

## NOVOMODO

Scalable Certificate Validation And Simplified PKI Management

by

Silvio Micali

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

(On Sabbatical Leave)

silviom@rcn.com

**Conference's Areas of Inquiry:** *Scalability of PKI; new approach to attribute certificates; and how the required PKI may differ from the PKI traditionally defined.*

### Abstract

In [1], a scalable and small-bandwidth certificate validation scheme was presented. We call this system *NOVOMODO*, to emphasize the *new way* in which it approaches the field.

In this paper, we recall the NOVOMODO technology and

- Compare the efficiency and security of NOVOMODO and OCSP; and
- Discuss how NOVOMODO may simplify PKI management in several applications (e.g., attribute certs).

## 1. Traditional Certificate Validation And NOVOMODO

In essence, a digital certificate  $C$  consists of a CA's digital signature securely binding together several quantities: SN, a serial number unique to the certificate, PK, the public key of the user, U, the user's identifier,  $D_1$ , the issue date,  $D_2$ , the expiration date, and additional fields. In symbols,  $C = \text{SIG}_{CA}(\text{SN}, \text{PK}, \text{U}, D_1, D_2, \dots)$ .

It is widely recognized that digital certificates provide the best form of Internet authentication. On the other hand, they are also difficult to manage. Certificates may expire after one year (i.e.,  $D_2 - D_1 = 1$  year). However, they may be revoked prior to their expiration; for instance, because their holders leave their companies or assume different duties within them. Thus, each transaction enabled by a given digital certificate needs a suitable proof of the current validity of that certificate, and that proof often needs to be archived as protection against future claims.

Unfortunately, the technologies used today for proving the validity of issued certificates do not scale well. At tomorrow's volume of digital certificates, today's validity proofs will be either too hard to obtain in a secure way, or too long and thus too costly to transmit (especially in a wireless setting). *Certificate validation is universally recognized as a crucial problem.* Unless efficiently solved, it will severely limit the growth and the usefulness of our PKIs.

### 1.1 Traditional Certificate Validation

Today, there are two main approaches to proving certificates' validity: Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP).

#### CRLs

CRLs are issued periodically. A CRL essentially consists of a CA-signed list containing all the serial numbers of the revoked certificates. The digital certificate presented with an electronic transaction is

then compared to the most recent CRL. If the given certificate is not expired but is on the list, then everyone knows from the CRL that the certificate is not valid and the certificate holder is no longer authorized to conduct the transaction. Else, if the certificate does not appear in the CRL, then the certificate is deduced to be valid (a double negative).

CRLs have not found much favor; for fear that they may become unmanageably long. (A fear that has been only marginally lessened by more recent CRL-partition techniques.) A few years ago, the National Institute of Standards and Technology tasked the MITRE Corporation [3] to study the organization and cost of a Public Key Infrastructure (PKI) for the federal government. This study concluded that CRLs constitute by far the largest entry in the Federal PKI's cost list.

### OCSP

In the OCSP, a CA answers a query about a certificate  $C$  by returning its own digital signature of  $C$ 's validity status at the current time. The OCSP is problematic in the following areas.

*Bandwidth.* Each validity proof generated by the OCSP has a non-trivial length. If RSA or other factoring based signature schemes are used, such a proof in fact requires at a minimum 2,048 bits for the CA's signature.

*Computation.* A digital signature is a computationally complex operation. In certain large applications, at peak traffic, the OCSP may require computing millions of signatures in a short time, which is computationally very expensive to do.

*Communication (if centralized).* Assume a single validation server implements the OCSP in a centralized manner. Then, all certificate-validity queries would have, eventually, to be routed to it, and the server will be a major "network bottleneck" causing considerable congestion and delays. If huge numbers of honest users suddenly query the server, a disrupting "denial of service" will probably ensue.

*Security (if distributed).* In general, distributing the load of a single server across several (e.g., 100) servers, strategically located around the world, alleviates network congestion. In the OCSP case, however, load distribution introduces worse problems than those it solves. In order to sign its responses to the certificate queries it receives, each of the 100 servers should have its own secret signing key. Thus, compromising any of the 100 servers is

compromising the entire system. Secure vaults could protect such distributed servers, but at *great* cost.

## 1.2 NOVOMODO

NOVOMODO works with standard certificate formats (e.g., X.509v3) and enables a CA to prove the validity status of each certificate  $C$  at any time interval (e.g., every day, hour, or minute) starting with  $C$ 's issue date,  $D_1$ .  $C$ 's time granularity may be specified within the certificate itself, unless it is the same for all certificates. To be concrete, below we assume a one-day granularity for all certificates, and that each certificate expires 365 days after issuance.

*One-way hashing.* NOVOMODO uses a one-way hash function  $H$  (such as SHA [4]) enjoying the following properties:

1.  $H$  is at least 10,000 times faster to compute than a digital signature;
2.  $H$  produces 20-byte outputs, no matter how long its inputs; and
3.  $H$  is hard to invert: given  $Y$ , finding  $X$  such that  $H(X)=Y$  is practically impossible.

