Certificate Revocation using Fine Grained Certificate Space Partitioning

Vipul Goyal

Department of Computer Science
University of California, Los Angeles
vipul@cs.ucla.edu

Abstract

A new certificate revocation system is presented. The basic idea is to divide the certificate space into several partitions, the number of partitions being dependent on the PKI environment. Each partition contains the status of a set of certificates. A partition may either expire or be renewed at the end of a time slot. This is done efficiently using hash chains.

We evaluate the performance of our scheme following the framework and numbers used in previous papers. We show that for many practical values of the system parameters, our scheme is more efficient than the three well known certificate revocation techniques: CRL, CRS and CRT. Our scheme aims to strike the right balance between CA to directory communication costs and query costs by carefully selecting the number of partitions.

1 Introduction

A certificate is a digitally signed statement binding the key holder's (principal's) name to a public key and various other attributes. The signer (or the issuer) is commonly called a certificate authority (CA). Certificates act as a mean to provide trusted information about the CA's declaration w. r. t. the principal. The declaration may be of the form:

"We, the Certificate Authority, declare that we know Alice. The public key of Alice is ..."

"We further declare that we trust Alice for ..." (optional part)

Certificates are tamper-evident (modifying the data makes the signature invalid) and unforgeable (only the holder of the secret, signing key can produce the signature). Certificates are the building blocks of a Public Key Infrastructure (PKI).

When a certificate is issued, the CA declares the period of time for which the certificate is valid. However, there may be situations when the certificate must abnormally be declared invalid prior to its expiration date. This is called certificate revocation. This can be viewed as "blacklisting" the certificate. This means that the existence of a certificate is a necessary but not sufficient evidence for its validity. A method for revoking certificates and distributing this revocation information to all the involved parties is thus a requirement in

PKI. The reasons for revoking a certificate may be: suspected or detected key compromise, change of principal name, change of relationship between a principal and the CA (e.g., Alice may leave or be fired from the company) or end of CA's trust into the principle due to any possible reason.

The revocation mechanism should have an acceptable degree of timeliness, i.e., the interval between when the CA made a record of revocation and when this information became available to the relying parties should be small enough to be acceptable. Further, it is very important for the revocation mechanism to be efficient as the *running* expenses of a PKI derives mainly from administering revocation [Stu95].

Existing Techniques for Certificate Revocation. Certificate Revocation List (CRL) is the first and the simplest method of certificate revocation. A CRL is a periodically issued and digitally signed list containing the serial number of all the revoked certificates issued by a particular CA. However, it is widely recognized [Mic97, Goy04, Riv98] that CRLs are too costly and cannot provide a good degree of timeliness. Certificate Revocation System (CRS) [Mic96, Mic97, Mic02] was introduced by Micali and could answer the user queries with exceptional efficiency. The main problem with CRS is that it is not suitable in case of a distributed query answering system. The CA to directory communication[1] is too high shooting up the overall cost of the system [NN98, ALO98]. Aiello et al [ALO98] proposed an improvement to CRS aimed at reducing this communication but their approach had problems as we discuss in section 2.2. Certificate Revocation Tree (CRT) [Koc98] is the third well known technique for certificate revocation. Though the CA to directory communication is very low, the query cost is too high, again shooting up the overall cost of the system.

Another technique for certificate revocation is the Online Certificate Status Protocol (OCSP) [OCSrg] designed by IETF. In OCSP, the CA simply digitally signs the response to a certificate status query. Thus, OCSP may provide very high degree of timeliness but is recognized to be non-scalable since the CA is required to compute a signature for answering every query. Further, OCSP has no distributed implementation, i.e., it cannot be used in the settings where there should be a number of un-trusted directories answering the user queries. Using techniques from Identity based encryption [BF01], Gentry [Gen03] proposed a new cryptosystem having attractive properties in terms of revocation. However it was not a generic revocation solution and could not be used with existing cryptosystems (such as RSA). Other revocation techniques include [DBW01, BLL00, GGM00, MJ00]. See [Zhe03] for an analysis of these techniques.

