Far East J. Math. Sci. (FJMS) **19**(2) (2005), 133-143

KEY MANAGEMENT ASSOCIATED WITH SYMMETRIC AND ASYMMETRIC CRYPTOGRAPHY

S. VAJJIRAVELU, R. SATHYA

and

N. CH. S. N. IYENGAR

(Received June 4, 2005)

Submitted by K. K. Azad

Abstract

Cryptography is the art and science of encoding and decoding messages using mathematical algorithms that utilize a secret key. The concept has broadened to include managing messages that have some combination of: privacy (by being unreadable to anyone but the sender and receiver); integrity (not modified while en route); and non-repudiation (digitally signed in such a way that the originator cannot plausibly claim he or she did not originate it). Cryptographic keys are used to encrypt/decrypt data or to create/verify digital signatures. The secure distribution of these keys to the appropriate communicating parties is referred to as key distribution or key establishment. The life cycle associated with the initialization, distribution, and cancellation of the keys is referred to as key management. The purpose of this paper is to discuss key management, with particular emphasis on key distribution.

1. Introduction

Before we discuss key management, it is important to understand that there are two basic types of cryptography:

2000 Mathematics Subject Classification: 94A60.

Key words and phrases: key management, secret key cryptography, public key cryptography.

(1) *Symmetric* or *secret key*.

(2) *Asymmetric* [3] or *public key*.

Symmetric or *secret key cryptography* is characterized by the fact that the same key is used to perform both the encryption and decryption. The communicating parties must have copies of the same cryptographic key, and a method to securely convey these keys to the appropriate parties must be available. Compromise of the secret key naturally leads to the compromise of any data that was encrypted using that key.

Asymmetric or *Public key cryptography* is characterized by the fact that the key used to perform a cryptographic operation (e.g., digital signature creation) is not the key used to perform the inverse cryptographic operation (e.g., digital signature verification). Public key cryptography is based on the notion of *key pairs*. One key is referred to as the *public key* and can be revealed to anyone. The other key is referred to as the *private key* and is not revealed to anyone other than the end-entity associated with that key (although there are exceptions such as private key backup with a trusted third party when required). These keys are mathematically related; however, knowledge of the public key does not divulge enough information to allow an attacker to determine the private key efficiently.

We will first discuss key management associated with a secret key only. This will be followed by a discussion of public key cryptography and how public key and secret key cryptography can be used together.

2. Symmetric or Secret Key Cryptography

When the first electronic symmetric cryptosystems were deployed, key management was physical in nature and it required a significant amount of human involvement. Keys were centrally generated and recorded on media such as paper or magnetic tape and the keying material was physically distributed to the appropriate locations. This was sometimes accomplished through the use of couriers (sometimes humorously referred to as “sneaker net”). The keying material was physically destroyed when no longer needed. However, modern symmetric cryptosystems are more advanced and typically use some form

of electronic key distribution. One possible model for electronic distribution of symmetric keys is based on a trusted third party component known as a *Key Distribution Center* (KDC) [7]. Before an end-entity (e.g., an end user) can access a target resource (e.g., a server), the end-entity makes a request to the KDC to establish a session key that can be used to secure the communication between the end-entity and the target resource. This model is illustrated in Figure 1.

The outbound arrows between the KDC and the communicating parties are logical representations of the key distribution process. In practice, the KDC may distribute one copy of the symmetric key directly to the end-entity and another copy to the target resource, or both copies of the symmetric key may be distributed back to the end-entity and the end-entity would then pass the symmetric key to the target resource. In both cases two copies are needed since the symmetric key is encrypted for the intended recipients (i.e., one copy of the key is encrypted for the end-entity and another copy of the key is encrypted for the target resource).

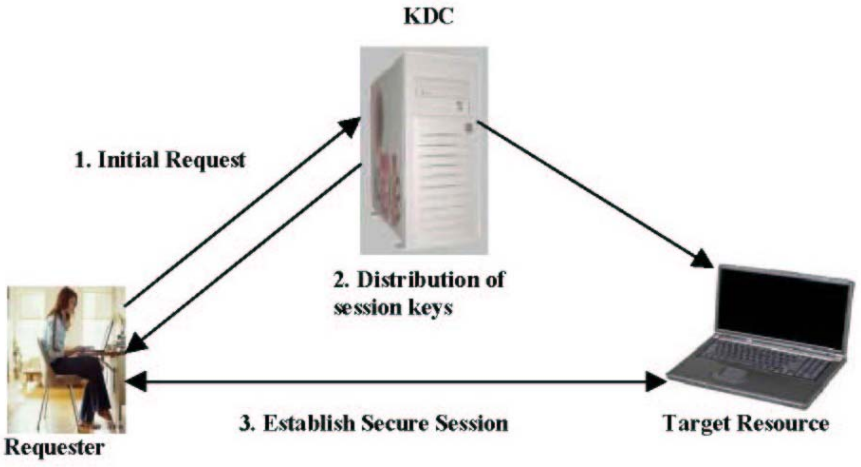

Figure 1. Key distribution center model.

