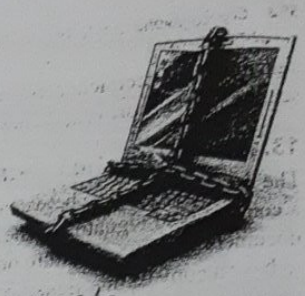
INFORMATION SECURITY

Neha

MODULE-5

13



Digital Signature

Alice → Bob

Objectives

This chapter has several objectives:

- To define a digital signature
- To define security services provided by a digital signature
- To define attacks on digital signatures
- To discuss some digital signature schemes, including RSA, ElGamal, Schnorr, DSS, and elliptic curve
- To describe some applications of digital signatures

We are all familiar with the concept of a signature. A person signs a document to show that it originated from her or was approved by her. The signature is proof to the recipient that the document comes from the correct entity. When a customer signs a check, the bank needs to be sure that the check is issued by that customer and nobody else. In other words, a signature on a document, when verified, is a sign of authentication—the document is authentic. Consider a painting signed by an artist. The signature on the art, if authentic, means that the painting is probably authentic.

When Alice sends a message to Bob, Bob needs to check the authenticity of the sender; he needs to be sure that the message comes from Alice and not Eve. Bob can ask Alice to sign the message electronically. In other words, an electronic signature can prove the authenticity of Alice as the sender of the message. We refer to this type of signature as a **digital signature.**

In this chapter, we first introduce some issues related to digital signatures and then we walk through different digital signature schemes.

13.1 COMPARISON

Let us begin by looking at the differences between conventional signatures and digital signatures.

13.1.1 Inclusion

A conventional signature is included in the document; it is part of the document. When we write a check, the signature is on the check; it is not a separate document. But when we sign a document digitally, we send the signature as a separate document. The sender sends two documents: the message and the

signature. The recipient receives both documents and verifies that the signature belongs to the supposed sender. If this is proven, the message is kept; otherwise, it is rejected.

13.1.2 Verification Method

The second difference between the two types of signatures is the method of verifying the signature. For a conventional signature, when the recipient receives a document, she compares the signature on the document with the signature on file. If they are the same, the document is authentic. The recipient needs to have a copy of this signature on file for comparison. For a digital signature, the recipient receives the message and the signature. A copy of the signature is not stored anywhere. The recipient needs to apply a verification technique to the combination of the message and the signature to verify the authenticity.

13.1.3 Relationship

For a conventional signature, there is normally a one-to-many relationship between a signature and documents. A person uses the same signature to sign many documents. For a digital signature, there is a one-to-one relationship between a signature and a message. Each message has its own signature. The signature of one message cannot be used in another message. If Bob receives two messages, one after another, from Alice, he cannot use the signature of the first message to verify the second. Each message needs a new signature.

13.1.4 Duplicity

Another difference between the two types of signatures is a quality called *duplicity*. In conventional signature, a copy of the signed document can be distinguished from the original one on file. In digital signature, there is no such distinction unless there is a factor of time (such as a timestamp) on the document. For example, suppose Alice sends a document instructing Bob to pay Eve. If Eve intercepts the document and the signature, she can replay it later to get money again from Bob.

13.2 PROCESS

Figure 13.1 shows the digital signature process. The sender uses a **signing algorithm** to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the **verifying algorithm** to the combination. If the result is true, the message is accepted; otherwise, it is rejected.

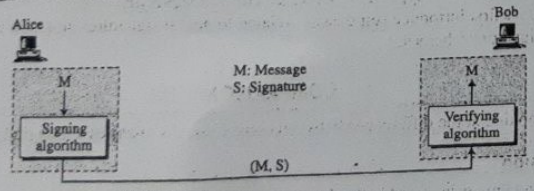


Fig. 13.1 *Digital signature process*

13.2.1 Need for Keys

A conventional signature is like a private "key" belonging to the signer of the document. The signer uses it to sign documents; no one else has this signature. The copy of the signature is on file like a public key; anyone can use it to verify a document, to compare it to the original signature.

In a digital signature, the signer uses her private key, applied to a signing algorithm, to sign the document. The verifier, on the other hand, uses the public key of the signer, applied to the verifying algorithm, to verify the document.

We can add the private and public keys to Fig. 13.1 to give a more complete concept of digital signature (see Fig. 13.2). Note that when a document is signed, anyone, including Bob, can verify it because everyone has access to Alice's public key. Alice must not use her public key to sign the document because then anyone could forge her signature.

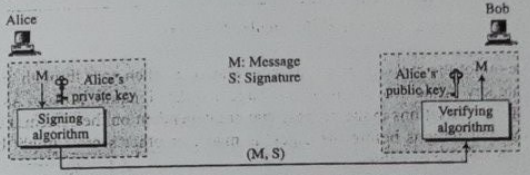


Fig. 13.2 *Adding key to the digital signature process*

Can we use a secret (symmetric) key to both sign and verify a signature? The answer is negative for several reasons. First, a secret key is known by only two entities (Alice and Bob, for example). So if Alice needs to sign another document and send it to Ted, she needs to use another secret key. Second, as we will see, creating a secret key for a session involves authentication, which uses a digital signature. We have a vicious cycle. Third, Bob could use the secret key between himself and Alice, sign a document, send it to Ted, and pretend that it came from Alice.

