Can We Eliminate Certificate Revocation Lists?

Ronald L. Rivest
rivest@theory.lcs.mit.edu
http://theory.lcs.mit.edu/ rivest

MIT Laboratory for Computer Science, Cambridge, Mass. 02139

Abstract. We briefly consider certificate revocation lists (CRLs), and ask whether they could, and should, be eliminated, in favor of other mechanisms. In most cases, the answer seems to be "yes." We suggest some possible replacement mechanisms.

1 Introduction

The notion of a "digital certificate" was introduced by Kohnfelder in his 1978 MIT bachelor's thesis[2]. The idea, now common, was that a certificate is a digitally signed statement binding the key-holder's name to a public key. The signer (or issuer) is often called a certificate authority (CA). Since then, the notion of a certificate has been expanded to include:

- Labeling a public key with a label or attribute, such as a nickname, group name, SDSI name, account number, photo, etc.
- Authorizing a key (or all keys with a given label or name) to do something.

The SPKI/SDSI effort[1, 6] explores some of these varieties of certificates. In any case, a certificate typically specifies the issuer, the subject, an issue date, and an expiration date.

Certificates are an essential component of any infrastructure to support digital signatures. Suppose a "signer" Alice sends a signed message M to an "acceptor" (or "verifier") Bob. Alice signed the message with her private key, and Bob can verify her signature with her public key. The message M might be a piece of email, a request for a copy of a document that Bob has, or a contract.

Presumably, Bob is making some decision about the message, such as whether to ignore it altogether, or to reply to the request it contains. Otherwise why should Alice bother to sign it? And Bob's decision presumably depends on *which* key signed it, and the properties that have been associated with that key. If Bob makes a favorable decision, he *accepts* the message (and sends Alice a reply, or the document, or whatever).

In addition to the message M and her signature on M, Alice may send along other evidence or credentials that will help induce Bob to accept the message. Such evidence typically takes the form of a set of certificates. For example, Alice may include a certificate binding the name "Alice Smith" to her public key. Or, she may include a certificate authorizing her public key to request a copy of a confidential corporate memo.

The standard problem we now address, is that certificates have the potential of going "stale." Alice may change her last name, or be fired from the company. The name-binding certificate or the authorization certificate may no longer be appropriate, and the issuer may thus wish to "revoke" a previously-issued certificate.

Periodically-issued "certificate revocation lists" are one common approach to revoking certificates; each such list specifies what unexpired certificates have been revoked, and when the next CRL will be issued. (See, for example, Menezes et al.[3], section 13.6.3.) The CRL is signed by the issuer. For example, a company might issue a weekly CRL for its employee's certificates. The acceptor has to download the most recent CRL from each relevant CA in order to check the validity of the signer's certificates.

2 The acceptor should set recency requirements

Bob, the acceptor, will care that the certificates that Alice supplies as evidence in support of her message are not stale (e.g. revoked). He wants her evidence to be *recent*; he wants to see recently-issued certificates. Alice may supply day-old, week-old, or year-old certificates. Bob must decide whether they are acceptable evidence or not. He will need to set "*recency requirements*" for the decision he is going to make about whether or not to accept and act upon Alice's message.

We now come to one of the major points of this paper:

Proposition 1: *Recency requirements must be set by the acceptor, not by the certificate issuer (CA).*

The reason is that *the acceptor* is the one who is running the risk if his decision is wrong, not the CA. If "Bob" is an electronic badge-checker at a door to a sensitive room, he may want at most day-old evidence of employment at the company. Weekly-issued CRL's can't meet his requirements.

We thus have as a corollary:

Corollary 1: *Periodically-issued CRL's are wrong, because they are inconsistent with Proposition 1.*

One can also criticize CRL's since they make "negative statements." In line with the principles of good writing, one would prefer only to make "positive statements." One of the goals of this paper is to explore the extent to which a certificate infrastructure can be built entirely around positive statements.

3 The signer should supply all relevant evidence

This section proposes a design principle for certificate infrastructures. The justification for this principle is economy, rather than security. Nonetheless, we give it here as it helps round out our vision for certificate infrastructures.