### The Basic NOVOMODO System

*Making a certificate  $C$ .* In addition to traditional quantities such as a serial number  $SN$ , a public key  $PK$ , a user name  $U$ , an issue date  $D_1$ , an expiration date  $D_2$  ( $=D_1+365$ ), a certificate  $C$  also includes two 20-byte values unique to it. Specifically, before issuing a certificate  $C$ , a CA randomly selects two different 20-byte values,  $Y_0$  and  $X_0$ , and from them computes two corresponding 20-byte values,  $Y_1$  and  $X_{365}$ , as follows. Value  $Y_1$  is computed by hashing  $Y_0$  once:  $Y_1=H(Y_0)$ ; and  $X_{365}$  by hashing  $X_0$  365 times:  $X_1=H(X_0)$ ,  $X_2=H(X_1)$ , ...,  $X_{365}=H(X_{364})$ . Because  $H$  always produces 20-byte outputs,  $Y_1$ ,  $X_{365}$ , and all intermediate values  $X_j$  are 20-byte long. The values  $Y_0, X_0, X_1, \dots, X_{364}$  are kept secret, while  $Y_1$  and  $X_{365}$  are included in the certificate:  $C=\text{SIG}_{CA}(SN,PK,U,D_1,D_2,\dots,Y_1,X_{365})$ . We shall call  $Y_1$  the *revocation target* and  $X_{365}$  the *validity target*.

*Revoking and validating a not-yet-expired certificate  $C$ .* On the  $i$ -th day after  $C$ 's issuance (i.e., on day  $D_1+i$ ), the CA computes and releases a 20-byte proof of status for  $C$  as follows. If  $C$  is revoked, then, as a proof of  $C$ 's revocation, the CA releases  $Y_0$ , that is, the  $H$ -inverse of the revocation target  $Y_1$ . Else, as a proof of  $C$ 's validity on that day, the CA releases  $X_{365-i}$ , that is, the  $i$ -th  $H$ -inverse of the validity target  $X_{365}$ . (E.g., the proof that  $C$  is valid 100 days after issuance consists of  $X_{265}$ .) The CA may release  $Y_0$  or

$X_{365-i}$  by providing the value in response to a query or by posting it on the World Wide Web.

*Verifying the status of a not-yet-expired certificate C.*

On any day, C's revocation proof,  $Y_0$ , is verified by hashing  $Y_0$  once and checking that the result equals C's revocation target,  $Y_1$ . (I.e., the verifier tests for himself that  $Y_0$  really is the H-inverse of  $Y_1$ .) Note that  $Y_1$  is guaranteed to be C's revocation target, because  $Y_1$  is certified within C. On the i-th day after C's issuance, C's validity proof on that day,  $X_{365-i}$ , is verified by hashing i times the value  $X_{365-i}$  and checking that the result equals C's validity target,  $X_{365}$ . (I.e., the verifier tests for himself that  $X_{365-i}$  really is the i-th H-inverse of  $X_{365}$ .) Note that a verifier knows the current day D as well as C's issuance date  $D_1$  (because  $D_1$  is certified within C), and thus immediately computes  $i=D-D_1$ .

NOVOMODO Security (Sketch)

- *A proof of revocation cannot be forged.*  
The proof of revocation of a certificate C consists of the H-inverse of C's revocation target  $Y_1$ . Because H is essentially impossible to invert, once a verifier checks that a given 20-byte value  $Y_0$  is indeed C's proof of revocation, it knows that  $Y_0$  must have been released by the CA. In fact, only the CA can compute the H-inverse of  $Y_1$ : not because the CA can invert H better than anyone else, but because it computed  $Y_1$  by starting with  $Y_0$  and hashing it! Because the CA never releases C's revocation proof as long as C remains valid, an enemy cannot fake a revocation proof.
- *A proof of validity cannot be forged.*  
On day i, the proof of validity of a certificate C consists of the i-th H-inverse of C's validity target  $X_{365}$ . Because H is essentially impossible to invert, once a verifier checks that a given 20-byte value  $X_{365-i}$  is indeed C's proof of validity on day i, it knows that the CA must have released  $X_{365-i}$ . In fact, only the CA can compute the i-th H-inverse of  $X_{365}$ : not because the CA can invert H better than anyone else, but because it computed  $X_{365}$  by starting with  $X_0$  and hashing it 365 times, thus computing along the way all the first 365 inverses of  $X_{365}$ ! If certificate C become revoked on day i+1, the CA has already released the values  $X_{365-1}, \dots, X_{365-i}$  in the preceding i days (when C was still valid) but has not released and will never release the value  $X_{365-i-1}$  (or any other value  $X_j$  for  $j < 365-i$ ) in the future. Consequently, to forge C's validity proof on day i+1, an enemy should compute on

his own the i+1st H-inverse of  $X_{365}$  (i.e., the H-inverse of  $X_{365-i}$ ), which is very hard to do! Similarly, an enemy cannot compute a validity proof for C on any day after i+1. To do so, it should again be able to invert H on input  $X_{365-i}$ . For instance, if it could compute C's validity proof on day i+2,  $X_{365-i-2}$ , then by hashing it once it would easily obtain  $X_{365-i-1}$ , the H-inverse of  $X_{365-i}$ .

NOVOMODO Efficiency

- *A certificate C includes only two additional 20-byte values,  $Y_1$  and  $X_{365}$ .*  
This is a negligible cost. Recall that C already consists of a CA signature (at least 2048-bit long) of data that includes a public key PK (at least 1024-bit long), and that C may include comments and plenty of other data in addition to SN, PK, U,  $D_1$  and  $D_2$ .
- *Generating  $Y_1$  and  $X_{365}$  requires only 366 hashings total.*  
This too is a negligible cost. Recall that issuing a certificate already requires computing a signature.
- *Proofs of revocation and proofs of validity are only 20-bytes long.*  
Our 20-byte proofs are trivial to transmit and trivial to store, making the 20-byte technology ideal for wireless applications (because here bandwidth is still limited, and so is the storage capacity of many cellular phones and other wireless devices).

