Certificate Revocation the Responsible Way

Jonathan K. Millen
SRI International
millen@csl.sri.com

Rebecca N. Wright
AT&T Labs – Research
rwright@research.att.com

Abstract

Public-key certificates are managed by a combination of the informal web of trust and the use of servers maintained by organizations. Prompt and reliable distribution of revocation notices is an essential ingredient for security in a public-key infrastructure. Current schemes based on certificate revocation lists on key servers are inadequate. An approach based on distributing revocation notices to "dependers" on each certificate, with cascading forwarding, is suggested. Research is necessary to investigate architectural issues, particularly reliability and response time analysis.

1. Introduction

The more the Internet is used for important personal and commercial transactions, the more important are the basic security concerns of privacy and authentication of network communications, and an increasing number of organizations and individuals are affected. The fundamental tool for communication security is encryption. Encryption was largely the province of the military until the late 1970s, when the U.S. Data Encryption Standard (DES) was published for use in commerce and banking [2]; and public-key encryption, in the form of the Diffie-Hellman [3] and RSA [9] algorithms, was also invented. Public-key encryption made the key distribution problem easier, at least at first sight, since the key used to encrypt messages for a particular recipient did not have to be kept secret—it could be published freely. The corresponding decryption key was kept by the owner. This meant that courier services to carry secret master keys from one party to another were no longer necessary, and furthermore, for all pairs of n parties to communicate privately, only n rather than $n(n-1)/2$ keys had to be distributed.

Public-key cryptography first became easily accessible to the general public through a software package called PGP (Pretty Good Privacy), developed for UNIX and PCs and available for free [6]. It enabled users to generate their own public and corresponding secret keys, and put the keys to use with the RSA algorithm. If the sender and receiver of a message had the PGP software, it could be used either to send private messages or to authenticate messages with a digital signature. For digital signatures, the sender applies his own secret key to a message, and the recipient can confirm the identity of the sender by applying the sender's known public key.

Because public-key algorithms are typically slower than symmetric-key algorithms, public-key encryption is used primarily to distribute symmetric keys, and the symmetric keys are applied to the protected data using DES and other methods.

As RSA and other public-key techniques were judged over time to be as secure as established symmetric-key algorithms, their use spread to organizations and the government itself. The value of data protected by these techniques increased, focusing more attention on a problem that still remained with public key distribution: in order to ensure that encrypted data is only going to the proper recipient, and that digitally signed data is coming from the proper originator, the user of a public key has to be sure that it belongs to the right party. How can a public key be authenticated?

2. Public-Key Certificates

The immediate answer is to create a *certificate*, which is a signed message linking a public key to the identity of its owner. A certificate should be digitally signed by someone whose public key is already known, and who is trusted not to sign a certificate without confirming that the named owner actually possesses the corresponding secret key. In this way, a small number of public keys can be exchanged in a secure "out-of-band" manner, such as face-to-face or by secure courier; these keys can be "bootstrapped" to validate additional keys. This dependence of one key on another creates a public-key certificate infrastructure. There have been a number of proposals for the architecture of such infrastructures, ranging from an Internet standard for Privacy-Enhanced Mail (PEM) to the PGP-motivated informal "web of trust."

Note that a certificate might be issued either by an individual or an organization. An organization that signs public key certificates is called a *certification authority.* If an organization issues a certificate, it is usually because it conveys some privileges, such as access to an internal network, or to files in a certain category, or management authority on certain tasks. In this case the organization as well as the individual would hold the secret key. Most private organizations who have set up their own enterprise-level certificate systems, like PEM, make use of the X.509 standard for certificate format. Certificate dependencies form a hierarchy.

In PGP terminology, an individual who signs someone else's certificate is an *introducer.* There are no rules for the network of dependencies in the web of trust; it may have cycles, for example. The same certificate may have several signatures on it; in fact, multiple signatures are considered desirable, to increase the chance that someone who receives the certificate will find a trusted introducer among them, without having to trace back through a chain of certificates.

