91

Preserving Public Key Hierarchy

91.1 Public Key Infrastructure (PKI) 1176
Standard PKI Hierarchical Construction • The Impact of a Root Key Compromise

91.2 Constructing Cryptographically Secure Digital Timestamps 1177
Timestamp • Hash of the Certificate • Certificate Authority Certificate Hash • Digital Signature of the Time Authority

91.3 Separation of Hierarchies 1178

91.4 Walk-Through of Issuance of a Certificate Containing a CSDT 1179

91.5 Recovery Walk-Through 1180

91.6 Known Issues 1180

91.7 Summary 1181

Bibliography 1181

Geoffrey C. Grabow

Public key infrastructures (PKIs) have always been designed with a top-level key called a root key. This single key is responsible for providing the starting point of trust for all entities below it in the hierarchy. If this root key is ever compromised, the entire trust hierarchy is immediately questionable.

The root key is primarily responsible for digitally signing subordinate Certificate Authorities (CAs). A compromise of the root means that an unauthorized CA will appear perfectly valid to users. Users will then engage in a transaction completely unaware that the security upon which they are relying is worse than worthless.

This single root key introduces a single point of failure.

It is a standard practice in security to design and build systems with a series of checks and balances to prevent any one part of the system from causing a catastrophic failure. However, this practice, for all practical purposes, has been ignored when it comes to a hierarchical PKI.

It is the intention of this chapter to propose a system in which this single point of failure is removed.

Cryptographically secure digital timestamps (CSDTs) have been used for a wide variety of purposes, including document archiving, digital notary services, etc. By adding a CSDT to every digital certificate issued within a PKI, one now has a method for ensuring not only that the certificate is valid, but also at what point in time that validity was declared.

When properly configured, certificates within a PKI, which are protected using CSDTs, can survive the compromise of the root key. If the root key is exposed, certificates still have their original value, and all that is lost is the ability to create new certificates. This allows transactions to continue, and the recovery process only requires the replacement of the root key.

AU7495—Chapter91—25/1/2007—21:08—PARTHIBAN—14751—XML MODEL CRC12a – pp. 1175–1181.

A significant advantage of the system proposed herein is that it works within the parameters set forth in existing PKI standards.

91.1 Public Key Infrastructure (PKI)

Public key (or asymmetric) cryptography uses two different keys, usually referred to as a public key and a private key. Any information encrypted by K_{PUB}(Recipient) can only be decrypted by K_{PRI}(Recipient), and vice versa. The two keys are mathematically linked and it is computationally infeasible[1] to determine the private key from the public key. This allows the recipient to create a key pair and to publish K_{PUB}(Recipient) in a location that anyone can find it. Once the sender has a copy of K_{PUB}(Recipient), encrypted information can be sent to the recipient without the problem of transporting a secret key.

Sender:

$$\text{DATA} + K_{PUB}(\text{Recipient}) + \text{Encryption algorithm} = EK_{PUB}(\text{Recipient})[\text{Data}]$$

Recipient:

$$EK_{PUB}(\text{Recipient})[\text{Data}] + K_{PRI}(\text{Recipient}) + \text{Decryption algorithm} = \text{Data}$$

The reverse of this process is also true. If the recipient encrypts data with K_{PRI}(Recipient), it can be decrypted with K_{PUB}(Recipient). This means that anyone can decrypt the information and confidentiality has not been achieved; but if it can be decrypted using K_{PUB}(Recipient), then only K_{PRI}(Recipient) could have encrypted it, thereby identifying the individual[2] who sent the data. This is the principle behind a digital signature. However, in a true digital signature scheme, only a hash of the data is encrypted/decrypted to save processing time.

91.1.1 Standard PKI Hierarchical Construction

While asymmetric key systems have solved the key management problem in traditional symmetric key systems, they have introduced a new problem called "trust management." This problem raises the question of "How can I be sure the public key I am using really belongs to the intended recipient?" This problem, typically referred to as a man-in-the-middle attack, happens when a third party (attacker) introduces its public key to the sender, who is fooled into believing that it is the public key of recipient, and vice versa. Obviously, this would allow the attacker to read and potentially modify all communication between the sender and the recipient without either of them being aware of the attacker whatsoever.

