Efficient Certificate Revocation

by

Silvio Micali

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MASS 02139

March 22, 1996

ABSTRACT

We apply off-line/on-line signatures to provide an alternative solution to the problem of certificate revocation.

The new systems dismiss with traditional CRLs (Certificate Revocation Lists) and yield public-key infrastructures that are substantially cheaper to run than traditional ones.

1 Introduction

Basic cryptographic primitives such as encryption, digital signatures, and hash functions have been the subject of a great deal of attention, and have reached a remarkable level of sophistication, speed, and security.

In comparison, however, research in cryptographic infrastructure has been lagging behind. This is a pity. Though it may lack the mathematical appeal of encryption or digital signatures, cryptographic infrastructure is crucial to the successful massive deployment of cryptography.

One key aspect of this infrastructure is that of certificate revocation. In many settings, it is necessary to certify certain data, as well as to revoke already issued certificates. For instance, in a Public-Key Infrastructure (PKI) it is necessary to certify users' public keys, as well as to revoke already issued certificates. For concreteness, and because it is an example of great importance, we shall focus on the case of public verification keys of a digital signature system.

Certificate revocation of a given public verification key becomes necessary when the corresponding secret key has been compromised, or when the job or qualification of a user U change. In fact, a given public key is not associated just to U alone, but to "U, vice president of Bank B," to "U, member of the committee C," to "U, private citizen," and so on. Thus, when U's situation changes, his or her public key certificates must be revoked. Recent studies [?] actually estimate that 10% of the certificates will need to be revoked on a yearly basis when public-key cryptography is used on a grand scale.

To verify the digital signature of a user U, one needs not only U's public key PK_U (for simplicity, we shall from now on assume that there is a single "incarnation" of U), but a certificate that PK_U really belongs to U. But even this is not good enough, because the verifier of a signature of U should make sure that *PK_U's certificate has not been revoked.* If one verifies the signature of the same user U quite often (e.g., many times in the same day), then continually inquiring about the possible revocation of PK_U's certificate may be unnecessary, but determining whether a certificate has been revoked is mandatory in essentially all other cases.

It is thus apparent that finding *effective* mechanisms for handling certificate revocation is practically very important for using digital signatures on a massive scale.

Unfortunately, as we shall recall below, traditional solutions to the certificate revocation problem are based on a construct known as a Certificate Revocation List (CRL), which makes them quite slow and expensive. It is thus our goal to provide simple and efficient certificate revocation systems that dismiss with CRLs.

Before presenting our proposed systems, let us first review the players of a PKI, and then the traditional way to achieve certificate revocation. For concreteness sake, we shall use the estimates relative to the Federal PKI, the envisaged public-key infrastructure for the Federal Government of the United States [**?**].

THE MAIN PLAYERS

Though the typical PKI has several players and subplayers, for the purpose of this abstract we distinguish three main types of players:

CAs: A certification authority (CA) is a trusted agent that issues and revokes certificates.

CAs are not in the business (nor do they have the equipment or technical capacity) of providing on-line certificate-information services on a grand scale. They inform the Directory of the decisions they make.

Directory: The Directory is a non-trusted (or much less trusted) agent that receives certificate information from the CAs and handles user queries about it.

Users: Users are non-trusted agents who receive certificates from the CAs, produce and verify digital signatures, and query the Directory for certificate information.

The Federal PKI envisages having three million users and 100 CAs, each serving 30,000 users, and estimates that a CA revokes 10% of the certificates she issues. (Whether these estimates are precise or realistic is not the point here, because the new certificate revocation system will improve on the traditional one anyway.)

CRL-BASED CERTIFICATE REVOCATION

Let us now recall (an abridged version of) a typical PKI.

A CA has an already certified (or universally known) public key. To certify that PK_U is U's public key, a CA typically digitally signs PK_U together with her own name and signing algorithm, U's name and qualification, a certificate serial number, the current date (i.e., the certification date), and the expiration date.[1] The CA's signature of PK_U is then sent to the Directory and/or given to U himself.