Public keys and their certificates may become invalid for various reasons. The secret key may have been lost or compromised. The owner's identifying information, which might include an Internet e-mail address or employer, may have changed. The certificate might have been used to enable organizational privileges which have been withdrawn by the employer. Some certificates have an expiration date, but if there is none, or if a certificate becomes invalid prior to the expiration date, there should be some way to revoke the certificate. Holders of a certificate must be notified that it is no longer valid, otherwise they will continue to use the same public key to encrypt messages to the owner, and accept signatures from the owner as proof of identity and associated privileges.

There is some logical structure here that needs careful investigation. The signature on a certificate is no good unless the recipient both trusts the signer and holds the certificate of the signer. If the signer's certificate is revoked, that does not invalidate certificates already signed, but new ones should not be accepted if they are signed by the revoked key. The timing issues can be subtle; a logic to support suitable policies was developed by Stubblebine and Wright [10]. Should revocation notices be accepted if they are not signed by the key in the certificate? In some sense, they need to be, since loss or destruction of the owner's copy of a key is reason for the owner to revoke it. These are general concerns about the principles of revocation, and they can be considered separately from questions about how to manage distribution of revocation notices.

Some of the questions that have to be asked about public-key certificate management, therefore, are:

- Was the key issued by the individual owner or an organization?
- Is the certificate signed by an introducer or a certificate authority?
- Was the key obtained from a server or via non-network channels?
- Does the key convey privileges or authority?
- What are the causes and consequences of revocation?
- How is the distribution of revocation notices handled?

In this note we focus on distribution of revocation notices, and how it can be handled in an environment where the other questions can have several possible answers.

3. Handling Revocation

Current proposed standards for revocation, such as in the Internet draft standard Public Key Infrastructure [7], involve *certificate revocation lists* (CRLs) maintained on key servers. A key server is a stable, reliable computer system accessible over the network that distributes certificates. To revoke a certificate, the owner sends the key server a *revocation notice*, which is a signed message identifying the certificate to be revoked.

Upon receipt of a revocation notice, the key server updates its CRL and no longer gives out the revoked certificate. Current certificate holders might be notified either with a "push" or a "pull" system. As an example of a "push" system, a key server could periodically broadcast its current CRL to a set of subscribers. A "pull" approach would be to ask a certificate holder who obtained the certificate from a particular key server to check with the key server periodically, or before using a certificate.

Any scheme involving periodic revocation updates, whether of a push or a pull nature, poses a risk due to the delay between the time the certificate is revoked (which may itself be some time after the cause for revocation took place) and the time of the update.

Push schemes are impractical if they have to broadcast an entire CRL update to a large number of subscribers. One can try to reduce the need for revocation by limiting certificates to brief expiration periods, but this approach leads to a more complex system that still requires a "suicide bureau" to maintain revocations due to key compromise, according to Rivest [8]. It also means that replacement certificates must be distributed.

Pull schemes require action on the part of the certificate holder. It is burdensome to check for revocation on every use, and often it is the owner rather than the certificate holder who suffers more from a failure to check revocation. The certificate holder might not always know who, or which server, to consult about the status of a certificate, particularly if there are several different kinds of certificates with different management characteristics.

For a key server that handles a large number of certificates and queries, there can be significant efficiency considerations for updating and accessing a CRL. Solutions such as such as Kocher's revocation tree [4] have been proposed for improved efficiency.

From a social point of view we want to acknowledge the fact that many certificates are issued by individuals, perhaps using PGP, and distributed without the use of a key server. Certificates might be posted on web servers to publicize them, but web servers typically do not support key server responsibilities such as CRL maintenance.

We need a better way to handle revocations. Some desired properties of a revocation scheme are:

- It should be workable for individuals.
- It should be "server-light," so that massive institutional facilities will not be required.
- It should be decentralized.
- It should support prompt revocation, given specified delay constraints.
- It should require only a realistic workload for those using the system.
- The workload should be allocated in proportion to the self-interest of users.

4. Suggestion: Dependers

One way to balance these goals is suggested here. The idea is to associate a list of "dependers" with a certificate. A depender may be either an individual who wants to be notified promptly of revocation if and when it occurs, or a server that should update its data base upon notification.

The basic advantage of a depender list is that it narrows the burden of notification to the minimal set of interested parties. It becomes practical, also, to perform push-style notification immediately, rather than delaying for a periodic schedule.

The basic depender concept is illustrated in Figure 1.