This problem is solved through the use of a Certificate Authority. The CA digitally signs a certificate that belongs to the sender and another certificate that belongs to the recipient. The certificate includes the name and public key of its owners, the integrity of which can be checked through the use of the CA's public key. Unfortunately, that means that the sender and the recipient must belong to the same CA. If they are not members of the same CA, a hierarchy of CAs must be established (see Exhibit 91.1).

Each entity in Exhibit 91.1 has its own certificate that is signed by an entity higher up in the hierarchy. This is the method used to transfer trust from a known entity to one that is unknown. The exception to this is the Root, which creates a self-signed certificate. The Root must establish trust through direct contact and business relationships with the CAs.

[1] "Computationally infeasible" indicates that the time or resources required to determine the private key, given only the public key, are well beyond what is available.

[2] This assumes that the private keys are generated, used, stored, and destroyed in a secure and proper manner.

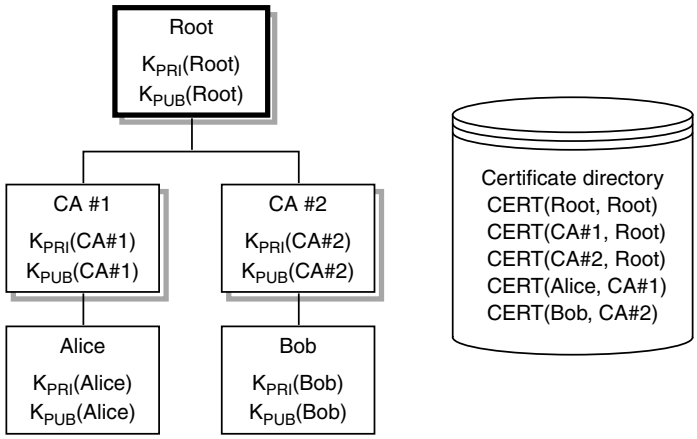

EXHIBIT 91.1 Basic PKI hierarchy.

In this environment, Alice can digitally sign a document and send it to Bob, along with a copy of her certificate as well as the certificate of CA#1. Because Bob already has a trust relationship with CA#2, and CA#2 has a trust relationship with the Root, Bob can validate the certificate of CA#2 and then validate Alice's certificate. Once Bob trusts Alice's certificate, he believes that anything that he can verify with Alice's public key must have been signed by Alice's private key, and therefore must have come from Alice.

91.1.2 The Impact of a Root Key Compromise

The problem with this hierarchical construction is the total reliance on the security of the Root private key. If the K_{PRI}(Root) is compromised by an attacker, that attacker can create a fraudulent CA#3, and then fraudulent users under that CA. Because CA#3 can be positively validated using the public key of the Root, Alice, Bob, and everyone who trusts the Root will accept any users under CA#3. This puts Alice, Bob, and everyone else in this hierarchy in a situation in which they are trusting fraudulent users and are unaware that there is a problem.

If this occurs, the entire system falls apart. No transactions can take place because there is no basis for trust. An even more significant impact of this situation is that as soon as Alice and Bob are informed about the problem, they will not only stop trusting users under CA#3, but also not be able to trust anyone in the entire hierarchy. Because a CA#3 was created fraudulently, any number of fraudulent CAs can be created and there is no way to determine the CAs not to be trusted from those that should be.

If one cannot determine which CAs are to be trusted, then there is no way to determine which users' certificates are to be trusted. This causes the complete collapse of the entire hierarchy, from the top down.

91.2 Constructing Cryptographically Secure Digital Timestamps

Cryptographically secure digital timestamps (CSDTs) are nothing new. A wide variety of applications have been making use of secure timestamps for many years. It is not the intention of this chapter to delve into the details of the actual creation of a CSDT, but rather to indicate the minimum required data for inclusion within digital certificates.

91.2.1 Timestamp

Of course, because one of the primary components of a CSDT is the timestamp itself, a "trusted" time source is required. This can be achieved in several accepted methods and, for the purposes of this construct, it will be assumed that the actual timestamp within the CSDT is the correct one.

To allow for high-volume transaction environments, a 16-bit sequence number is appended to the timestamp to ensure that there can be no two CSDTs with the identical time. This tie-breaker value should be reset with each new timestamp. Therefore, if the time resolution is 0.0001 seconds, it is possible to issue 65,536 CSDTs that all happen within that same 0.0001 second, but the exact sequence of CSDT creation can be determined at any future time.