When sending a recipient R his own signature of a message M, $SIG_U(M)$, user U typically also sends the certificate for his own public key PK_U (which is otherwise obtainable from the Directory, or is already in storage with R). The recipient R typically checks the correctness of the certificate for PK_U, and then that of $SIG_U(M)$ against PK_U. However, it is estimated that 20% of the time R will also need to check whether U's certificate has been revoked.

[1]Before so certifying U's public key, it is necessary to perform additional steps, such as properly identifying user U.

To enable a recipient to establish whether a given certificate has been revoked, each CA periodically issues and gives the Directory a CRL containing an indication of all the revoked (not yet expired) certificates originally issued by her. A CRL typically consists of the issuer's digital signature of (1) a CRL *header* comprising the issuer name (as well as the type of her signature algorithm), the current date, the date of the last update, and the date of the next update, together with (2) a complete list of the revoked (and not yet expired) certificates, each with its serial number and revocation date. Since it is expected that a CA revokes many of her certificates, a CRL is expected to be quite long.

After performing some checks on the CA's CRL (e.g., checking the CA's digital signature, checking that the CRL has arrived at the expected time, that a certificate declared revoked in the previous CRL of that CA —and not yet expired— still is revoked in the current CRL, etc.), the Directory stores it under its CA's name.

When a user queries it about the revocation of a certificate issued by a given CA, the Directory responds by sending the user the latest CRL of that CA. The user can then check the CRL signature, the CRL dates (so as to receive a reasonable assurance that he is dealing with the latest one), and whether or not the certificate of interest to him belongs to it.

Note that in a CRL-based system the requesting user need not trust the Directory. In fact, the CA's signature of a CRL proves to the user that he is getting *precisely all the revoked certificates of the given CA at the given date*. The user would instead need to trust the Directory if it simply told him whether or not a given certificate was revoked, or if the CA provided the Directory (and the Directory the user) with an independent piece of signature for each revoked certificate (because the Directory could withhold this individual proof of revocation).

While CRLs are quite effective in helping users establishing which certificates are no longer deemed valid without trusting the Directory, they are also extremely expensive, because they tend to be very long and need to be transmitted very often.

CRL COST

CRLs constitute by far the largest entry in a traditional PKI's cost list. In the Federal PKI, according to NIST's estimates/assumptions, CRLs are sent-out bi-weekly, and each revoked certificate is specified by means of about 9 bytes: 20 bits of serial number and 48 bits of revocation date. Thus, in the Federal PKI, each CRL is expected to comprise thousands of certificate serial numbers and their revocation dates; its header, however, has a fixed length, consisting of just 51 bytes.

At 2 cents per kilobyte, the impact of CRL transmission on the estimated yearly costs of running the Federal PKI is quite stunning. If each user is assumed to verify just 5 digital signatures a day on average, then the total PKI yearly costs are $732 Millions, of which 563 Millions are due to CRL transmission. (Of course costs are much higher if more signatures are verified per day. For instance, if each user verifies 100 digital signatures per day on average, then the total PKI yearly costs are $10,848 Millions, of which 10,237 Millions are due to CRL transmission.)

While "2-cents/kilobyte" appears to be a quite high estimate, it is still clear that CRLs would be very expensive and time-consuming anyway. Let us thus investigate a simple and convenient alternative cryptographic design that substantially decreses these costs.

As a side remark, updating CRLs bi-weekly appears to be quite dangerous. Certificate-revocation information should be updated at least daily.

2 An Alternative Certificate Revocation System

One possible way to avoid CRLs high costs is having each CA daily send the Directory a digital signature for each certificate, indicating whether, at the current date, it has been revoked or not.

Such a simple strategy, however, also has its own disadvantages. Recall that in the Federal PKI it is envisaged that each CA serves 30,000 users. Thus, even assuming that each user has a single public key (while 5 may be more realistic), a CA would have to sign 30,000 new messages every day. This may prove to be barely feasible for many signature schemes; particularly, in view of the fact that certification keys should be quite long. Moreover, it may become even harder if (as will be the case) users have multiple public keys. In addition, while the length of a single secure signature is much shorter than the typical CRL, it may consist of some 1,000 bits. Thus, we aim at improving also on such a "direct signing" approach, both in terms of computation and in terms of bit transmission.

