Asymmetric Encryption

Session 3

LEARNING OBJECTIVES

On completion of this session you should:

- Understand the principles of public key encryption
- Be familiar with public key encryption algorithms
- Understand how public key encryption can be used in exchanging key in private key encryption
- Appreciate the use of hash function to authenticate messages
- Understand how public key encryption can be used to produce digital signature

Contents

Reading

Prescribed readings

Reading 1: Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 76 to 87.

Skim: W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 87 to 93.

Reading 2: Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 93-96, specially Fig. 3.9.

Reading 3: http://csrc.nist.gov/CryptoToolkit/aes/rijndael/ http://csrc.nist.gov/encryption/aes

Reading 4: Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 97 to 104.

Reading 5: Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 104.

3.0 Introduction

One important issue in a secure communication between two parties is to guarantee the authentication of the message, i.e., protection of data against alteration in transit. Any communication is said to be authentic when it is not altered and comes from its real source. Two essential aspects that we want to ensure are: 1) the content is unaltered and 2) the source is genuine. We may also wish to verify a message's timeliness. In this session, we will discuss the techniques that can be used to guarantee authentication of a message.

As mentioned in the previous study guide, one type of encryption technique is asymmetric encryption, also called public key encryption. In this study guide, we will present the basic concepts behind this encryption technique, various algorithms to implement this technique, and how it can be used to produce digital certificates.

3.1 Message Authentication

Message authentication can be achieved by using private key encryption. If only sender and receiver, nobody else, share the same key, then only they will be able to encrypt and decrypt the message. An error detection code and sequence numbering will ensure non-alteration of data. A timestamp can tell whether the message has been delayed more than what is normally expected. Two important techniques dealing message authentication are discussed below.

Message Authentication Code (MAC):

A small block of data, called message authentication code (MAC), is generated, appended and transmitted with the message. MAC is a function of the message as well as the secret key. MAC ensures non-alteration of data in the sense that, if a single bit of the massage is changed the calculated MAC code will also change. The transmitted message is: message + MAC code. The receiver at the receiving end recalculates the code using the message it has received. If newly generated MAC code matches the received MAC code, we can be assured of the followings [1, p.61]

- The message has not been altered. The attacker can not alter both the message and the code to make a match, because alternation of code needs the key which the attacker does not have.
- Because nobody else has the key, the receiver can be assured of the originator of the message.
- If the message includes sequence number (X.25, HDLC, TCP), proper sequence is maintained.

Hash function:

Hash function is used in message authentication as well as in digital signature. Like MAC, hash function is a one way function that accepts a message *M* and produces a fixed-size message digest H(*M*). Unlike MAC, hash function takes only the message (not the key) as input to generate the message digest. Hash function can be used with both symmetric and

asymmetric encryption. A hash function acts as a fingerprint of the message and can be called secured if it possesses the following properties [1, p. 65]

- It can be applied to a block of data of arbitrary size.
- Produces a fixed length message digest.
- For a given hash value h, it is computationally infeasible to find a message x such that $H(x)=h$, ie., given the message digest, it is infeasible to find the message itself. This indicates that hash function does not have an inverse function.
- For any given block x, it is computationally infeasible to find $y \neq x$ with $H(x)=H(y)$, i.e., no two different messages can generate the same message digest.
- It is computationally infeasible to find any pair (x,y) such that $H(x)=H(y)$.

An n-bit hash function is generated by operation on a sequence of n-bit block. The input is processed on one block at a time in an iterative fashion. One simple hash function is the bit-by-bit exclusive OR on every block. The Secure Hash Algorithm (SHA) was first developed by the National Institute of Standard & Technology (NIST) in 1993 and a revised version, known as SHA-1, in 1995. The SHA-1 algorithm takes a maximum length of 2^64 bits as input, processes the input in 512-bit blocks and produces 160-bit message digest. MD5 developed by Ron Rivest takes input message of arbitrary length, processing is done in 512-bit blocks and produces 128-bit message digest.