Our Contribution. Motivated by the CA to directory communication cost (CDCC) and the query cost (QC) imbalances in both CRS and CRT, we propose a new system aimed at balancing the two. CRS and CRT are the two extremes, CDCC being very high and QC being extremely low in the former, and CDCC being extremely low and QC being too high in the latter. Our technique aims at striking the right balance between CDCC and QC to minimize the system cost. The basic idea is to divide the total certificate space into

[1]The terms "CA to directory communication cost" and "directory update cost" are used interchangeably in this paper

several partitions. The number of partitions is a key parameter which can be optimized to reduce the overall communication cost. Each partition has a unique serial number, is digitally signed and contains the status of a set of certificate. At the end of a time slot, a partition may either expire or be renewed depending upon whether there was a status change for any of the certificates covered by it or not. Renewing a partition is done by exposing a link of the hash chain whose tip is embedded in that partition. Our system is named CSPR (certificate space partitioning with renewals).

As we show in section 4, the overall communication cost of our system is less than the three widely used revocation techniques, i.e., CRL, CRS and CRT for many practical values of the system parameters.

Rest of the paper is organized as follows: section 2 gives a background on hash chains and common certificate revocation techniques, section 3 introduces the proposed system called CSPR, section 4 evaluates the CSPR costs and compares it with other techniques in use, section 5 concludes the paper.

2 Background

2.1 Hash Chains

A hash chain of length L is constructed by applying a one-way hash function $H(.)$ recursively to an initial seed value s.

$$H^L(s) = \underbrace{H(H(...H(s)))}_{\text{L times}}$$

The last element $H^L(s)$, also called the tip T of the hash chain, has the property that using $H^L(s)$, $H^{L-1}(s)$ can not be computed but its correctness can be verified.

2.2 Certificate Revocation Techniques

Certificate Revocation List (CRL) is the first and the simplest method of certificate revocation. A CRL is simply a periodically issued, time-stamped and digitally signed list containing the serial number of all the revoked certificates issued by a particular CA.

Certificate revocation status (CRS) was introduced by Micali [Mic96, Mic97, Mic02]. It was also patented and commercialized. The basic idea is as follows. For certificate creation, the CA chooses two random numbers Y_0 and N_0 and computes $Y = H^{365}(Y_0)$ and $N = H(N_0)$. These two fields are included in the certificate and are signed along with the other usual fields. The number 365 denotes the number of days in the year. On the ith day,

1. if the certificate is revoked, the CA releases N_0, which can be verified by hashing and comparing with N specified in the certificate.

2. if the certificate is still valid, the CA releases $H^{365-i}(Y_0)$ which can be verified by hashing i times and comparing with Y specified in the certificate.

Aiello et al [ALO98] extended CRS by reducing the overall CA to directory communication while still maintaining the same tiny query communication. This is done by including $\log_2(N)$ hash chain tips in each certificate, N being the number of certificates in the system. Although the reduction in CDCC was high for low revocation rates, the same cannot be said for system with higher revocation rates. Further, this improvement comes at the price of a significant increase in the certificate transmission costs due to increase in the certificate size.

Certificate Revocation Tree (CRT) was introduced by Kochar [Koc98, NN98]. A CRT is based on a Merkle hash tree [Mer89] containing certificate serial number ranges as the tree leaves. The root of the hash tree is signed by the CA. Now, the certificate status proof for a certificate with serial number s consists of the path node siblings from the root to the appropriate leaf (having s in its range), in addition to the signature on the root of the tree.

3 The Proposed Technique

We start by explaining a few notations to be used in the rest of the paper.