We thus propose to substitute CRLs with an alternative and simple information structure, which we call a CRS (for Certificate Revocation Status). Unlike CRLs, each CRS is a short and individualized piece of information for a given certificate. Computing a CRS is much faster than signing a message, and its length is much shorter than that of a digital signature. As a result, the new system is quite preferable to both CRLs and "direct signing."

Even an intuitive analysis (a precise one is presented in Section 2.2) of a CRL-based system shows that the total CAs-to-Directory communication due to CRL updating is small (and can occur *off-line*), while the total Directory-to-Users communication due to certificate-revocation queries is huge. Thus we have developed the new system according to the following natural strategy:

> Increase the amount of information sent by CAs to the Directory during an update, but design this larger information so as to enable the Directory to answer more succintly certificate-revocation queries.

As we shall see, this tilting of the scale results in substantial savings.

2.1 CRS Usage

CA OPERATIONS

- *(Making a Certificate.)* A CA produces the certificate of a user's public key by digitally signing together (1) traditional quantities (e.g., the user's public key, the user's name, the certificate's serial number, the type of signature algorithm of the issuer, the certification date, and the expiration date) and (2) two new quantities: a 100-bit value Y (for "YES") and a 100-bit value N (for "NO"). These values are, at least with very high probability, unique to the certificate.

  The CA generates Y by selecting a secret 100-bit value, Y_0, and then evaluating on it a given one-way function F 365 times (i.e., as many as the days in a year).[2] Thus, $Y = Y_{365} =$

[2] Rather than straightforwardly and repeatedly applying F, it is preferable that a CA C use a one-way hash function H, "individualizing" each application of H by specifying the name of the CA, the certificate in question, the iteration-number, the date, and other quantities. For instance, she may choose $Y_1 = H(Y_0, C, 1, date, serial\ number)$, $Y_2 = H(Y_1, C, 2, date, serial\ number)$, and so on. This helps prevent the occurrence of "accidental collisions" between the Y_i values of different certificates. It also prevents the possibility that an enemy may, by evaluating F at random points, "hit" some Y_i value of some certificate.

$F^{365}(Y_0)$. The CA generates N by selecting a secret value N_0 and then evaluating F on it once; that is, $N = F(N_0)$.

The CA may select Y_0 and N_0 at random (in which case she must separately store them) or pseudo-randomly (e.g., she computes them by means of a secure pseudo-random function [?] from a secret master key —which she keeps in storage— and other inputs such as the certificate serial number, and the issue date). In the latter case, the CA can recompute Y_0 and N_0 when needed, rather than storing them at all times.

- (Updating the CRS.) Daily, a CA sends the Directory the following information:

  (*a*) An authenticated and updated "list" of all serial numbers corresponding to issued and not-yet-expired certificates.

  For simplicity, let this information consist of a dated and signed commitment of the CA to a 2^{20}-bit string S, whose nth bit of S is 1 if serial number n corresponds to an issued and not-yet-expired certificate, and 0 otherwise. (Note that S comprises as many bits as there are serial numbers for a given CA).

  (*b*) For each not-yet-expired certificate made by her, she sends a 100-bit value computed as follows. Assume that the current day is the ith day in some given system of reference (i.e., the ith day of the year, or the ith day after the start date of the certificate, and so on). Then, if the certificate is still valid, the CA sends the value Y_{365-i} $(= F^{365-i}(Y_0)$, which she may easily compute by evaluating F $365 - i$ times on input Y_0). If the certificate has been revoked that very day, she sends the value N_0.

  (If so desired, for each revoked certificate, the CA also sends the Directory a richer piece of information: her direct digital signature that the certificate has been revoked, including additional information, such as the revocation date, reasons for revocation, etc. We call such a piece of data a *full* revocation certificate.)