A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer's public key.

We should make a distinction between private and public keys as used in digital signatures and public and private keys as used in a cryptosystem for confidentiality. In the latter, the private and public keys of the receiver are used in the process. The sender uses the public key of the receiver to encrypt; the receiver uses his own private key to decrypt. In a digital signature, the private and public keys of the sender are used. The sender uses her private key; the receiver uses the sender's public key.

A cryptosystem uses the private and public keys of the receiver: a digital signature uses the private and public keys of the sender.

13.2.2 Signing the Digest

In Chapter 10, we learned that the asymmetric-key cryptosystems are very inefficient when dealing with long messages. In a digital signature system, the messages are normally long, but we have to use asymmetric-key schemes. The solution is to sign a digest of the message, which is much shorter than the

message. As we learned in Chapter 11, a carefully selected message digest has a one-to-one relationship with the message. The sender can sign the message digest and the receiver can verify the message digest. The effect is the same. Figure 13.3 shows signing a digest in a digital signature system.

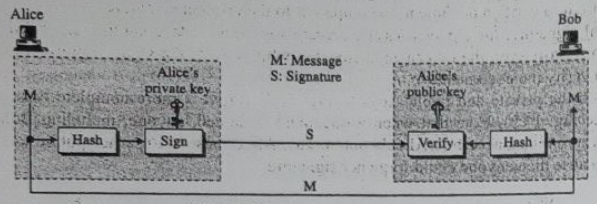


Fig. 13.3 *Signing the digest*

A digest is made out of the message at Alice's site. The digest then goes through the signing process using Alice's private key. Alice then sends the message and the signature to Bob. As we will see later in this chapter, there are variations in the process that are dependent on the system. For example, there might be additional calculations before the digest is made, or other secrets might be used. In some systems, the signature is a set of values.

At Bob's site, using the same public hash function, a digest is first created out of the received message. Calculations are done on the signature and the digest. The verifying process also applies criteria on the result of the calculation to determine the authenticity of the signature. If authentic, the message is accepted; otherwise, it is rejected.

13.3 SERVICES

We discussed several security services in Chapter 1 including *message confidentiality, message authentication, message integrity*, and *nonrepudiation*. A digital signature can directly provide the last three; for message confidentiality we still need encryption/decryption.

13.3.1 Message Authentication

A secure digital signature scheme, like a secure conventional signature (one that cannot be easily copied) can provide message authentication (also referred to as data-origin authentication). Bob can verify that the message is sent by Alice because Alice's public key is used in verification. Alice's public key cannot verify the signature signed by Eve's private key.

A digital signature provides message authentication.

13.3.2 Message Integrity

The integrity of the message is preserved even if we sign the whole message because we cannot get the same signature if the message is changed. The digital signature schemes today use a hash function in the signing and verifying algorithms that preserve the integrity of the message.

A digital signature provides message integrity.

13.3.3 Nonrepudiation

If Alice signs a message and then denies it, can Bob later prove that Alice actually signed it? For example, if Alice sends a message to a bank (Bob) and asks to transfer $10,000 from her account to Ted's account, can Alice later deny that she sent this message? With the scheme we have presented so far, Bob might have a problem. Bob must keep the signature on file and later use Alice's public key to create the original message to prove the message in the file and the newly created message are the same. This is not feasible because Alice may have changed her private or public key during this time; she may also claim that the file containing the signature is not authentic.

One solution is a trusted third party. People can create an established trusted party among themselves. In future chapters, we will see that a trusted party can solve many other problems concerning security services and key exchange. Figure 13.4 shows how a trusted party can prevent Alice from denying that she sent the message.

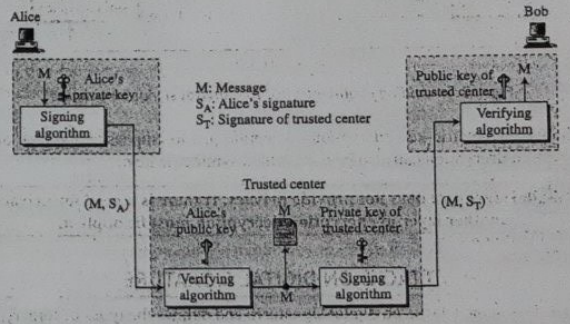


Fig. 13.4 *Using a trusted center for nonrepudiation*

Alice creates a signature from her message (S_A) and sends the message, her identity, Bob's identity, and the signature to the center. The center, after checking that Alice's public key is valid, verifies through Alice's public key that the message came from Alice. The center then saves a copy of the message with the sender identity, recipient identity, and a timestamp in its archive. The center uses its private key to create another signature (S_T) from the message. The center then sends the message, the new signature, Alice's identity, and Bob's identity to Bob. Bob verifies the message using the public key of the trusted center.