N	The total number of certificates handled by the CA
R	The estimated number of certificates (out of N) that will eventually be revoked prior to expiration
T	The number of time slots for which a certificate is issued. One time slot is the duration between two certificate status information releases by the CA (e.g., for a weekly CRL, time slot is one week). It represents the maximum amount of time between when a certificate gets revoked by the CA and when the new status is made available to the relying parties.
q	Estimated average number of queries per day handled by the system
T_i	Representation of the ith 'absolute' time slot, i.e., the ith time slot after the CA has started operation
P	The number of partitions in which the total certificate space of N certificates is divided. P is the key parameter in our scheme. We discover later the technique to find out the optimal value of P for a given set of system parameters.
S_i	The serial number of the ith partition
n	The number of certificates whose status is contained in one partition. We have $n \times P = N$
P_i^j	The version of the ith partition created and released at the beginning of jth time slot T_j. We talk more about versions of a partition later.
D	Number of directories in the system
U	Number of updates to the directories per day. Hence, U is equal to the number of time slots in one day.
$S_{CA}(M)$	Signature on the message M with the private key of the CA.
L_{SR}	The number of bits needed to hold a serial number (of a certificate or partition)
L_H	The length of the hash function output in bits
L_S	The length of a digital signature in bits

L_P	The length of a partition in bits
L_T	The number of bits needed to hold a time slot number

3.1 Creating Partitions

Unlike CRS and like CRT and CRL, our solution works for non custom built certificates. While CRS requires the certificates to have two additional fields Y and N, our technique can be used with any set of certificates having serial numbers. This may be especially important while migrating an existing PKI from one certificate revocation solution to another.

As discussed before, the basic idea is to divide the certificate space into a number of partitions, each partition containing revocation status of the certificates with serial numbers in a particular range. Note that a partition may have several 'versions'. When a certificate contained in a partition changes status (i.e., gets revoked), the current version of that partition is said to have expired and a new version, reflecting the new certificate status, is created and released by the CA at the beginning of the next time slot. If none of the certificates in a particular partition gets revoked during a time slot, the same version is renewed by the CA by exposing a hash chain link at the beginning of the next time slot. The details follow.

The CA divides the whole certificate space into P partitions[2], each partition containing the status information of n certificates having consecutive serial numbers. Each partition is given a unique serial number which is one less than the serial number of the first certificate in that partition. Hence, a partition having serial number S_i contains the revocation status of certificates with serial numbers from $S_i + 1$ to $S_i + n$. Each partition contains a field called "Certificate Status Data" (CSD). This field contains the revocation status of all the certificates in that partition. The jth bit of the CSD of the ith partition is 0 if the certificate with serial number $S_i + j$ is revoked and is 1 otherwise. Clearly, the size of this field is n bits, one bit each for holding the status of one certificate. We represent the CSD field of the ith partition as CSD_i.

For creating a version P_i^j of the ith partition to be released at the beginning of T_j, a hash chain of length L with seed k_i^j is constructed and its tip is specified in the partition. We talk more about the choice of L later. The seed k_i^j is chosen randomly by the CA. We have:

$$P_i^j = S_{CA}\left(T_j, S_i, H^L\left(k_i^j\right), CSD_i\right) \tag{1}$$

Where CSD_i is the certificate status data at the beginning of T_j.

It is also possible to later add more certificates to the already existing set of certificates by adding new partitions. Since certificates may not always be added in chunks of n, we allow the new partition being added to have some non-existent certificates also. More precisely, a new partition will be created with all of its n *CSD* bits as 1. It is possible that there may not yet exist certificates with some of the serial numbers lying in the serial number range of that partition. Consequently, as more and more certificates are later added, there will be no need to add more partitions until there are no non-existent certificates in that partition.

[2]We discuss how to select P optimally later on

3.2 System Operation

At the beginning of the time slot T_k, the CA does the following:

1. For all the partitions for which there was no change in the certificate status data (i.e., none of the certificates in that partition were revoked) during time slot T_{k-1}, the CA reveals the next link of the hash chain. For partition P_i^j, the link $H^{L-(k-j)}(k_i^j)$ is revealed. All the directories in the system are updated with this new hash chain link. Hash chain traversal techniques [Jak02, CJ02, Sel03] may be used by the CA to efficiently compute the next link to be revealed. Note that CSD need not be changed for a certificate which expired (but was not revoked) during the time slot T_{k-1}. Hence, it is perfectly possible that the status of a certificate is 1 even after it has expired.