In addition, the CA may take advantage of a CRS update to send

  (*c*) The new certificates made that day.

Note 1: The above system constitutes another application of what may be called "light-weight signatures," a simple and effective type of off-line digital signing [?] that has proved quite useful; for instance, in the (independently developed) micro-payment system of [?].

The value $Y = Y_{365}$ is, in fact, the public-key of a second, more limited, digital signature scheme, whose secret key is Y_0. This second scheme is capable of signing a limited number of messages (namely, the integers between 1 and 365), but it is very fast, since there are one-way functions F that are extremely easy to evaluate. (Such simpler signature schemes were originally developed by Lamport, Winternitz, and Merkle. See [?] for a comprehensive exposition.)

In an off-line step, the CA uses a first (traditional) signature scheme to sign the public key Y within the certificate, and then, in an on-line step, she uses the second signature scheme to sign a value in the interval [1,365] in a very quick fashion.

The further signature of integer i, Y_{365-i}, indicates that a certificate is valid up to date i. Of course, if a certificate is valid up to date i, it is also valid up to any date between 0 and i. Indeed, if $j < i$, the signature Y_{365-j} was released before Y_{365-i}. Illegally extending the validity of a certificate is very hard and requires signing a message never signed before by the legitimate signer.

Note 2: The above scheme can be modified so that, after a certificate becomes invalid, the CA starts to send the Directory "F-inverses" of N, so that, at day i, the total number of released F-inverses of Y and N is precisely i. This, of course, requires that N be chosen by evaluating F on N_0 365 times rather than just once.

DIRECTORY OPERATIONS

- *(Response to CRS Update.)* For every CA, the Directory stores all not-yet-expired certificates issued by her, organized by serial number, and for each of them it also stores its latest YES-value, if the certificate is still valid, or the 100-bit value $F^{-1}(N)$ otherwise.

  For every CA, the Directory performs some obvious checks on the received type-(a), type-(b), and type-(c) information. (In particular, it checks that the CA's signature and date for each new certificate is correct; that the ith bit of string S equals 1 for every new certificate whose serial number is i; and that it receives a 100-bit value for each not-yet-expired issued certificate.)

  In addition, for every certificate, letting V be the corresponding 100-bit value received from the right CA, the Directory checks that either $F(V) = Y_{i-1}$ (i.e., that $F(V)$ equals the 100-bit value received the day before about that certificate, which it has in storage) or $F(V) = N$.

  (If the CA sends the Directory *full* revocation certificates, then the Directory also checks their correctness.)

- *(Response to Users' Inquiries.)* Assume, for simplicity, that recipients obtain the certificates of the public keys of the senders directly from the senders. Thus, users query the Directory just for determining the revocation status of a certificate already known to them.

  When a user U inquires about the status of a given certificate (e.g., by specifying its CA and its serial number), the Directory retrieves and sends to U the latest 100-bit value relative to that certificate.

  Should U inquire about a serial number that does not correspond to any not-yet-expired certificate issued by the CA, then the Directory, using the type-(a) information received by the CA, sends U a proof this is the case.

  (If full revocation certificates are used, then the Directory may just send U this piece of information, when appropriate, in response to U's query, rather than the right N_0. Alternatively, the Directory may send such a full revocation certificate in response to an additional specific request of U.)

Note 3: The Directory is not much trusted; in particular, it is no more trusted than before. In fact, it cannot "make valid" a revoked certificate. Indeed, if the current date is i, and the certificate has been revoked at date $j < i$, the Directory has only received from the CA the 100-bit values $Y_{365-(j-1)}, \ldots, Y_{365-1}$. Thus, to make the certificate appear valid, it should be able to compute Y_{365-i} $(= F^{-(i-(j-1))}(Y_{365-(j-1)}))$, and thus invert F at least once on input $Y_{365-(j-1)}$, which it cannot do, because F is a one-way function and because (unlike the CA) it does not know Y_0.