91.2.2 Hash of the Certificate

For a CSDT to be bound to a particular certificate, some data must be included to tie it to the certificate in question. A hash generated by a known and trusted algorithm, such as SHA-1 or MD5, is used to provide this connection. This is the same hash that is calculated and encrypted during the Certificate Authority signing process.

More importantly, it is critical to know that the time in the CSDT is the time when the CA signs the certificate. Therefore, not just the hash of the certificate should be included, but rather the entire digital signature added to the certificate by the CA. Using the CA's signature will also provide for future changes in CA signing standards.

However, because one of the goals of this chapter is to provide a new feature to existing certificate standards without changing the standards, one cannot append information to the certificate after the signature. Rather, the CSDT must be added to the certificate prior to it being signed by the CA and inserted into an x.509v3 extension field.

91.2.3 Certificate Authority Certificate Hash

As an additional measure, the hash of the CA's certificate is embedded in the CSDT to provide a record of which CA made the request to the Time Authority (TA).

91.2.4 Digital Signature of the Time Authority

To prevent tampering, the CSDT must be cryptographically sealed using a standard digital signature. Because the total amount of data in a CSDT is small, this can be accomplished by simply encrypting the data fields with the private key of the TA. However, to allow for growth and additional fields to be added in the future, it is better to encrypt a hash of all of the data to be secured.

91.3 Separation of Hierarchies

Of course, the x.509 standard already includes a timestamp so it can be determined at what date and time a certificate was signed by its CA. However, if the root private key was compromised and a fraudulent CA is created, that CA could simply set the time to any value desired prior to signing the certificate.

What is proposed is the inclusion of a timestamp signed by an authority that exists outside the hierarchy of which the CA is part (see Exhibit 91.2).

When a CA creates a certificate, it would follow its normal process for acquiring the public key and other data to be included in the certificate. However, prior to signing the certificate, it would request a CSDT from the Time Authority (TA). This CSDT would then be generated by the TA and returned to the CA. The CA would add the CSDT to the certificate, then sign it in the usual manner.

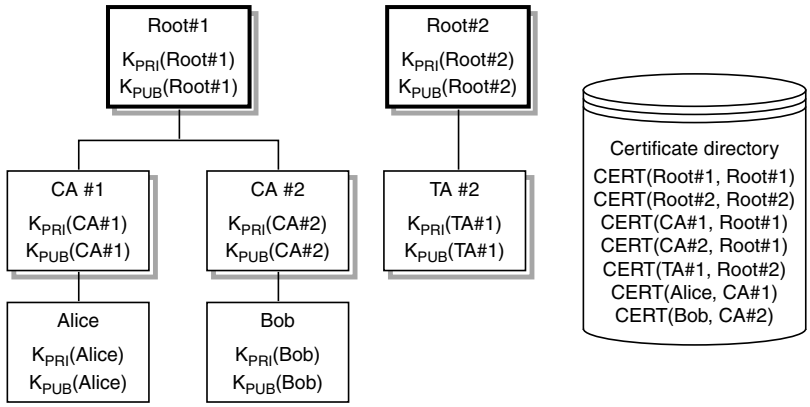

EXHIBIT 91.2 PKI with time authority.

Should Root#1 be compromised at some point thereafter, all of the CAs created prior to the compromise can still be trusted because access to Root#1 does not give the ability to create the CSDTs. Users can then be informed that anything signed by the Root after a specific date is not to be trusted, but anything signed before that date is still trustworthy.

91.4 Walk-Through of Issuance of a Certificate Containing a CSDT

The sequence of events to add a CSDT to a public key certificate is as follows:

1. User generates the public/private key pair.
2. User sends public key and user-specific information to the Registration Authority (RA).
3. RA validates user's request and forwards the certificate request to the CA.
4. CA forms the certificate and calculates the User Certificate Hash (UCH).
5. CA sends a digitally signed request to the Time Authority (TA) containing the UCH.
6. TA receives the request and validates the CA's signature on the request using the CA's public key certificate.
7. TA gets the current time from its secure time source.
8. TA calculates the sequential tie-breaker counter value.
9. TA forms the contents of the CSDT:
   a. UCH (Step 4)
   b. Timestamp (Step 7)
   c. Tie-breaker counter (Step 8)
   d. Hash of CA's certificate (same value used in Step 6)
10. TA calculates the hash of the contents of the CSDT.
11. TA encrypts hash with its private key.
12. TA returns CSDT to the CA.
13. CA validates the TA's signature on the CSDT using the TA's public key certificate.
14. CA verifies UCH in the CSDT against the UCH sent to the TA.
15. CA adds CSDT to the user certificate.
16. CA performs a standard signing process on the completed certificate.
17. CA sends digital certificate to the user.