2. For all the partitions for which the certificate status data changed during time slot T_{k-1}, a new version is released by the CA. The new version P_i^k for P_i^j is created as follows:

$$P_i^k = S_{CA}\left(T_k, S_i, H^L\left(k_i^k\right), CSD_i\right)$$

Where CSD_i is the new certificate status data for this partition.

As an optimization, the CA need not send the whole new partition version to the directories. Instead, only the information which enables the directories to create the new partition version using the older version is sent. We call this information the partition update information (PUI). For a partition, the serial number of all the certificates revoked from it along with the new hash chain tip and the new signature suffice as PUI.

Now, during time slot T_k, a directory answers a user certificate status query for a serial number s by first locating the appropriate partition and then sending that partition and the latest hash chain link revealed for that partition to the user. More precisely, the directory finds an S_i s.t. $(S_i+1) \leq s \leq (S_i+n)$, locates its current (un-expired) version P_i^j, and then sends back P_i^j along with $H^{L-(k-j)}(k_i^j)$. Note that there is no need to trust the directory for sending the un-expired versions only. A directory will be unable to produce the hash chain link $H^{L-(k-j)}(k_i^j)$ for a version P_i^j which expired prior to the beginning of time slot T_k.

The verifier can verify the status of the certificate from the response by:

1. making sure that $(S_i+1) \leq s \leq (S_i+n)$ and the $(s-S_i)$ th bit of the CSD contained in the sent P_i^j is 1,

2. verifying the CA signature on P_i^j,

3. verifying that hashing the sent hash chain link (*current time slot – time slot number* T_j *specified in* P_i^j) times matches with the hash chain tip specified in P_i^j.

Now we comment on the choice of L, the length of the hash chain. If a partition does not expire due to changes in its CSD, it will automatically expire when its hash chain links are exhausted. At this point, a new version will have to be created by the CA. Thus, simply, L should be large enough to ensure that the probability of the hash chain links

being exhausted before the partition expires is reasonably low. A good choice for L seems to be T, the total number of time slots for which a certificate is valid. Assuming $L = T$, by the time the hash chain gets exhausted, all the certificates in the partition would have already expired.

3.3 System Costs

Now we determine the average daily cost of the proposed system in terms of bits. We have the following.

CA to Directory Communication Cost per Update (CDCCPU). CA to directory communication per update is comprised of the hash chain links for the unexpired partitions and the PUI for the expired partitions. The aggregate of all PUIs consists of the serial numbers of all the certificates revoked during the previous time slot plus the new hash chain tip and signature for all the expired partitions. Clearly, the average number of certificates revoked per time slot is R/T. Assuming E to be the average number of partitions expiring per time slot, we have the following:

$$CDCCPU = (P - E)\, L_H + \frac{R}{T}.L_{SR} + E\,(L_H + L_S)$$

Or,

$$CDCCPU = P.L_H + \frac{R}{T}.L_{SR} + EL_S \tag{2}$$

E is clearly upper bounded by the number of certificates being revoked per time slot, i.e., R/T. However, since multiple revoked certificates may be from the same partition, E may actually be lesser than this value. E is equal to the number of bins having at least one ball when R/T balls are thrown into P bins. Hence, E may be computed as follows:

Probability of the ith bin not having the ball when a ball is thrown into P bins $= (1-\frac{1}{P})$

Probability of the ith bin not having a ball when R/T balls are thrown into P bins $= (1-\frac{1}{P})^{\frac{R}{T}}$

Expected number of bins having at least one ball = Total number of bins – Expected number of bins having no balls

Or,

$$E = P - P\left(1 - \frac{1}{P}\right)^{R/T} \tag{3}$$

Directory Query Cost per Day (QC). A query response consists of the appropriate partition plus a hash chain link indicating the proof of renewals. Hence, the daily query cost QC is

$$QC = q.\,(L_P + L_H)$$

Total System Cost per Day (TC). Total daily cost TC of the system consists of updating each of the D directories U times plus the query costs. Hence, we have:

$$TC = U.D.\left(P.L_H + \frac{R}{T}.L_{SR} + EL_S\right) + q.\left(L_P + L_H\right)$$