Similarly, the Directory cannot "revoke" a valid certificate. Indeed, in order to convince U that the certificate has been revoked, it should be able to compute $F^{-1}(N)$, which again it cannot do.

Nor is the Directory trusted about saying that a given serial number "does not correspond to any certificate." Indeed, it always provides U with a CA-prepared proof of this fact.

USER OPERATIONS

If U has inquired about a certificate of CA with a given serial number and the Directory sends him a proof that no certificate with that serial number exists, U checks this proof.

Else, let i be the current date; let Y and N be, respectively, the YES and the NO value specified within the certificate of interest; and let V be the 100-bit value U receives from the Directory.

Then, U checks whether $F^i(V) = Y$ (in which case he concludes that the certificate is valid); if this is not the case, he checks whether $F(V) = N$ (in which case he concludes that the certificate has been revoked).

If none of these two cases applies, and the Directory does not provide him with a proof that the given serial number does not correspond to any non-yet-expired certificate, then U concludes that the Directory is purposely denying him service.

2.2 CRS Advantages

"COMPLETENESS"

Note that a side advantage of the new system is that it always provides a complete and satisfactory answer to any possible query of a user to the Directory, and without trusting the latter in any special way.

By contrast, in a CRL-based system, if a user queries —by error, malice, or other reason— the Directory about a serial number n that does not belong to any not-yet-expired certificate issued by a given CA, the Directory cannot prove this to the user. Indeed, showing that the latest CRL of that CA does not contain n is not such a proof. (It may actually be construed as proving that certificate number n is valid.) Even giving the user all not-yet-expired certificates issued by that CA is not such a proof: the user may suspect that the Directory is purposely withholding the "right" certificate.

COMMUNICATION COSTS

Let us assess the communication costs directly attributable to certificate revocation in a CRL-based PKI and in a CRS-based one. (Thus, for instance, we disregard the costs associated in the distribution of new certificates to users and the Directory.) We express these costs in *bits* (rather than "dollars"), and as function of abstract quantities. (In fact, estimating these quantities is always quite difficult. Nonetheless, to receive some concrete indication of the performance of the new system, we shall see what happens after substituting in the estimates of the Federal PKI.)

Let n denote the total number of certificates, p the revocation rate (i.e., the percentage of certificates that will be revoked before their expiration), k the average number of certificates handled by a CA, and q the (total) expected number of daily certificate-status queries. Then we have the following expected costs.

- TOTAL DAILY CRL-UPDATE=$20pn$.

  (In fact, for each revoked certificate, the corresponding CA sends the Directory 20 bits of serial number, and the expected number of revoked certificates is pn.

  Recall that n may be much greater than a million, but a serial number should only identify a certificate among those issued by a given CA, and thus only among k certificates —and k can be assumed to be less than 2^{20} in any reasonable PKI, since a CA should also physically identify its users before issuing their certificates.)

- TOTAL DAILY CRL-QUERIES=$68pkq$

  (In fact, each of the expected q certificate-revocation queries is answered with a CRL, and the expected length of a CRL —ignoring the bits of its CA signature— is $68pk$. This is so because a CRL comprises 68 bits —20 of serial number and 48 of revocation date— for each certificate revoked by a given CA, and a CA is expected to revoke pk certificates.)

- TOTAL DAILY CRS-UPDATE=$120n$.

  (In fact, for each of the n not-yet-expired certificates, the corresponding CA will send the Directory 20 bits of serial number and 100 bits of certificate revocation status.)

- TOTAL DAILY CRS-QUERIES=$100q$.

  (In fact, each user query will cause the sending of 100 bits of certificate-revocation status.)

NOTE: All above costs are computed based only on the *number* of bits transmitted, no matter what the monetary cost of their transmission may be. Thus, they do *not* depend on the questionable "2-cent/kilobyte assumption."

It is thus immediately seen that the daily cost of CRS updating is $6p^{-1}$ times more expensive than a CRL one. However, even ignoring the cost of the CA signatures, the daily cost of CRS querying is $pk68/100$ cheaper than a CRL one.