This is necessary to prevent someone in a position to intercept the session key from being able to use the key to eavesdrop on the subsequent communication. This implies that the KDC and the communicating parties must have been pre-initialized with keys that can be used to protect the distribution of the session keys. These are sometimes referred to as *Key Encrypting Keys* (KEKs). Note that this pre-initialization step

should not be considered unusual since some form of initial bootstrap process is typically required in any cryptographic system. A classic example of an electronic secret key distribution based on this model is *Kerberos* V5 [1].

Kerberos enables electronic key distribution in a client/server network environment. Kerberos is comprised primarily of two logical components:

(1) *The Authentication Server* (AS).

(2) *The Ticket Granting Server* (TGS).

The AS and TGS may be physically separate, or they may reside on the same platform, or they may even be part of the same process. Collectively, these two logical components can be thought of as the KDC as described above. Since Kerberos is a well known, publicly available symmetric cryptosystem, we will use Kerberos to illustrate the concepts associated with symmetric key management.

2.1. Initialization

In the Kerberos scheme, end-entities and target resources are referred to as principals. Kerberos maintains a database of all principals and their associated symmetric keys. This allows the session keys for each principal to be protected when they are in transit. These symmetric keys are initialized separately and must be established before an end-entity can send a request to the AS.

2.2. Distribution

When an end-entity needs to communicate with a target resource for the first time, the end-entity makes a request to the AS. The request contains the end-entity's identifier as well as the identifier of the target resource. Typically the initial target resource is the TGS and for the purposes of this example, we will assume that the request is for a session with the TGS. However, this may not always the case [6].

Assuming that the end-entity and target resource (i.e., TGS) are in the Kerberos database, the AS will generate a symmetric key (referred to as a *session key*) and encrypt one copy of the session key using the symmetric key of the end-entity and another copy of the session key using

the symmetric key of the target resource (this is referred to as a *ticket*). These encrypted copies of the session key are returned to the requesting end-entity. The end-entity decrypts its copy of the session key and uses it to encrypt the end-entity's identity information and a time stamp (this is referred to as the *authenticator*). The time stamp is necessary to prevent replay attacks. The session key encrypted by the AS for the target resource (i.e., the ticket for the TGS) and the encrypted authenticator are then sent to the target resource. The two communicating parties now have copies of the session key and are able to communicate securely.

From that point forward the end-entity can make additional requests to either the AS or the TGS in order to establish a session key between the end-entity and any other target resource. The difference between requesting this information from the AS and from the TGS is that the end-entity's copy of the session key is encrypted with the end-entity's symmetric key obtained from the Kerberos database when dealing with the AS, but the end-entity's copy of the session key is encrypted using the TGS session key when dealing with the TGS.

2.3. Cancellation

In terms of key cancellation, we need to consider that there are actually two types of keys being used.

(i) Each principal (i.e., end user or server) has a shared secret key used to protect the distribution of the session keys. The lifetime of these shared secret keys is typically very long. Cancellation of a given key is usually facilitated through replacement (i.e., keys can be changed in accordance with local policy) or deletion of an entry (e.g., when a principal no longer belongs within the Kerberos realm).

(ii) The second type of key is the session key. The lifetime of the session keys is directly coupled with the lifetime of the session itself. Once the session between the client and server is terminated, the session key is no longer used.

3. Asymmetric or Public Key Cryptography

Asymmetric cryptography is often implemented in association with a supporting infrastructure referred to as *Public Key Infrastructure* (PKI).

PKI refers to the policies, procedures, personnel, and components necessary to support the key management process. The primary components of the PKI include the *Certification Authority* (CA) and *Local Registration Authority* (LRA). The consumers of the PKI-related services are referred to as *end-entities* and may be end users, devices, processes, or servers.

3.1. Initialization

The generation of the public/private key pairs associated with the end-entities can occur within the CA, LRA, or the end-entity's system. If necessary (depending on where key generation occurred), the private component of the public/private key pair must be securely distributed to the appropriate end-entity. Several protocols have been defined to accomplish this [8], [1]. The private key is stored securely in standard formats such as PKCS #5 and #8. The public key component is populated in a signed data structure issued by a CA. This data structure is referred to as a *public key certificate*. The latest version of the public key certificate (version 3) is defined in [5] and a high-level representation of a version 3 public key certificate is provided in Figure 2.

The *digital signature* appended to the public key certificate provides two things.