Algorithms such as SHA-1, MD5 were not designed for use as MAC, because it does not rely on a secret key. HMAC was developed by incorporating a secret key into the existing hash algorithm. HMAC has been chosen to implement MAC for IP security, and used in other protocols, e.g., TLS (Transport Layer Security) and SET (Secure Electronic Transaction) [1, pp. 71].

Reading 1:
Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 76 to 87.

Skim:
W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 87 to 93.

3.2 Asymmetric (Public key) Encryption

Asymmetric encryption is a more recent development than symmetric key encryption. Asymmetric encryption involves a pair of keys (called public and private keys) while symmetric encryption uses only one key (private key, discussed in previous session). The key pair (k1 and k2) is related such that the data encrypted by k1 can only be decrypted by its corresponding pair k2. Any of the keys can be used for encryption and the other for decryption. One of the keys is kept secret by the owner (hence called private key) and the other is published (called public key). One important property of asymmetric encryption is that, even if we have one key, we can't compute the other. An asymmetric encryption has the following six components [1, p.74].

- Plaintext: The data that is fed as input to the algorithm.
- Encryption algorithm: The algorithm that transforms the plaintext.
- Public and private keys: A pair of related keys used for encryption and decryption process.
- Ciphertext: The scrambled message produced as output.
- Decryption algorithm: The algorithm takes the ciphertext and the matching key as inputs and produces the original plaintext.

Asymmetric encryption provides confidentiality, authentication and data integrity. It is used in digital signature and exchanging keys. Asymmetric encryption is computationally intensive and much slower than symmetric encryption.

Asymmetric encryption can be used in two modes:

Encryption mode:

1. Each user (say, Bob and Alice) generates a pair of keys, distributes his/her own public key, but keeps the private key secret.

2. Bob wants to send an encrypted message to Alice. He maintains the public keys of many people. He encrypts the message with Alice's advertised public key and sends the encrypted message over a communication channel which may be insecure.

3. Even if someone intercepts the message, he can't decrypt it because he does not have the matching key (i.e., private key).

4. When the message reaches Alice, *only* she can decrypt it by using her private key.

Authentication mode:

1. Each user generates a pair of keys, distributes his/her own public key, but keeps the private key secret.

2. Bob wants to send an encrypted message to Alice and authenticate himself so that Alice is sure about the originator of the message. He encrypts the message with his own private key and sends it over a communication channel which may be insecure.

3. When the message reaches Alice, she can decrypt it by using only Bob's public key (Alice has public key of many people).

4. Since the message can be decrypted using only Bob's public key (no other public key can decrypt the message), Alice is assured that Bob is the owner/sender of the message.

Reading 2:
Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 93-96, specially Fig. 3.9.

Reading 3:
http://csrc.nist.gov/CryptoToolkit/aes/rijndael/
http://csrc.nist.gov/encryption/aes

3.3 Encryption Algorithms

The two most widely used public key encryption algorithms are RSA and Diffie- Hellman. Basic steps in these two algorithms are as follows.

RSA Algorithm

In 1978, Ron Rivest, Adi Shamir and Len Adlemean at MIT proposed the RSA (Rivest-Shamir-Adleman) encryption algorithm. This algorithm is based on the difficulty of factoring large numbers. RSA is a block cipher in which plaintext and ciphertext are integers between 0 to n -1 for some n. The key generation process is as follows.

1. Select large prime numbers p and q, and calculate $n = pq$.
2. Select an integer $e > 1$ such that GCD(e, $(p - 1)(q - 1)$) = 1. GCD is Greatest Common Divisor.
3. Solve the congruence, $ed \equiv 1 \bmod ((p - 1), (q - 1))$

   for an integer d where $1 < d < (p - 1)(q - 1)$.
4. The public encryption key is (e,n).
5. The private encryption key is (d,n).