NOVOMODO proofs can be so short because they derive their security from elementary cryptographic components, such as one-way functions, which should exhibit an exponential amount of security. (Quite differently, digital signature schemes have complex security requirements. Their typical number-theoretic implementations offer at best a sub-exponential amount of security, and thus necessitate much longer keys.)

NOVOMODO proofs remain 20-bytes long whether the total number of certificates is a few hundred or a few billion. In fact there are  $2^{160}$  possible 20-byte strings, and the probability that two certificates may happen to have a common proof of revocation or validity is negligible.

Note too that the length of our 20-byte proofs does not increase due to encryption or

authentication. Our 20-byte proofs are intended to be public and thus need not be encrypted. Similarly, our 20-byte proofs are self-authenticating: by hashing them the proper number of times they yield either the validity target or the revocation target specified within the certificate. They will not work if faked or altered, and thus need not be signed or authenticated in any manner.

Finally, a 20-byte proof of validity on day  $i$ ,  $X_{365-i}$ , need not additionally include the value  $i$ : in a sense, it already includes its own time stamp! Indeed, as discussed before,  $i$  is the difference between the current day and the certificate's issue day, and if hashing  $X_{365-i}$   $i$  times yields the validity target of certificate  $C$ , then this proves that  $X_{365-i}$  is  $C$ 's proof of validity on day  $i$ .

- *The 20-byte proofs are computed instantly.*  
A proof of revocation  $Y_0$  or a proof of validity  $X_{365-i}$  is just retrieved from memory. (Alternatively, each  $X_{365-i}$  could be recomputed on the fly on day  $i$ ; for instance by at most 364 hashings, if just  $X_0$  is stored during certificate issuance. Surprisingly more efficient strategies are discussed in the next section.)

### NOVOMODO and Wireless

NOVOMODO is ideal for wireless implementations. Its scalability is enormous: it could accommodate billions of certs with great ease. The bandwidth it requires is negligible, essentially a 30-bit serial number for the query and 20-byte for the response. The computation it requires is negligible, because a certificate-status query is answered by a single table look-up and is immediately verified. Of course, great scalability, minimum bandwidth and trivial computation make NOVOMODO the technology of choice in a wireless environment.

But there is another use of NOVOMODO that provides an additional advantage in wireless applications. Namely, every morning --e.g., at midnight-- a wireless user may receive a 20-byte proof of the validity of his certificate for the remainder of the day. (This 20-byte value can be obtained upon request of the user, or pushed to the user's cellular device automatically --e.g., by means of a SMS message or other control message..) Due to its trivial length, this proof can be easily stored in most cellular telephones and PDAs. Then, whenever the user wants to transact on that day, the user simply sends its own certificate *together with* the cert's 20-byte proof of validity for that day. Because the proof

of validity is universally verifiable, the verifier of the cert and proof need not call any CA or any responder. The verifier can work totally off-line. In the cellular environment, in which any call translates into money and time costs, this off-line capability is of great value.

## 2. NOVOMODO vs. OCSP

NOVOMODO and OCSP are both on-demand systems: namely, a user sends a query about the current validity of a certificate and gets back an unforgeable and universally verifiable proof as a response. But there are differences in

- 1) Time accuracy;
- 2) Bandwidth;
- 3) CA efficiency;
- 4) Security; and
- 5) Operating costs.

### TIME ACCURACY

In principle, an OCSP response may specify time with unbounded accuracy, while a NOVOMODO response specifies time with a predetermined accuracy: one day, one hour, one minute, etc. In low-value applications, one-day validity is plenty acceptable. For most financial applications, Digital Signature Trust considers a 4-hour accuracy sufficient. (Perhaps this is less surprising than it seems: for most financial transactions, orders received in the morning are executed in the afternoon and orders received in the afternoon are executed the next business day.) In any event, time is not specified by a real number with infinitely many digits. In an on-demand validation system, a time accuracy of less than one minute is seldom meaningful, because the clocks of the querying and answering parties may not be that synchronized. Indeed, in such a system, a time accuracy of 15 seconds is *de facto* real time.

To handle such an extreme accuracy, NOVOMODO needs to compute hash chains that are roughly 1M long (i.e., needs to compute validity fields of the type  $X_{1M}$ ), because there are at most 527,040 minutes in a year. If chains so long could be handled efficiently, NOVOMODO would *de facto* be real time. Computing 1M hashings is not problematic at certificate issuance: 1M hashings can be performed in less than 1 second even using very reasonable platforms, and a certificate is typically issued only once a year, and not under tremendous time pressure. Similarly, 1 second of computation is not problematic for the verifier of a cert validity proof (e.g., a merchant relying on the certificate) considering that he generally focuses just on an individual transaction,

and has more time at hand. Computing 1M hashings per certificate-status request would, however, affect the performance of the server producing validity proofs, because it typically handles many transactions at a time. Fortunately, this server needs not to compute all these hashings on-line starting with  $X_0$ , but by table look up –capitalizing on having in storage the full hash-chain of every certificate. Nonetheless, storing 1M-long hash-chains may be a problem in applications with huge numbers of certificates. But, fortunately, as we shall mention later on, even ordinary servers can, using better algorithms, re-compute 1M-long hash chains with surprising efficiency.