(i) The integrity of the certificate can be verified so any modifications to the data contained within the certificate after it was issued can be detected.

(ii) The identity of the issuing CA can be verified.

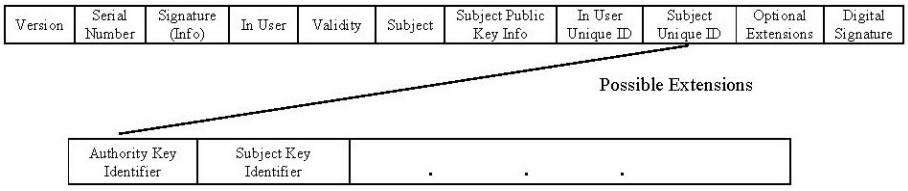

Figure 2. Public key certificate (version 3).

This allows the users of the certificate to determine if the certificate originated from a trustworthy source. Since both the content and source of the certificate can be verified, the certificate can be distributed via potentially non-secure channels. For example, the public key certificate can be stored "in the clear" in a public repository (e.g., an X.500 directory) which allows end-entities to easily retrieve these certificates when required. When end-entities enroll with the PKI, they typically use one or more shared secrets to demonstrate they are the end-entity that they claim to be. The shared secrets may have been established at some point in the past, or they may be distributed to the end-entity as part of a formal registration process. This latter method is typically required when the certificate(s) will be used in conjunction with high assurance and/or sensitive transactions. This often requires that the end-entity presents himself or herself to an LRA along with acceptable forms of identification (e.g., a driver's license or employee ID). In any case, the shared secrets facilitate the initial bootstrap process in which end-entities are first initialized with their keys.

3.2. Distribution

Once the end-entities are initialized with the PKI, they can engage in secure communication (e.g., secure e-mail) with their peers. Theoretically, it would be possible for end-entities to use public key cryptography to encrypt data for their peers by using the public key of each peer (assuming that the asymmetric algorithm supports encryption/decryption). However, there are a few practical issues that make this an unattractive approach.

(i) Asymmetric cryptography is slow when compared to symmetric cryptography (therefore suitable only for the encryption of small amounts of data).

(ii) It would be extremely wasteful to encrypt the data *N* times, once for each intended recipient – especially when dealing with large amounts of data (consider an e-mail with file attachments). This is also true even when symmetric algorithms are used.

(iii) Not all asymmetric algorithms support encryption/decryption (e.g., RSA does, but DSA does not).

(iv) PKI supports both asymmetric and symmetric cryptography by taking the advantage of the speed of symmetric cryptography, avoiding the key distribution problems and encryption of the data multiple times.

Symmetric cryptography is used for data encryption (but the data is only encrypted once regardless of the number of recipients), and asymmetric cryptography is used for the distribution of the secret key to the intended recipients. Figure 3 illustrates how this works with an example.

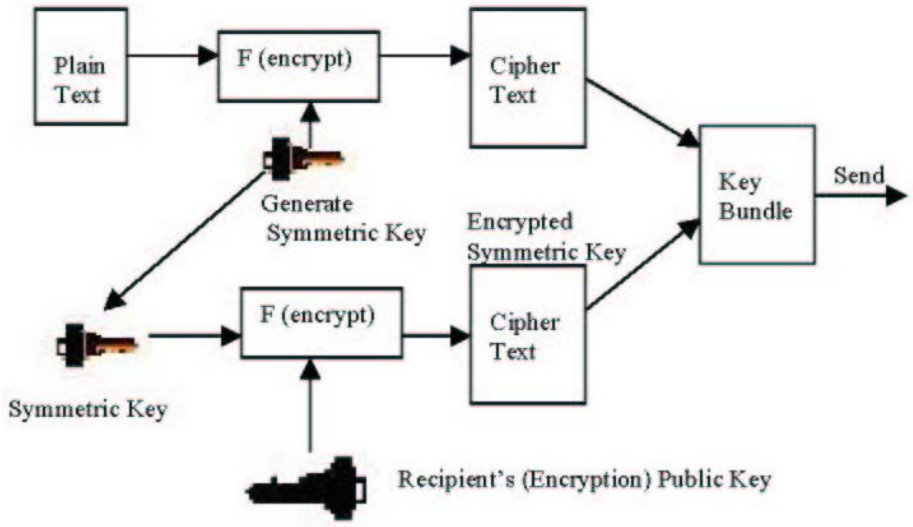

Figure 3. Asymmetric key distribution.