Further, from (1), we have:

$$L_P = L_T + L_{SR} + L_H + \frac{N}{P} + L_S$$

Thus,

$$TC = U.D.\left(P.L_H + \frac{R}{T}.L_{SR} + EL_S\right) + q.\left(L_T + L_{SR} + 2.L_H + \frac{N}{P} + L_S\right) \quad (4)$$

3.4 Optimal Number of Partitions

In this section, we determine the optimal value of P to minimize the total daily system cost. As will be clear in the next section, the total number of partitions P would usually be much higher than the number of certificates expiring per time slot. Hence, to simplify our analysis, we approximate E by R/T. We put this approximation of E in equation (4) and compute the minima of its R.H.S. using differentiation. Differentiating R.H.S. of (4) w.r.t. P and putting the result equal to zero, we get:

$$U.D.L_H + q\left(-\frac{N}{P^2}\right) = 0$$

or,

$$P = \sqrt{\frac{N.q}{U.D.L_H}} \quad (5)$$

The above equation also gives useful insights on the issue of PKI expansion. As more and more certificates are added to the PKI to increase N, the number of queries per day, i.e. q, is also expected to increase by the same factor. This is because the number of daily verifications of a certificate (and hence the queries pertaining to it) is independent of N. q is also expected to increase linearly with N in the Rivest's model [Riv98] of "Certificate holder supplies all validity evidences to the verifier" in which, instead of the verifier, the holder sends periodic queries to obtain recent validity evidence for its certificate. Thus from (5), along with q, P also increases linearly with N. This means that the optimal size of (or the number of certificates in) a partition, i.e., n $(=N/P)$ remains unaltered as the PKI expands. Hence, no extra efforts are needed to maintain optimality as the size of the PKI changes.

The above argument may not hold in some specialized PKIs. It is still possible to maintain optimality in such systems by periodically re-computing the optimal value of n and setting it as the partition size for new partitions being created. As certificates expire, older partitions will continue to get removed from the system[3]. Hence, the system forever keeps 'migrating' to the (currently) optimal value of the partition size.

[3]A partition will be removed from the system when all the certificates in it get expired.

4 Evaluation and Comparison

Following the framework of CRS [Mic96] and CRT [NN98], we evaluate the cost of the proposed system called CSPR from hereon. The following values of the parameters are assumed.

$N = 3 \times 10^6$, $R = 3 \times 10^5$, $q = 3 \times 10^6$, $T = 365 \times U$, $L_H = 160$, $L_S = 1000$, $L_{SR} = 30$, $L_T = 20$

The above values are mostly taken from [NN98]. The evaluations are done for the values of D between 0 and 10,000 and U between 1 and 100. Observe the value of T, the number of time slots. Since we assume that a certificate is issued for one year; T is equal to the number of days in one year (365) multiplied by the number of time slots in one day (U).

The comparison is done with CRL, CRS and CRT. Before going further, we compute the daily communication cost of each of these techniques.

Certificate Revocation Lists Daily Cost. Average CRL Size $= \frac{R}{2}.L_{SR} + L_S$

$$\text{Total Daily Cost} = D.U\left(\frac{R}{2}.L_{SR} + L_S\right) + q\left(\frac{R}{2}.L_{SR} + L_S\right) \tag{6}$$

Certificate Revocation System Daily Cost.

$$\text{Total Daily Cost} = D.U\left(N.L_H\right) + q.L_H \tag{7}$$

Certificate Revocation Tree Daily Cost. CA to Directory Communication per update consists of the serial numbers of the certificates revoked during the previous time slot and the new signature on the root of the Merkle tree. Query response consists of tree path node siblings from the appropriate leaf to the root along with the root signature. Hence, we have:

$$\text{Total Daily Cost} = D.U\left(\frac{R}{T}.L_{SR} + L_S\right) + q\left(L_H \log \frac{R}{2} + L_S\right) \tag{8}$$

Table 1 summarizes the total daily system costs in bits for CRL, CRS and CRT using (6), (7) and (8) respectively and for CSPR using (4) with optimal number of partitions found using (5). The values are computed for various choices of D and U.