#### BANDWIDTH

NOVOMODO has an obvious bandwidth advantage over OCSP. The former uses 20-byte answers, while the latter typically uses 256 bytes.

#### CA EFFICIENCY

A validity query is answered by a (complex) digital signature in the OCSP case, and by a (trivial) table look-up in the NOVOMODO case, as long as the CA stores the entire X-chain for each certificate.

Note that, with a population of 1 million certificates, the CA can afford to store the entire X-chain for each certificate when the time accuracy is one day or one hour. (In the first case, the CA would have to store 365 20-byte values; that is, 7.3K bytes per cert, and thus 7.3B bytes overall. In the second case, 175.2B bytes overall.) If the time accuracy were 15 seconds, then each hash chain would consist of 1M 20-byte values, and for the entire system the overall storage requirement would be around 10.5 tera-bytes: a sizable storage.

To dramatically decrease this storage requirement, the CA may store just a single 20-byte value (i.e.,  $X_0$ ) for each cert, and re-compute from it each  $X_i$  value by at most 1M hashings. Alternatively, Jacobsson [5] has found a surprising time/storage tradeoff. Namely, the CA may re-compute all  $n$   $X_i$  values, in the right order, by storing  $\log(n)$  hash values and performing  $\log(n)$  hashings each time. If  $n$  were 1M, this implies just storing 20 hash values per cert and performing only 20 hashings each time the cert needs validation. Other non-trivial tradeoffs are possible. In particular, for our 1M-chain case, Reyzin [R] has shown that a CA can compute all  $X_i$  values ( $i=1M$  down to 1) by storing only 3 hash values and performing at most 100 hashings each time.

In sum, even in a de facto real-time application (i.e., using a 15-second time accuracy) NOVOMODO can,

by just storing 60 bytes per cert, replace a complex digital signature operation with a trivial 100-hash operation.

#### SECURITY AND OPERATING COSTS

The last two differences are better discussed after specifying the type of implementation of NOVOMODO and OCSP under consideration.

#### Centralized NOVOMODO vs. Centralized OCSP: Security Analysis

Whenever proving certificate validity relies on the secrecy of a given key, a secure *vault* ought to protect that key, so as to guarantee the integrity of the entire system. By a centralized implementation of NOVOMODO or OCSP, we mean one in which a single vault answers all validity queries. Centralized implementations are preferable if the number of deployed certificates is small (e.g., no more than 100K), so that the vault could handle the query volumes generated even if almost all certificates are used in a small time interval, triggering almost simultaneous validity queries. In such implementations, NOVOMODO is preferable to OCSP in the following respects.

#### CENTRALIZED NOVOMODO OFFERS BETTER DOOMSDAY PROTECTION

In the traditional OCSP, if (despite vaults and armored guards) an enemy succeeds in penetrating the vault and compromises the secret signing key, then he can both "resurrect" a previously revoked certificate and "revoke" a still valid one. (Similarly, if the CRL signing key is compromised in a CRL system.) By contrast, in NOVOMODO penetrating the secure vault *does not help an adversary to forge the validity of any previously revoked certificate*. In fact, when a certificate becomes revoked at day  $i$ , not only is its revocation proof  $Y_0$  made public, but, simultaneously, all its  $X_i$  values (or at least the values  $X_0$  through  $X_{365-i}$ ) are deleted. Therefore, after a successful compromise, an enemy finds nothing that enables him to "extend the validity" of a revoked certificate. To do so, he should succeed in inverting the one-way hash  $H$  on  $X_{365-i}$  without any help, which he is welcome to try (and can indeed try without entering any secure vault). The worst an enemy can do in a NOVOMODO system after a successful compromise is to fake the revocation of valid certificates, thus preventing honest users from authenticating legitimate transactions. Of course, this would be bad, but *not as bad as enabling dishonest users to authenticate illegitimate transactions*.

### **Distributed NOVOMODO vs. Distributed OCSP: Security and Operating-Cost Analysis**

Centralized implementations of NOVOMODO and OCSP require all queries about certificate validity to be routed to the same vault. This easily results in long delays and denial of service in applications with millions of active certificates. To protect against such congestion, delays, and denial of service, one might spread the load of answering validity queries across several, geographically dispersed, responder servers. However, in the case of the OCSP each additional responder needs to have a secret signing key, and thus needs to be hosted in a vault, making the cost of ownership of an OCSP system very onerous. A high-grade vault meeting the requirements of financial institutions costs at least \$1M to build and \$1M to run. (A good vault would involve armored concrete, steel doors, back-up power generators, protected fuel depot to run the generator for potentially a long time, etc. Operating it would involve a minimum of 4 different teams for 24X7X365 operations, plus managerial supervision, etc.) In an application requiring 10 such vaults to guarantee reasonably fast response at peak traffic, the cost of ownership of the OCSP system would be \$10M of initial investment and an ongoing budget of \$10M/year. Even if less secure vaults and operations were used, millions of dollars in initial and ongoing costs would still be necessary.

In the NOVOMODO case, however, a distributed implementation can be achieved with a single vault (which a CA would have anyway) and an arbitrary number of “un-trusted responders” (i.e., ordinary servers). Let us see the exact details of a distributed NOVOMODO system assuming, to be concrete, that (a) there are 10M certs; (b) there are 1,000 servers, strategically located around the globe so as to minimize response time; and (3) the time granularity is one-day.