Let us now see what happens with the Federal PKI estimates. According to these estimates, $p = 0.1$, $k = 30,000$, and $q = 3,000,000$ (assuming that each of the 3 Million users verifies 5 signatures per day and that for 1/5 of those he queries the Directory). Therefore, replacing the old certificate revocation system with the new one, the (smaller and *off-line*) update-cost increases by a factor of 60, but the (bigger) query-cost decreases by a factor of 2,040.

Overall, the old system sends a total of about 6×10^{11} bits daily between updates and queries. The new system sends a total of about 6.6×10^{8} bits daily. Thus,

the new system is about 900 times cheaper than the old one.

This figure holds even if one takes into account the fact that the new system makes each certificate 200-bit longer, and that these additional bits are transfered once a year to the Directory.

It is perhaps surprising that such a simple construct as a CRS yields so much saving. It is actually important to establish whether different types of CRS constructs, even more complex ones, may yield better savings. We hope that this optimization problem will attract attention.

THE CASE OF FULL REVOCATION CERTIFICATES

It should be noted that receiving a CRL corresponding to a given revoked certificate also proves the revocation date of that certificate. By contrast, receiving just the right N_0-value only proves that the certificate is no longer valid. Though this is the most important piece of information for a verifier, a CRL is more informative.

If more information is actually desired, then we may make use of a yet more informative construct than a CRL, that is, a full revocation certificate (i.e., a CA's digital signature of serial number, the revocation date, an encoded form of the reasons for revocation, etc.). If a full certificate just specified serial number and revocation date, then its length may be estimated to be some 1,000 bits (less if, say, the DSA is used in signing, more if RSA is used). Thus, when a user queries about a certificate that happens to be revoked, the Directory replies with some 1,000 bits, rather than 100 bits.

However, we expect that such 1,000-bit strings are transmitted to the users for a small fraction of the certificate-revocation queries. In fact, though the envisaged revocation rate may be 10%, we expect that significantly less than 10% of the produced signatures are relative to revoked public keys! Nonetheless, even using a 10% estimate for the latter event (and continuing not to count the cost of sending the CA signature within a CRL, while counting it in a full revocation certificate), the saving factor of the new scheme would decrease to 450.

Notice too that each of the above saving factors is achieved without putting too heavy a burden on a single CA. In fact, at each update, each of the 100 CAs must transmit, on the average, 3.6 Million bits (450K Bytes), a considerable but not unmanageable file —particularly if one considers that this file needs not be sent "on-line," but any convenient time during the day.

Of course, the Federal estimates could be wrong, but the savings of the new system should remain substantial, for PKIs of similar size, for a quite wide range of estimates.

IN SUM

In sum, the new certificate revocation system is quite practical, secure, and cost-efficient. It provides a preferable alternative to traditional approaches, and it will hopefully inspire additional research in this area.

Acknowledgements

Many thanks to Oded Goldreich, Bob Rosenthal, and ray Sidney for their kind and insightful comments.

References

[1] S. Even, O. Goldreich, and S. Micali. *On-Line/Off-line Digital Signing.* Proc. Crypto 89, Santa Barbara, CA, August 1989.

U.S. Patent No. 5,016,274.

[2] O. Goldreich, S. Goldwasser, and S. Micali. *How To Construct Random Functions.* J. of the ACM, Vol. 33, No. 4, October 1986, pp. 792-807.

[3] R. Merkle. *A Certified Digital Signature.* Proc. CRYPTO 89 (G. Brassard, editor), Springer Verlag, 1990, pp. 218-238.

[4] S. Micali. *Enhanced Certificate Revocation System.* Technical memo MIT/LCS/TM-542, November 1995.

[5] National Institute of Standards and Technology. *Public-key Infrastructure Study.* Gaithersburgh, MD, April 1994.

[6] R. Rivest and A. Shamir. *Pay Word and MicroMint: Two simple micropayment schemes.* Unpublished manuscript, November 7, 1995.