The improvement of CRS due to [ALO98] appears in parenthesis. It should be stated here that this improvement comes at the cost of increasing the certificate size by 3.5 KB. Assuming one certificate transmission per revocation status query, the increase in certificate transmission costs comes out to be 1.1×10^{10} bits per day (not added in the table) .

Table 2 lists the optimal values of P and the corresponding n found using (5).

	CRL	CRS	CRT	CSPR
$D = 0$	1.3×10^{13}	4.8×10^{8} (4.8×10^{8})	1.1×10^{10}	4.1×10^{9}
$D = 10$, $U = 1$	1.3×10^{13}	5.3×10^{9} (2.3×10^{9})	1.1×10^{10}	4.3×10^{9}
$D = 100$, $U = 1$	1.3×10^{13}	4.8×10^{10} (1.8×10^{10})	1.1×10^{10}	4.9×10^{9}
$D = 100$, $U = 10$	1.3×10^{13}	4.8×10^{11} (1.8×10^{11})	1.1×10^{10}	6.6×10^{9}
$D = 100$, $U = 100$	1.4×10^{13}	4.8×10^{12} (1.8×10^{12})	1.1×10^{10}	1.2×10^{10}
$D = 1000$, $U = 100$	1.4×10^{13}	4.8×10^{13} (1.8×10^{13})	1.1×10^{10}	2.9×10^{10}
$D = 10,000$, $U = 100$	1.8×10^{13}	4.8×10^{14} (1.8×10^{14})	1.3×10^{10}	8.8×10^{10}

Table 1: Daily System Costs in Bits for Common Revocation Techniques

	Number of Partitions P	Certificates per partition n
$D = 0$	3.0×10^{6}	1×10^{0}
$D = 10$, $U = 1$	7.5×10^{4}	4.0×10^{1}
$D = 100$, $U = 1$	2.4×10^{4}	1.2×10^{2}
$D = 100$, $U = 10$	7.5×10^{3}	4.0×10^{2}
$D = 100$, $U = 100$	2.4×10^{3}	1.2×10^{3}
$D = 1000$, $U = 100$	7.5×10^{2}	4.0×10^{3}
$D = 10000$, $U = 100$	2.4×10^{2}	1.2×10^{4}

Table 2: Optimal Values of P and the corresponding n for various values of D and U

Remark 1. As demonstrated by the above values, CRS and CRT are the two extremes, CRS having unbeatable query costs but having CA to Directory Communication as bottleneck and CRT having unbeatable CA to Directory Communication but having query costs as the bottleneck. The proposed technique is able to balance the two costs by optimally choosing the number of partitions P and thus minimizing the overall cost of the system. Hence, P is the key parameter for CSPR.

Remark 2. The formula for the total cost of the system may be modified by assigning suitable weight to CDCC and QC. For example, in some low budget systems, CDCC may be assigned more weight to prevent CA from becoming the communication bottleneck, while in others, QC may be assigned more weight to improve the user experience. Accordingly, the formula for P is modified (by taking the weight during differentiation) and P is still able to play the role of cost balancer between CDCC and QC.

Remark 3. The computation required to validate a certificate status proof is similar in CRL, CRT and CSPR (dominated by a signature verification). For lower update rates, CRS has an advantage as it does not require a signature verification. However, as update rate increases, the computation starts becoming comparable to others. This is because the average number of hash function evaluations required for validating one certificate status proof in CRS is $365 \times U/2$.