#### Distributed NOVOMODO: CA Operations (Initialization Cost)

Every morning: Starting with the smallest serial number, compile a 10M-entry array F as follows: For each certificate C having serial number j, store C's 20-byte validity/revocation proof in location j. Then, date and sign F and send it to each of the 1,000 servers.

#### Distributed NOVOMODO: User Operations (Query Cost)

To learn the status of a certificate C, send C's serial number, j, (and CA ID if necessary) to a server S.

#### Distributed NOVOMODO: Server Operations (Answer Cost)

Every morning: If a properly dated and signed array F is received, replace the old array with the new one.

At any time: answer a query about serial number j by returning the 20-byte value in location j of the current F.

#### Distributed NOVOMODO Works:

##### 1. *Preparing Array F is instantaneous.*

If the whole hash chain is stored for each cert, then each entry is computed by a mere table look-up operation. (Else, it can be computed on the spot by using Reyzin's method.)

##### 2. *F contains no secrets.*

It consists of the accurate and full account of which certificates are still valid and which revoked. (The CA's goal is indeed making this non-secret information as public as possible in the most efficient manner)

##### 3. *Transferring F to the servers is straightforward.*

This is so because F contains no secrets, requires no encryption, and poses no security risks. Though 10M certs are a lot, sending a 200M-byte file to 1000 servers at regular intervals is very doable.

##### 4. *Each server answer is 20-byte long.*

Again, each answer requires no encryption, signature or time stamp.

##### 5. *No honest denial of service.*

Because each value sent is just 20-byte long, because each such a value is immediately computed (by a table look up), and because the traffic can be spread across 1000 servers, no denial of service should occur, at least during legitimate use of the system.

##### 6. *Servers need not be trusted.*

They only forward 20-byte proofs received by the CA. Being self-authenticating, these proofs cannot be altered and still hash to the relevant targets.

#### DISTRIBUTED NOVOMODO OFFERS BETTER CA SECURITY

Distributed NOVOMODO continues to enjoy the same doomsday protection of its centralized counterpart: namely, an enemy successfully entering the vault cannot revive a revoked certificate. Sophisticated adversaries, however, refrain from drilling holes in a vault, and prefer software attacks whenever possible. Fortunately, software attacks, though possible against the distributed/centralized OCSP, cannot be mounted against Distributed NOVOMODO.

In the OSCP, in fact, the CA is required to receive outside queries from untrusted parties, and to answer them by a digital signature, and thus by means of its precious secret key. Therefore, the possibility exists that OSCP's required "window on the outside world" may be maliciously exploited for exposing the secret signing key.

By contrast, in distributed NOVOMODO there are no such "windows:" the CA is in the vault and never receives or answers any queries from the outside; it only outputs non-secret data at periodic intervals. Indeed, every day (or hour) it outputs a file F consisting of public information. (The CA may receive revocations requests from its RAs, but these come from fewer trusted entities via authenticated channels ---e.g., using secure smart cards.) The untrusted responders do receive queries from untrusted parties, but they answer those queries by means of their file F, and thus by public data. Therefore, a software attack against NOVOMODO ordinary responders may only "expose" public information.

### 3. NOVOMODO and Simplified PKI Management

PKI management (e.g., [7] [8]) is not trivial. NOVOMODO may improve PKI management in many applications by

- Reducing the number of issued certs;
- Enabling privilege management on the cert; and
- Sharing the registration function with multiple independent CAs.

Let us informally explain these improvements in PKI management in a series of specific examples. (Note that features and techniques used in one example can be easily embedded in another. We do not explicitly do this to avoid discussing an endless number of possible variations.)

#### 3.1 Turning a Certificate ON/OFF (and Suspending It)

EXAMPLE 1: MUSIC DOWNLOADING  
Assume an Internet music vendor wishes to let users download any songs they want, from any of its 1000 servers, for a \$1/day fee. This can be effectively accomplished with digital certificates. However, in this example, U may be quite sure that he will download music a few days of the year, yet he cannot

predict *which* or *how many* these days will be. Thus the Music Center will need to issue for U a different one-day certificate whenever U so requests: U requests such a certificate and, after payment or promise of payment, he receives it and then uses with any of the 1000 music servers on that day. Issuing a one-day cert, however, has non-trivial management costs both for the vendor and the user. And these costs must be duplicated each time the user wishes to enjoy another "music day."

NOVOMODO technology can alleviate these costs as follows. The first time that U contacts the vendor, he may be issued a certificate C with issue date  $D_1=0$ , expiration date  $D_2=365$ , and a validity field  $X_{365}$ , a revocation target  $Y_1$ , and a suspension field  $Z_{365}$ . (The vendor's CA builds the suspension field very much as a validity field: by starting with a random 20-byte value  $Z_0$  and then hashing it 365 times, in case of one-day granularity. It then stores the entire hash chain, or just  $Z_0$ , or uses a proper time/storage method to be able to generate any desired  $Z_i$ .) At day  $i=1, \dots, 365$ , if U requests "a day of music" for that day, then the vendor simply releases the 20-byte value  $X_{365-i}$  to indicate that the certificate is valid. Else, it releases  $Z_{365-i}$  to indicate that the certificate is "suspended." Else, it releases  $Y_0$  to indicate that the certificate is revoked.<sup>1</sup>

That is, rather than giving U a new single-day certificate whenever U wishes to download music, the vendor gives U a *single, yearly* certificate. At any time, this single certificate can be turned ON for a day, by just releasing the proper 20-byte value. Thus, for instance, NOVOMODO replaces issuing (and embedding in the user's browser) 10 single-day certificates by issuing a single yearly cert that, as it may happen, will be turned ON for 10 out of the 365 days of the year.<sup>2</sup>

#### 3.2 Turning ON/OFF Many Certificates For The Same User

<sup>1</sup> Optionally, if U and the music vendor agree to --say-- a "week of music starting at day i," then either the 20-byte values for those 7 days are released at the proper time, or the single 20-byte value  $X_{365-i-7}$  is released at day i.