Notification would not be practical if the owner of a key had to notify everyone who had the key. Everyone with a copy of the PGP software, for example, has the certificate of the originator, Phil Zimmerman, and he would not and could not put everyone on his depender list.

The burden of notification can be distributed by requiring the depender list to cover only an immediate set of those who received the key directly, and invoking transitivity to cover the network. With some exceptions, we could make a rule saying that anyone who transmits a certificate should put the recipient on a depender list. That is, if A sends a certificate to B,

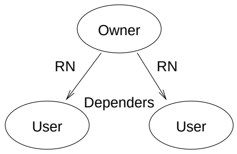

Figure 1: Dependers (RN: Revocation Notice)

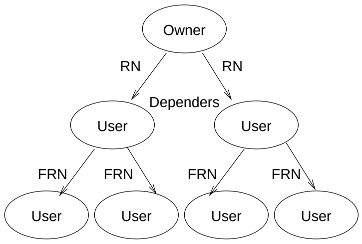

Figure 2: Cascading (FRN: Forwarded Revocation Notice)

then A puts B on A's depender list for that certificate, regardless of who owns the certificate or where it came from.

A cascaded or hierarchical architecture for forwarding revocation notices is illustrated in Figure 2.

The responsibility to maintain a depender list can be avoided in some cases by sending the revocation notice to a server. The server could operate on either a push or a pull model, implementing the push side as a forwarder of revocation notices, and, on the pull side, allowing users to query the status of a particular certificate. The server approach is illustrated in Figure 3.

The server approach is still decentralized in that different key owners may choose different servers on which to make their certificates available. Also, the server does not necessarily have to have a single combined CRL; it just has to be able to respond to queries about individual certificates, using information that may be stored in association with each certificate (or each user who might have a certificate).

Given the existing mixture of certificate management methods in the Internet, we will probably need a hybrid scheme that results from combining the cascading and server approaches, as suggested in Figure 4.

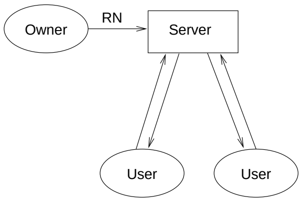

Figure 3: Using a Server

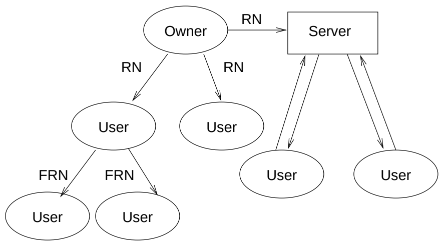

Figure 4: Hybrid Certificate Management

5. Implications

What would it take to implement a depender architecture? First, when a public key certificate is requested and sent from a holder to a user, the user has the option of asking to be a depender, and the holder has options of recording that dependency, or referring the user to a server or some depender who can forward the notice, or just refusing. This leads to a kind of protocol:

A $\rightarrow$ B: Please send me the certificate of: ____
B $\rightarrow$ A: (public key certificate)
A $\rightarrow$ B: Please register me as a depender
B $\rightarrow$ A: (Option 1) Okay
B $\rightarrow$ A: (Option 2) Check the server at URL: ____
B $\rightarrow$ A: (Option 3) Ask one of my dependers: ____
B $\rightarrow$ A: (Option 4) No

Such a protocol would have to be implemented in some form by PGP and by certificate servers.

5.1. PGP Implications

PGP already has many commands for implementing various aspects of key and certificate management. It would not be inconsistent with its overall architecture to provide commands for adding dependers and for generating and revocation notices to them. Certificates are kept in data bases called "key rings;" the format of a key ring would have to be extended to hold depender information. Since revocation notices would be sent by e-mail, it would still be a task for individuals to respond to revocation notices by invoking the PGP command to process them.

5.2. Web Server Implications

Certificates are often posted by individuals on web pages on servers maintained by their organizations. The server itself may have no centralized awareness that some of its pages contain certificates. It is still possible, however, for a page containing a certificate to be updated automatically in response to a revocation notice. The notice could be sent in the form of an HTTP query invoking a CGI (Common Gateway Interface) script. CGI scripts are programs run by the server in response to queries similar to web page requests [1]. A CGI query specifies a particular script to use (an executable file in a CGI directory on the server) and gives it parameters relevant to the particular query, in this case a certificate ID. The CGI script can authenticate the request, access the web page containing the certificate, and erase it or mark it. Standard CGI scripts could be developed for this purpose, given conventions for the organization of web pages containing certificates.

