, July 2005.

[10] P. Kocher. On certificate revocation and validation. In *Proceedings of Financial Cryptography 1998*, pages 172–177, 1998.

[11] E. Kushilevitz and R. Ostrovsky. Computationally private information retrieval with polylog communication. In *Proceedings of IEEE Symposium on Foundation of Computer Science*, pages 364–373, 1997.

[12] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC Press, 1997. ISBN 0-8493-8523-7.

[13] R. Merkle. *Secrecy, Authentication, and Public-Key Systems.* PhD thesis, Stanford University, 1979. PH.D Dissertation, Department of Electrical Engineering.

[14] S. Micali. Certificate revocation system. United States Patent 5666416, September 1997.

[15] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. Internet public key infrastructure online certificate status protocol - OCSP. Internet Request for Comments: RFC 2560, 1999. Network Working Group.

[16] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. In *Symposium on Network and Distributed Systems Security (NDSS'04)*, February 2004.

[17] M. Naor and K. Nissim. Certificate revocation and certificate update. *IEEE Journal on Selected Areas in Communications (JSAC)*, 18(4):561–570, April 2000.

[18] John Solis and Gene Tsudik. Simple and flexible revocation checking with privacy. In *Workshop on Privacy-Enhanced Technologies (PET'06)*, July 2006.

[19] Sun Microsystems. Sun Fire T1000 and T2000 Servers Benchmarks. `http://www.sun.com/servers/coolthreads/t1000/benchmarks.jsp`.

[20] International Telecommunication Union. Recommendation x.509 (1997 e): Information technology open systems interconnection - the directory: Authentication framework, 6-1997, 1997. Also published as ISO/IEC International Standard 9594-8.