In this example, we are using a symmetric algorithm to encrypt data (e.g., an e-mail message) and an asymmetric algorithm such as RSA to enable the secure distribution of the secret key that was used to encrypt the e-mail. Essentially, the system generates a secret key that is then used to encrypt the message. Any number of symmetric algorithms could be used for this purpose (e.g., CAST-128 or Rijndael). The secret key is then encrypted using the intended recipient's public (encryption) key. If multiple recipients were involved, the original data would still be encrypted once using the generated secret key, and the secret key would be encrypted *N* times, once for each recipient. The public key certificate for each recipient can be retrieved from a repository, or perhaps the certificate may have been conveyed to the originator in a previous

exchange. On receipt, each recipient can use his/her corresponding private decryption key to decrypt the symmetric key necessary to decrypt the original data. This provides an efficient and secure key distribution mechanism that does not suffer from the drawbacks discussed in the previous section.

Asymmetric cryptography can also be used to support a process known as key agreement. In this case, the communicating parties negotiate an ephemeral secret key. (See *Diffie-Hellman key exchange protocol*.)

3.3. Cancellation

In terms of key cancellation, there are two things to consider:

(1) Secret key used to encrypt the data.

(2) Public/private key pair.

In terms of the secret key, this survives as long as the data is encrypted, which could be indefinitely. However, the secret key is deleted/destroyed when the associated file is deleted or (permanently) decrypted. There may also be cases when the protection is no longer considered adequate (e.g., the symmetric algorithm has been compromised or the key length used no longer provides adequate protection). In this event, the file is decrypted and re-encrypted using a new algorithm and/or key. The original key would be deleted/destroyed.

In terms of the public/private key pairs, public key certificates are issued with a fixed lifetime, typically on the order of 2-5 years depending on the purpose of the certificate and the associated local policy. In some PKIs, the certificates (and associated private key) are renewed automatically before the existing certificate(s) expire. In other PKIs, end-entities must request new certificate(s) when their existing certificate(s) expire. It is possible to establish a different lifetime for the private component of the public/private key pair when that key pair is used in conjunction with digital signatures (see *digital signature schemes*). This is done in comprehensive PKIs where it is desirable to have a grace period between the time the private signing key can no longer be used and the time that the associated public key certificate expires so that digital

signatures created before the corresponding private key expired can still be verified without exposing the end user to needless warning messages. These comprehensive PKIs generally update the public/private key pairs (and public key certificate) automatically. Finally, it is possible to revoke certificates before they naturally expire. This might be done for a variety of reasons, including suspected private key compromise. (See *certificate revocation* for additional information related to certificate revocation [2].)

4. Summary

(i) The Kerberos database is initialized with entries for each principal. Each entry includes the shared secret key associated with that principal which is used to protect the session keys. The session keys are generated by the KDC in response to requests from clients and are securely distributed to the appropriate principals. The shared secret keys generally have long lifetimes, but the session keys are ephemeral (i.e., short-lived). One of the criticisms associated with symmetric only key management is that it does not scale well. It has also been criticized for having a single point of failure (i.e., what happens when the KDC goes down?) as well as a single point of attack (all of the pre-initialized symmetric keys are stored in the KDC database). However, alternative key management schemes exist that help to alleviate some of these problems. In particular, asymmetric cryptography can be used to exchange secret keys as discussed in the next section.

(ii) PKI provides comprehensive key management through a combination of asymmetric and symmetric cryptography. Symmetric cryptography is used for bulk data encryption/decryption and asymmetric cryptography is used for key distribution. The use of asymmetric cryptography to facilitate key distribution is an extremely powerful tool that serves to eliminate many of the problems associated with symmetric-only cryptosystems.

References

[1] C. Adams and S. Farrell, Internet X.509 Public Key Infrastructure: Certificate Management Protocols, Internet Request for Comments 2510, March 1999.

[2] C. Adams and S. Lloyd, Understanding PKI: Concepts, Standards and Deployment Considerations, 2nd ed., Addison-Wesley, 2003.

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

[4] Pres Herrington, Tim Draelos and Rick Craft, A Key Management Concept for the CTBT International Monitoring System.
http://www.cscap.nuctrans.org/Nuc_Trans/links/ctbt-paper.html

[5] ITU-T Recommendation X.509, Information Technology - Open Systems Interconnection - The Directory: Public Key and Attribute Certificate Frameworks, March 2000 (Equivalent to ISO/IEC 9594-8:2001).

[6] J. Kohl and C. Neuman, The Kerberos Network Authentication Service (V5), Internet Request for Comments 1510, September 1993.

[7] R. Needham and M. Schroeder, Using encryption for authentication in large networks of computers, Communications of the ACM 21(12) (1978), 993-999.

[8] B. Ramsdell, S/MIME Version 3 Certificate Handling, Internet Request for Comments 2632, June 1999.

[9] William Stallings, Cryptography and Network Security, 3rd ed., Prentice-Hall, 2003.

School of Computer Sciences
Vellore Institute of Technology
Deemed University
Vellore-632014, Tamil Nadu, India