<sup>2</sup> The vendor could also use the method above to issue a cert that specifies a priori the number of days for which it can be turned ON (e.g., a 10-day-out-of-365 cert). Because it has a more predictable cost, such certs are more suitable for a gift.

EXAMPLE 2: SECURITY-CLEARANCE MANAGEMENT

Digital certificates work really well in guaranteeing that only proper users access certain resources. In principle, privileges could be specified on the cert itself. For instance, the State Department may have 10 different security-clearance levels,  $L_1, \dots, L_{10}$ , and signify that it has granted security level 5 to a user  $U$  by issuing a certificate  $C$  like

$$C = \text{SIG}_{\text{SD}}(\text{SN}, \text{PK}, U, L_5, D_1, D_2, \dots)$$

Where again  $D_1$  and  $D_2$ , represent the issue and expiration dates.

However, specifying privileges on the cert itself may cause a certificate-management nightmare: whenever its privileges change, the cert needs to be revoked. Indeed, the security level of an employee may vary with his/her assignment, which often changes within the same year. For instance, should  $U$ 's security-clearance level be temporarily upgraded to 3, then the State Department should revoke the original  $C$  and issue a new cert  $C'$ . This task could be simplified somewhat by having  $U$  and thus  $C'$  retain the same public key (and expiration date) as before; for instance, by having

$$C' = \text{SIG}_{\text{SD}}(\text{SN}', \text{PK}, U, L_3, D_1', D_2, \dots)$$

However,  $U$  still faces the task of "inserting" the new  $C'$  into his browser in a variety of places: his desk-top PC, his lat-top, his cell phone, his PDA, etc. Now, having the CA take an action to re-issue a certificate in a slightly different form is one thing, but counting on users to take action is a totally different thing!

This management problem is only exacerbated if short-lived certificates (e.g. certificates expiring one day after issuance) are used. In the context of the present example, single-day certs may enable a State Department employee or user  $U$  to attend a meeting where a higher security level is needed. (If  $U$  had such a cert in a proper cellular device, smart card or even mag stripe card, he could, for instance, use it to open the door leading to the meeting that day.) The use of short-lived certificates is much broader, and has been advocated because it dispenses with the difficulty of revocation to a large extent (no point revoking a cert that will expire in 24hours, at least in most applications). However, issuing short-lived certs so that they reside in all pertinent users' browsers still is a management cost.

These management costs can be alleviated with use of NOVOMODO as follows. Assuming that one-day time accuracy is enough, the State Department issues

to a user  $U$  a certificate containing 10 validity fields and 1 revocation field: e.g.,  
 $C = \text{SIG}_{\text{SD}}(\text{SN}, \text{PK}, U, D_1, D_2, A_{365}, B_{365}, C_{365}, D_{365}, E_{365}, F_{365}, G_{365}, H_{365}, I_{365}, J_{365}, Y_1)$

where the first validity field,  $A_{365}$ , corresponds to security-clearance level 1 ... and the 10th validity field,  $J_{365}$ , corresponds to security-clearance level 10, while, as usual,  $Y_1$  is  $C$ 's revocation field. Cert  $C$  is used as follows. If, on day  $n$ ,  $U$  is in good standing (i.e., cert  $C$  is still valid), and  $U$ 's security-clearance level is 5, then the State Department publicizes (e.g., sends to all its responders in a distributed NOVOMODO implementation) the 20-byte validity proof  $E_{365-n}$ . If, on day  $m$ ,  $U$ 's security-clearance level becomes 2, then the State Department publicizes  $B_{365-m}$ . And so on. As soon as  $C$  becomes invalid (e.g., because  $U$  is terminated as an employee or because  $U$ 's secret key is compromised), then the State Department publicizes  $Y_0$  (and erases "future"  $A, B, C, D, E, F, G, H, I,$  and  $J$  values from its storage).

This way, cert  $C$ , though internally specifying its own privileges, needs *not* be revoked when these privileges change in a normal way, and users need *not* load new certs in their browsers. In essence, NOVOMODO has such minimal footprint, that a CA (rather than issuing, revoking, and re-issuing many related certs) can issue with great simplicity a single cert, having a much higher probability of not being revoked (because changes of security-clearance level do not translate into revocation). As a result, fewer certs will end up been issued or revoked in this application, resulting in simpler PKI management.

In sum,

*NOVOMODO replaces the complex certificate management relative to a set of dynamically changing properties or attributes by a single certificate (with minimum extra length) and a single 20-byte value for attribute.*

Telecom companies may use a method similar to that of Example 2 to switch a given wireless device from one rate plan to another, or for roaming purposes.