Denoting plaintext block as M and ciphertext block as C, the basic algorithm in encryption mode has the following steps:

$C = M^e \bmod n$

$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

And in authentication mode:

$C = M^d \bmod n$

$M = C^e \bmod n$

For RSA algorithm to work satisfactorily, the following requirement must be made [1, pp. 78].

1. It is possible to find values of e, d, n such that $M^{ed} = M \bmod n$ for all $M < n$.
2. It is relatively easy to calculate M^e and C^d for all values of $M < n$.
3. It is infeasible to determine d given e and n. This requirement is met when e and n are large values.

Diffie-Hellman Key Exchange

In 1976, Whitfield Diffie and Maritn Hellman proposed this public key encryption. This scheme was developed to address the issues of key exchange. The algorithm is limited to the exchange of keys. The algorithm has the following steps:

1. **A** and **B** agree on two large integers a and b such that $1<a<b$.
2. **A** selects a random number i and computes $I=a^i \bmod b$. **A** sends I to **B**.
3. **B** selects a random number j and computes $J=a^j \bmod b$. **B** sends J to **A**.
4. **A** computes $k1=J^i \bmod b$.
5. **B** computes $k2=I^j \bmod b$.

Someone may eavesdrop a,b,I,J but i and j remain secret. It is computationally infeasible to find i given I ($=a^i \bmod b$), or j given J ($=a^j \bmod b$) when the numbers a and b are very large. This problem is known as discrete algorithm problem and the effectiveness of this algorithm depends upon difficulty of this problem.

The other public key encryption algorithms are digital Signature Standard (DSS) and elliptic curve cryptography.

Reading 4:
Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 97 to 104.

3.4 Digital Certificate

A digital certificate is a data item is attached to or logically associated with the message that can be used to ascertain the originator of the message and integrity of the data. If Bob wants to digitally sign a document he appends a signed authenticator along with the message. An authenticator is a small number of bits that is a function of the message, and Bob signs it by using his private key. Alice uses Bob's public key to find a match with the received signed authenticator. Only Bob's public key will find a match

ensuring that Bob is the originator of the message. If the message is changed in transit, no match will be found. A secure hash code like SHA-1 can be used as an authenticator.

Digital certificate is similar to MAC, but has a different purpose. In case of dispute, a digital certificate may need to support non-repudiation, i.e., the originator of the message can't deny sending the message. Hence, the recipient must not be able to generate an exact signed authenticator created by the sender. A MAC can not function as digital signature since the recipient knows the key used to generate the MAC [2, p.110].

3.5 Key Management

Public key encryption plays a major role to address the problem of key distribution. It serves: 1) to distribute the public keys, and 2) to exchange the secret key (like session key or private key in symmetric encryption) . The distribution of public key is built on a trust model. In public key certificates, it has a block of data containing a public key and a user ID of the key owner, and the whole block is signed by a trusted third party. Typically the third party is a Certification Authority (CA) trusted by the community. X.509 is such a public key certificate standard.

A secret key like the session key can be exchanged using public key. Communication is faster if the same key is used for encryption and decryption by the sender and receiver. A session key is a secret key that is used by the parties only for a particular session. But at first the parties must exchange the session key. Say, Bob wants to communicate with Alice using a session key. He generates a session key, encrypt it with Alice's public key and send it to Alice. Only Alice can decrypt it by using her private key. Once Alice knows the session key, rest of the communication can be made using symmetric encryption. An example of session key is an SSL enabled web session. Every time we engage in a web session, a new session key is generated.

Reading 5:
Read W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010, pp. 104.

3.6 Conclusion

One problem of symmetric encryption is how to distribute the secret key. Asymmetric encryption uses a key-pair and can be used to distribute the secret session key. It can provide confidentiality, data integrity and authentication. However, asymmetric encryption is computationally intensive and much slower than symmetric encryption.

3.7 References

[1] W. Stallings, Network Security Essentials - Application and Standard, 4th edition, Prentice Hall, 2010.

[2] W. Ford and M. Baum, Secure Electronic Commerce, Prentice Hall, 2000.