Proposition 2. *The signer can (and should) supply* all *the evidence the acceptor needs, including recency information.*

Instead of having the *acceptor* query the CA for CRL's or the like, we ask that the *signer* obtain any such necessary evidence, and present it with his signature. We argue that

- The signer can query the CA just as well as the acceptor can.
- The recency information obtained may be useful again later to the signer.
- This structure puts any burden on the signer (the client), rather than on a possibly overworked acceptor (the server).
- In many case, this allows the acceptor (server) to be implemented in a *stateless* manner. For example, Bob can reply to Alice, "Sorry, please make sure that all of your evidence is at most one week old," and then forget about Alice until she comes back again, rather than having to rummage all over the Internet to see if Alice's certificates are still OK. A stateless server design is less vulnerable to denial-of-service attacks.

The only exception to having the signer provide all necessary evidence might be when the acceptor wants an "absolute recency check," for which he can do an on-line check himself.

4 New certificates are the best evidence

If Alice needs to get more recent evidence to convince Bob, then she might just as well get the relevant certificates re-issued.

Proposition 3. *The simplest form of "recency evidence" is just a (more-) recently-issued certificate.*

While this may seem to require an on-line CA, we note that a two-level scheme such as the elegant one suggested by Micali[4] can be used to allow an on-line agent to re-validate certificates previously issued by an off-line CA. Naor and Nissim[5] and Kocher[this proceedings] have related proposals.

This structure seems to give the right sort of behavior. If Bob is willing to take a month-old employment certificate, even if it might now be inaccurate, fine. If he demands a certificate that is at most a day old, then a recently-fired Alice won't be able to provide one.

5 Certificate guarantees

Of course, an issuer may know something about when or how often he might change his mind about the statement he is making in a certificate. A "standard" certificate says something like,

> [Standard certificate guarantee] "*This certificate is good until the expiration date. Unless, of course, you hear that it has been revoked.*"

Not a very useful statement. The acceptor is *always* required to check to see if a certificate has been revoked. SPKI/SDSI is at the other end of the spectrum:

[SPKI/SDSI certificate guarantee] "*This certificate is good until the expiration date. Period.*"

This is much better for the acceptor; he *never* has to check whether a certificate has been revoked. In many cases, this can be made to work well enough by having certificates with reasonably short validity periods, or by restricting such certificates to applications where the issuer has the authority to make his statement valid for the stated validity period. However, in other cases an issuer might be tempted to have validity periods that are too long, and cause an acceptor to suffer some undue risk.

In addition, the issuer may not be willing to re-issue certificates arbitrarily; there may be some limits to what he is willing to do. We imagine that a more general guarantee might take this into account, as well as any other information about when the validity of the statement might change. We suggest the following sort of guarantee.

[Proposed general certificate guarantee] "*This certificate is definitely good from (date-time-1) until (date-time-2). The issuer also expects this certificate to be good until (date-time-3), but a careful acceptor might wish to demand a more recent certificate. This certificate should never be considered as valid after (date-time-3).*"

The certificate goes through three phases: (1) guaranteed, (2) probable, and (3) expired, in contrast to the standard certificate, which goes through (1) probable (2) expired, or the SPKI/SDSI certificate, which goes through (1) guaranteed, (2) expired. An acceptor may reasonably accept a certificate in its "probable" phase if there is little at risk. If necessary, the acceptor may insist that the signer go back to the issuer and get a certificate that is still in its guaranteed phase.

We conjecture that such a structure can yield great benefits in helping both signers and acceptors clearly define their intentions and security policies. By giving certificates a nonempty guaranteed phase, the issuer is informing potential acceptors about the length of time that the certificate is necessarily valid, and simultaneously protecting himself from needless queries for more recent certificates within this period. This is definitely an improvement over CRL's, whose certificates have no guaranteed period. Furthermore, the probable phase allows low-value or low-risk transactions to proceed without needless checking.

6 Key Compromise is Different

One of the standard reasons why a certificate might be revoked is that there is evidence that the key-holder has lost control of, or lost, his private key. We suggest that this issue should be separated out and handled differently. Ordinary certificates should not be revoked merely because the key is compromised. Rather, the signer should present separate evidence to the acceptor that the key has *not* been compromised. Since, in this framework, the no-compromise evidence is separate, the ordinary certificates can continue to be "valid" even

though the key has been compromised. In this way, ordinary CA's do not have to deal with this issue at all.