### 3.3 Landlord CAs and Tenant CAs

A main PKI cost is associated to the RA function. Indeed, identifying a user  $U$  may require an expensive personal interview and verifying that indeed  $U$  knows the right secret key (corresponding to the to-be-certified public key  $\text{PK}$ ). It would be nice if this RA function could be shared across many CAs,

while enabling them to retain total independent control over their own certs.

EXAMPLE 3: ORGANIZATION CERTIFICATES

The Government and big organizations consist of both parallel and hierarchical sub-organizations: departments, business units, etc. An employee may be affiliated with two or more sub-organizations. For instance, in the U.S. Government, he may work for NIST and the Department of Commerce. Issuing a digital certificate for each such affiliation results in a high total number of certificates and a complex PKI management: every time an employee drops/adds one of his/her affiliations, it is best to revoke the corresponding cert/issue a new one. Ideally, two opposites should be reconciled: (1) The Organization issues only one cert per employee, and (2) Each Sub-Organization issues and controls a separate cert for each of its affiliates.

These two opposites can be reconciled by NOVOMODO as follows. To begin with, notice that NOVOMODO is compatible with de-coupling the process of certification from that of validation, the first process being controlled by a CA and the second by a validation authority (VA). For instance, assuming a one-day time accuracy, once a CA is ready to issue a certificate C with serial number SN, it sends SN to a VA, who selects  $Y_0$  and  $X_0$ , secretly stores the triplet (SN,  $Y_0$ ,  $X_0$ ), computes as usual  $Y_1$  and  $X_{365}$ , and then returns  $Y_1$  and  $X_{365}$  to the CA, who includes them within C. This way, the CA need not bother validating C: the CA is solely responsible for identifying the user and properly issuing C, while the VA is the only one who can prove C valid or revoked. This de-coupling may be exploited in a variety of ways in order to have *organization certificates* that flexibly reflect internal sub-organization dynamics. The following is just one of these ways, and uses Government and Departments as running examples. The Government as a whole will have its own CA, and so will each Department.

Envisaging k different Departments with corresponding CAs,  $CA^1 \dots CA^k$ , and one-day time accuracy, a Government certificate C has the following form:

$$C = \text{SIG}_{\text{GOV}}(\text{SN}, \text{PK}, \text{U}, D_1, D_2, X_{365}, Y_1, [X_{365}^1, Z_{365}^1], \dots, [X_{365}^k, Z_{365}^k])$$

where, as usual, SN is the cert's serial number, PK the public key of the user, U the user's identity,  $D_1$  the issue date,  $D_2$  the expiration date,  $X_{365}$  the validity field,  $Y_1$  the revocation field, and where

$X_{365}^j$  is the validation field of  $CA^j$ ; and  $Z_{365}^j$  is the suspension field of  $CA^j$ .

Such a certificate is generated by the Government CA with input from the Department CAs. After identifying the user U and choosing a unique serial number SN, the issue date  $D_1$ , and the expiration date  $D_2$ , the Government CA sends SN, PK, U,  $D_1$ ,  $D_2$  (preferably in authenticated form) to each of the Department CAs. The  $j$ th such CA then

- chooses two secret 20-byte values  $X_0^j$  and  $Z_0^j$ ,
- locally stores (SN, PK, U,  $D_1$ ,  $D_2$ ,  $X_0^j$ ,  $Z_0^j$ ) or, more simply, (SN,  $X_0^j$ ,  $Z_0^j$ ); and
- returns  $[X_{365}^j, Z_{365}^j]$  for incorporation in the Government certificate in position j (or with "label" j).

This certificate C is managed with Distributed NOVOMODO as follows, so as to work as a 1-cert, a 2-cert, ..., a k-cert; that is, as k independent certs, one per Department. On day n, envisaging 100 responders,

- the Government CA sends all 100 responders the 20-byte value  $X_{365-n}$  if C is still valid, and  $Y_0$  otherwise.
- the  $j$ th Department CA sends all 100 responders the 20-byte value  $X_{365-n}^j$  to signify that C can be relied upon as a j-cert and  $Z_{365-n}^j$  otherwise.

Therefore, the Government CA is solely responsible for identifying the user and issuing the certificate, but each of the Department CAs can *independently* manage what de facto is *its own* certificate. (This is absolutely crucial. If  $CA^1$  were the Justice Department and  $CA^2$  the DOD, then, despite some overlapping interests, it is best that each acts independently of the other.) The resulting certificate system is very economical to run. First, the number of certs is greatly reduced (*in principle*, there may be just one cert for employee). Second, a given employee can leave and join different Departments without the need of revoking old certs or issuing new ones. Third, different Department CAs may share the same responders. (In fact, whenever the mere fact that a given user is affiliated with a given Department is not a secret—something that will be true for most departments-- the servers essentially contain only "publishable information".) Thus a query about the status of C as a j-certificate is answered with two 20-byte values: one as a Government cert and one as a j-cert. This enables one to more nimbly revoke C at a "central level" (e.g., should U lose the secret key corresponding to PK).