91.5 Recovery Walk-Through

With any system providing assurance, it is necessary to have a plan of action in the event of some problem. The following outlines the minimum necessary steps if a CA is compromised.

Given:

- A CA signed by a CA Root
- A TA signed by a TA Root
- 10,000 users, each of which has generated a public/private key pair
- Each user has gone through the process of getting a public key certificate
- The CA root key is compromised by some form of attack

In infrastructures where CSDTs are not used, all 10,000 user certificates are immediately questionable and cannot be trusted for further transactions. A typical scenario requires the CA to have already created a second replacement root, and to have distributed the second root's self-signed public key certificate when the first was distributed. Users then are told to stop trusting the first root or to delete it from their applications. All users must then generate new key pairs and go through the enrollment process under the new root before business can return to normal.

This is obviously a scenario that requires considerable time and effort, and causes considerable inconvenience for users attempting to execute E-business transactions. Additionally, as the number of users increases, the recovery time increases linearly.

When CSDTs are employed and CSDT-aware applications are used, much of that effort is not required. Immediately upon determining that a compromise has occurred, the CA must:

- Inform the TA not to accept any further requests under the compromised key
- Inform its users
- Generate a new set of keys
- Issue no further certificates under the compromised key

Users need take no action other than to inform their applications of the date/time of the compromise of the CA. All future certificate validation is tested with the CA's certificate as well as the CSDT. If the CA's signature on a certificate is valid, but the CSDT is not present or indicates a date after the compromise, the certificate is rejected and the users are informed that they were presented with an invalid certificate.

91.6 Known Issues

Because events such as generating a hash, encryption, and decryption are processes of nonzero duration, it must be acknowledged that the actual time of certificate issuance is not the time within the CSDT. This is not a problem because the time within the CSDT, and within the certificate itself, are not to be used as an absolute time, but rather as a starting point from which the certificate is to be considered valid.

As with any cryptographic system, timely knowledge of any compromise of the system is a critical factor in limiting any "window of opportunity" for an attacker. In this case, it is up to the CA to inform its users that it has had a compromise. Information regarding a compromise of the TA must also be disseminated to users, but users need not take any direct action as a result.

One of the primary responsibilities of a CA is to ensure that everyone who wished to rely on its signature has access to its public key certificate. This is also true for the TA, which must use similar methods to establish trust in its public keys. This may cause some extra effort on the part the CA and its users.

91.7 Summary

What has been proposed and discussed in this chapter is a method of providing redundancy in a PKI where none has previously existed. Previous methods of breaking the Root private key into multiple parts created dual control over a single point of failure, but did nothing to provide any systemic redundancy.

It is worthwhile noting that this system is being prototyped by beTRUSTed, the trusted third-party service established by PricewaterhouseCoopers. Their testing, in cooperation with several PKI software vendors, may prove the usefulness and security of this system in a real-world environment.

As with any cryptographic system or protocol, the system of using CSDTs described herein must be analyzed and checked by numerous third parties for possible weaknesses or areas where an attacker may compromise the system.

Bibliography

1. Improving the Efficiency and Reliability of Digital Timestamping, http://www.surety.com/papers/BHSpaper.pdf.
2. How Do Digital Timestamps Support Digital Signatures?, http://x5.net/faqs/crypto/q108.html.
3. Digital Timestamping Overview, http://www.rsa.com/rsalabs/faq/html/7-11.html.
4. How to Digitally Timestamp a Document, http://www.surety.com/papers/1sttime-stampingpaper.pdf.
5. Answers to Frequently Asked Questions about Today's Cryptography, v3.0, Copyright 1996, RSA Data Security, Inc.