How does this work?

First of all, we propose that the "standard" way of declaring a key to be compromised is to publish a note, *signed by that key* declaring it to be compromised, or dead. We call this note a "suicide note." PGP uses this mechanism. So the key-holder can notify anyone that his key has been compromised by signing and distributing such a note. The key-holder might save such a note in a safe place, in case his private key is lost. Note that if a key gets published on the Internet, anyone can sign such a suicide note.

There is a very interesting question, which is seldom addressed in the conventional certificate literature, regarding the question as to *who should be allowed to declare a key as compromised or lost, and on what basis?* Clearly the original owner of the key should be able to; but who else?

For example, suppose Bob makes up a public-key pair, and then registers to use this pair with an on-line service that charges an annual fee. In return for his fee, Bob obtains an authorization certificate that enables him to use the service. Bob then distributes the private key to all of his friends so that they may obtain the service for free. Should the service, which now has evidence that Bob has distributed his key, be able to declare the key as compromised? Perhaps. It is reasonable to suppose that some services might want to receive an escrowed copy of a suicide note in return for the authorization certificate they are issuing, so that the signer would be discouraged from sharing his key. We'll see a slightly better approach in a second.

Now, back to the basic scenario. Alice presents a signed message M to Bob, together with a collection of certificates. Bob is happy with the certificates, but wants to know if Alice's key has been compromised. How can Alice convince Bob that her key has *not* been declared compromised?

We propose the use of a new kind of agent, called a "key compromise agent" (KCA), or a "suicide bureau"(SB). There may be many such agents; they form an association that certifies its members, and they share a high-speed reliable network. When Alice creates her public key pair, she registers the public key with one such SB. The suicide bureaus listen attentively to their network and elsewhere for any suicide notes signed by keys belonging to users registered with them. Anyone who receives a suicide note can forward it to any SB, who will broadcast it on the SB network.

Alice's SB can then give Alice a "certificate of health" of the form:

> [Certificate of health] "*The public key (....) was registered with this bureau on (date). Since then, no evidence has been received that the key has been lost or compromised.*"

Alice can then present this certificate of health to Bob with her other certificates. Bob can demand a more recent health certificate, if he wishes.

With this, we see that we have now eliminated CRL's entirely, and can have a signer present evidence of the desired sort entirely in the form of certificates.

The signer and acceptor can negotiate the recency required on these certificates, and the signer can go off and obtain more recent ones as necessary.

As a final wrinkle, we note that the signer might not need to give a suicide note to a service provider. A weaker form of delegation might be better. The signer could sign an authorization certificate authorizing the service provider to transmit a "bill of bad health" to the SB's. The SB might then no longer issue a certificate of health to the signer. This is not much different than giving a suicide note in escrow to the service provider, but it does help the SB's clearly identify the source of their information. (A suicide note sent by anonymous email would be acted upon, but it is preferable to have the sources of such information identified when possible. The authorization certificate allows that.)

7 Conclusions

We see that one can do without CRL's. Indeed, it is possible to organize a certificate infrastructure so that a signer can present just a collection of certificates to the acceptor as evidence in support of the signature and the signed message. The acceptor and signer might negotiate about the recency of some of the certificates, in which case it is the signer's responsibility to get more recent replacements. We suggest that such a framework would be an improvement over the use of CRL's.

Acknowledgments

I'd like to thank Stuart Stubblebine for useful comments and suggestions. The reader might profitably refer to some of his very relevant related work [7].

References

1. Carl M. Ellison. SPKI certificate documentation. (See `http://www.clark.net/pub/cme/html/spki.html`), 1998.
2. Loren M. Kohnfelder. Towards a practical public-key cryptosystem. B.S. Thesis, supervised by L. Adleman, May 1978.
3. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
4. Silvio Micali. Efficient certificate revocation. Technical Report TM-542b, MIT Laboratory for Computer Science, March 22, 1996.
5. Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. In *Proceedings 7th USENIX Security Symposium (San Antonio, Texas)*, Jan 1998.
6. Ronald L. Rivest and Butler Lampson. SDSI–a simple distributed security infrastructure. (see SDSI web page at `http://theory.lcs.mit.edu/ cis/sdsi.html`).
7. 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. (Oakland).