POSSIBLE ALTERNATIVES

In the above example, certificate C was only revocable in a central way, but it could easily be arranged that the responsibility of revocation is pushed down to individual Departments. For instance, to enable the jth Department CA, in full autonomy, to revoke as well as suspend C as a j-certificate, C may take the following form:

$$C = \text{SIG}_{\text{GOV}}(\text{SN}, \text{PK}, \text{U}, \text{D}_1, \text{D}_2, [\text{X}_{\text{N1}}^1, \text{Y}_1^1, \text{Z}_{\text{N1}}^1], \dots, [\text{X}_{\text{Nk}}^k, \text{Y}_1^k, \text{Z}_{\text{Nk}}^k]).$$

Also, different departments may have different time accuracies for their own certs. This too can be easily accomplished by having C of the following format,

$$C = \text{SIG}_{\text{GOV}}(\text{SN}, \text{PK}, \text{U}, \text{D}_1, \text{D}_2, [\text{TA}^1, \text{X}_{\text{N1}}^1, \text{Y}_1^1, \text{Z}_{\text{N1}}^1], \dots, [\text{TA}^k, \text{X}_{\text{Nk}}^k, \text{Y}_1^k, \text{Z}_{\text{Nk}}^k]).$$

where

$\text{TA}^j$  is the time accuracy of the jth CA; and  $\text{N}_j$  is the number of time units between  $\text{D}_1$  and  $\text{D}_2$ . (E.g., if  $\text{TA}^j$  is one day and  $\text{D}_1 - \text{D}_2 = 1$  year, then  $\text{X}_{\text{Nj}}^j = \text{X}_{365}^j$ .)

LANDLORD CAs, TENANT CAs, AND LEASED CERTS

Within a single organization, one major advantage of issuing certs structured and managed as above consists in enabling the cert to stay alive though the user moves from one sub-organization to another. It should be realized, however, that the above NOVOMODO techniques are also applicable outside a single-organization domain. Indeed, the Government CA can be viewed as a *landlord CA*, the k Department CAs as *tenant CAs* servicing *unrelated* organizations (rather than sub-organizations), and the certificate can be viewed as a *leased cert*. This terminology is borrowed from a more familiar example where the advantages of “joint construction and independent control” apply. Leased certs are in fact analogous to spec buildings having the identical floor footprints. Rather than building just his own apartment, a builder is better off constructing a 20-floor building, setting himself up in the penthouse apartment and renting or selling out right the other floors. Each of the 20 tenants then acts as a single owner. He decides in full autonomy and with no liability to the builder whom to let into his flat, and whom to give the keys. A 20-story building is of course less expensive than 20 times a single-story one: it may very well cost 10 times that. This economy of scale is even more pronounced in a leased cert. Indeed, the cost of issuing a regular cert

and that of issuing a leased one is pretty much the same. Thus issuing leased certs could be very profitable to a landlord CA, or at least repay it completely of the costs incurred for its own certs. On the other hand, tenant CAs have their advantage too, in fact

1. *they save on issuance costs*: they share the cost of issuing a cert k ways; and
2. *they save on infrastructure costs*: they share the same responders (since they contain only public data).

Natural candidates to act as landlord CAs for external tenant CAs are:

- credit card companies;
- large financial institutions, and again
- the Government (e.g., via the USPS or the IRS).

In many cases, in fact, they have long and close relationships with millions of “users” and may more easily issue them a digital cert without investing too many resources for user identification (e.g., a credit card company has been sending bills for years to its customers, and can leverage this knowledge). A credit card company may like the idea of issuing certificates as a landlord CA in order to run more effectively its own affinity program (having hotel chains, airlines etc. as their tenants). The IRS may have already decided to use digital certificates, and leased certs may later on provide them with a revenue stream that will repay of the costs incurred for setting up a faster and better service.

FURTHER ALTERNATIVES

So far, the way we have described landlord and tenant CAs requires that the landlord CA cooperates with its own tenant CAs during the issuance process, and thus that it has already identified its tenant CAs beforehand. It is actually possible, however, for a landlord CA to issue rental certs envisioning—say—20 tenant CAs, without having identified all or any of these tenants. Rather, future tenant CAs will be able to rent space in already issued certs. This capability is ideal for new cert-enabled applications. Rather than undergoing the expenses necessary to issue certs to millions of customers, a company offering a new certificate-enabled product may approach a landlord CA having issued millions of certs, rent space in them *after the facts*, and then sign on as customers a large portion of the landlord-CA users by *turning ON* all their corresponding certs *overnight* (without any customer identification and other issuing costs) and then starting managing them according to its own

criteria. We shall describe various techniques for enabling this functionality in a forthcoming paper.

## References

- [1] *Efficient Certificate Revocation*; by Silvio Micali; Proceedings 1997 RSA Data Security Conference.
- [2] *Online Certificate Status Protocol*, version 2. Working document of the Internet Engineering Task Force (IETF) RFC 2560.
- [3] Public Key Infrastructure, Final Report; MITRE Corporation; National Institute of Standard and Technology, 1994.
- [4] *Secure Hash Standard*; FIPS PUB 180, Revised July 11, 94 (Federal Register, Vol. 59, No. 131, pp. 35211-34460); revised August 5, 1994 (Federal Register Vol. 59, No. 150, pp. 39937-40204).
- [5] *Low-Cost Hash Sequence Traversal*; by Markus Jakobsson; To appear in Financial Cryptography 2002.
- [6] *General Time/Storage Tradeoffs for Hash-Chain Re-computation*; by Leo Reyzin; unpublished manuscript.
- [7] *Internet Public Key Infrastructure, Part III: Certificate Management Protocols*; by S.Farrell, A. Adams, and W. Ford; Internet Draft, 1996
- [8] *Privacy Enhancement for Internet Electronic Mail – Part II: Certificate-Based Key Management*; by S. Kent and J. Linn; 1989.