5.3. Authentication Implications

In order to avoid spurious revocations, it is important that revocations be authenticated in some way. CRLs are typically signed by the server that issues them. In our case, we propose that revocation notices of an individual's public key should be signed, when possible, by the corresponding private key; forwarded revocation notices would maintain the initial signature. In this way, it can be determined that the revocation notice either came from the owner of the key and should therefore be trusted, or the revocation notice came from someone who knows the private key (or who knows how to forge signatures from that key), in which case the key is by definition compromised and should be revoked. One advantage of this method is that since the key used to verify the revocation notice is the same as the key that is being revoked, a user receiving a revocation notice will always have the capability to verify a signature to revoke any key in their key ring.

The only case where the above method is not be possible to use is when a public key is being revoked because the private key has been lost. In this case, here are three possibilities. One is that the user is required to first obtain a new set of keys and use these to authenticate the revocation message; this has the disadvantage of requiring the new key to be disseminated before the old key can be revoked. The second is to allow unsigned revocation notices; this has the disadvantage that it allows denial-of-service attacks by which anyone can arbitrarily revoke anyone else's keys. Finally, revocation notices could be signed in the same way that other certificates are, by some combination of servers and other individuals. In practice, a combination of the three may be the best solution, where the trust others will place in the revocation notice will depend on which keys have signed it.

5.4. Reliability Implications

A cascading scheme for forwarding revocation notices has the difficulty that it depends on the correct and prompt operation of individuals or servers. In general, one should assume that at least some of links in a chain of notices will be down at a given time. Hence it is necessary to develop architectures and dependency-creation protocols that build some redundancy into the forwarding hierarchy, so that it will be more reliable than the individual nodes. We will need analysis of the reliability of a planned architecture. Also, if reliability concerns are modeled in terms of delays rather than simply operation vs. non-operation, there can and should be analysis of the propagation time for a revocation notice.

5.5. Summary

Public-key cryptography is in widespread use for both individual and institutional applications. Public keys need to be certified for authenticity and for the authority behind any associated privileges. Certificates are managed by some combination of the informal web of trust and the use of servers maintained by organizations.

Prompt and reliable distribution of revocation notices is an essential ingredient for security in a public-key infrastructure. Current schemes based on certificate revocation lists on key servers are inadequate. An approach based on distributing revocation notices to "dependers" on each certificate, with cascading forwarding, has been suggested. This can be combined with enhanced web server and key server services in a hybrid approach. Research is still necessary to investigate architectural issues, particularly reliability and response time analysis.

References

[1] National Center for Supercomputing Applications, "Common Gateway Interface," University of Illinois at Urbana-Champaign, http://hoohoo.ncsa.uiuc.edu/cgi.

[2] National Institute of Standards and Technology (NIST), "The Data Encryption Standard (DES)," Federal Information Processing Standard 46 (FIPS PUB 46).

[3] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory* 22(6), 1976, 644-654.

[4] P. Kocher, "On certificate revocation and validation," *Financial Cryptography*, LNCS 1465, 1998, 172-177.

[5] "Privacy Enhancement for Internet Electronic Mail," RFC 1421-1424, Internet Engineering Task Force (IETF).

[6] P. Zimmermann, *The Official PGP User's Guide,* MIT Press, 1995.

[7] C. Adams and R. Zuccherato, "Internet X.509 Public Key Infrastructure Data Certification Server Protocols," Internet Draft, PKIZ Working Group, 1998.

[8] R. Rivest, "Can we eliminate certificate revocation lists," *Financial Cryptography*, LNCS 1465, 1998, 178-183.

[9] R. Rivest, A Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM* 21(2), Feb. 1978, 120-126.

[10] S. Stubblebine and R. Wright, "An authentication logic supporting synchronization, revocation, and recency," *Proc. 3rd ACM Conference on Computer and Communications Security*, March 1996.

[11] "The Directory-Authentication Framework," CCITT Recommendation X.509.