Distribution Degree and System Timeliness	CRS	CRT	CSPR
Centralized system or a system having very low distribution degree	*Suitable*	*QC High*	*QC High*
Moderately low to moderately high timeliness or distribution degree	*CDCC High*	*QC High*	*Suitable*
Very high timeliness or distribution degree	*CDCC High*	*Suitable*	*TC High*

Table 3: Revocation Technique Selection

Future Work. An interesting observation in the proposed system is that the partitions (without hash chains) may actually be treated as ordinary certificates and our scheme may be recursively applied. The notion of partition expiry may be replaced with the notion of partition revocation. Level 2 partitions may be created which will contain the status of a number of level 1 (ordinary) partitions having consecutive serial numbers. A Level 2 partition will have hash chains as the renewal mechanism and will expire as soon as any of the level 1 partitions in it expires (gets revoked). Similarly, level 3 partitions and so on are possible, the extreme being a tree of partitions of different levels. The above approach may be worth exploring in environments where the number of directories or updates per day is high. This is because it may reduce the CA to directory communication costs which are quite high in such environments, though at the price of increasing the query costs.

5 Conclusions

We conclude by summarizing the suitability of various revocation schemes under different values of D and U in Table 3. D is an indicator of the distribution degree of the system while U is an indicator of system timeliness.

References

[ALO98] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast digital identity revocation (extended abstract). In *CRYPTO*, pages 137–152, 1998.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BLL00] Ahto Buldas, Peeter Laud, and Helger Lipmaa. Accountable certificate management using undeniable attestations. In *ACM Conference on Computer and Communications Security*, pages 9–17, 2000.

[CJ02] Don Coppersmith and Markus Jakobsson. Almost optimal hash sequence traversal. In *Financial Cryptography*, pages 102–119, 2002.

[DBW01] Gene Tsudik Dan Boneh, Xuhua Ding and Chi Ming Wong. A method for fast revocation of public key certificates and security capabilities. In *The 10th USENIX Security Symposium*, pages 297–308, 2001.

[Gen03] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293. Springer, 2003.

[GGM00] Irene Gassko, Peter Gemmell, and Philip D. MacKenzie. Efficient and fresh cerification. In *Public Key Cryptography*, pages 342–353, 2000.

[Goy04] Vipul Goyal. Certificate revocation lists or online mechanisms. In Eduardo Fernández-Medina, Julio César Hernández Castro, and L. Javier García-Villalba, editors, *WOSIS*, pages 261–268. INSTICC Press, 2004.

[Jak02] M. Jakobsson. Fractal hash sequence representation and traversal, 2002. ISIT '02; available at http://eprint.iacr.org/2002/001 and www.markus-jakobsson.com.

[Koc98] Paul C. Kocher. On certificate revocation and validation. In *Financial Cryptography*, pages 172–177, 1998.

[Mer89] Ralph C. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.

[Mic96] Silvio Micali. Efficient certificate revocation. Technical Report MIT/LCS/TM-542b, 1996.

[Mic97] Silvio Micali. Efficient certificate revocation. In *Proceedings 1997 RSA Data Security Conference*, 1997.

[Mic02] Silvio Micali. Novomodo: Scalable certificate validation and simplified pki management. In *1st Annual PKI Research Workshop - Proceeding*, 2002.

[MJ00] Patrick Drew McDaniel and Sugih Jamin. Windowed certificate revocation. In *INFOCOM*, pages 1406–1414, 2000.

[NN98] Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. In *Proceedings 7th USENIX Security Symposium (San Antonio, Texas)*, Jan 1998.

[OCSrg] Online certificate status protocol, version 2. In *Working document of the Internet Engineering Task Force (IETF), RFC 2560*, Available from http://www.ietf.org.

[Riv98] Ronald L. Rivest. Can we eliminate certificate revocations lists? In *Financial Cryptography*, pages 178–183, 1998.

[Sel03] Yaron Sella. On the computation-storage trade-offs of hash chain traversal. In *Financial Cryptography*, pages 270–285, 2003.

[Stu95] Stuart Stubblebine. Recent-secure authentication: Enforcing revocation in distributed systems. In *Proceedings 1995 IEEE Symposium on Research in Security and Privacy*, pages 224–234, May 1995.

[Zhe03] Peifang Zheng. Tradeoffs in certificate revocation schemes. *Computer Communication Review*, 33(2):103–112, 2003.