If in the future Alice denies that she sent the message, the center can show a copy of the saved message. If Bob's message is a duplicate of the message saved at the center, Alice will lose the dispute. To make everything confidential, a level of encryption/decryption can be added to the scheme, as discussed in the next section.

Nonrepudiation can be provided using a trusted party.

13.3.4 Confidentiality

A digital signature does not provide confidential communication. If confidentiality is required, the message and the signature must be encrypted using either a secret-key or public-key cryptosystem. Figure 13.5 shows how this extra level can be added to a simple digital signature scheme.

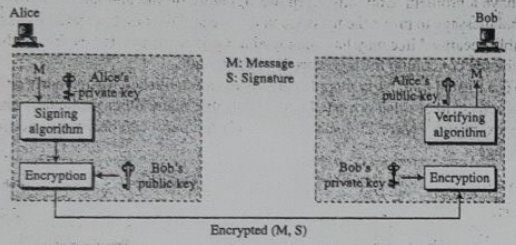


Fig. 13.5 *Adding confidentiality to a digital signature scheme*

We have shown asymmetric-key encryption/decryption just to emphasize the type of keys used at each end. Encryption/decryption can also be done with a symmetric key.

A digital signature does not provide privacy. If there is a need for privacy, another layer of encryption/decryption must be applied.

13.4 ATTACKS ON DIGITAL SIGNATURE

This section describes some attacks on digital signatures and defines the types of forgery.

13.4.1 Attack Types

We will look on three kinds of attacks on digital signatures: key-only, known-message, and chosen-message.

Key-Only Attack In the **key-only attack,** Eve has access only to the public information released by Alice. To forge a message, Eve needs to create Alice's signature to convince Bob that the message is coming from Alice. This is the same as the ciphertext-only attack we discussed for encipherment.

Known-Message Attack In the **known-message attack,** Eve has access to one or more message-signature pairs. In other words, she has access to some documents previously signed by Alice. Eve tries to create another message and forge Alice's signature on it. This is similar to the known-plaintext attack we discussed for encipherment.

Chosen-Message Attack In the **chosen-message attack,** Eve somehow makes Alice sign one or more messages for her. Eve now has a chosen-message/signature pair. Eve later creates another message, with the content she wants, and forges Alice's signature on it. This is similar to the chosen-plaintext attack we discussed for encipherment.

13.4.2 Forgery Types

If the attack is successful, the result is a forgery. We can have two types of forgery: existential and selective.

Existential Forgery In an **existential forgery,** Eve may be able to create a valid message-signature pair, but not one that she can really use. In other words, a document has been forged, but the content is randomly calculated. This type of forgery is probable, but fortunately Eve cannot benefit from it very much. Her message could be syntactically or semantically unintelligible.

Selective Forgery In **selective forgery,** Eve may be able to forge Alice's signature on a message with the content selectively chosen by Eve. Although this is beneficial to Eve, and may be very detrimental to Alice, the probability of such forgery is low, but not negligible.

13.5 DIGITAL SIGNATURE SCHEMES

Several **digital signature schemes** have evolved during the last few decades. Some of them have been implemented. In this section, we discuss these schemes. In the following section we discuss one that will probably become the standard.

13.5.1 RSA Digital Signature Scheme

In Chapter 10 we discussed how to use RSA cryptosystem to provide privacy. The RSA idea can also be used for signing and verifying a message. In this case, it is called the **RSA digital signature scheme.** The digital signature scheme changes the roles of the private and public keys. First, the private and public keys of the sender, not the receiver, are used. Second, the sender uses her own private key to sign the document; the receiver uses the sender's public key to verify it. If we compare the scheme with the conventional way of signing, we see that the private key plays the role of the sender's own signature; the sender's public key plays the role of the copy of the signature that is available to the public. Obviously Alice cannot use Bob's public key to sign the message because then any other person could do the same. Figure 13.6 gives the general idea behind the RSA digital signature scheme.

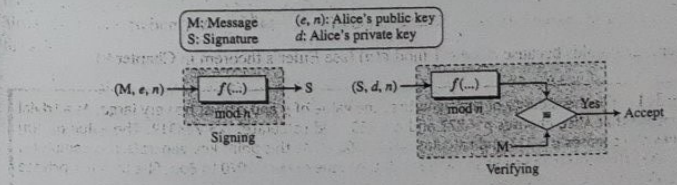


Fig. 13.6 *General idea behind the RSA digital signature scheme*

The signing and verifying sites use the same function, but with different parameters. The verifier compares the message and the output of the function for congruence. If the result is true, the message is accepted.

Key Generation Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA cryptosystem (see Chapter 10). Alice chooses two primes p and q and calculates $n = p \times q$. Alice calculates $\phi(n) = (p - 1)(q - 1)$. She then chooses e, the public exponent, and calculates d, the private exponent such that $e \times d = 1 \bmod \phi(n)$. Alice keeps d; she publicly announces n and e.

In the RSA digital signature scheme, d is private; e and n are public.

Signing and Verifying

Figure 13.7 shows the RSA digital signature scheme.

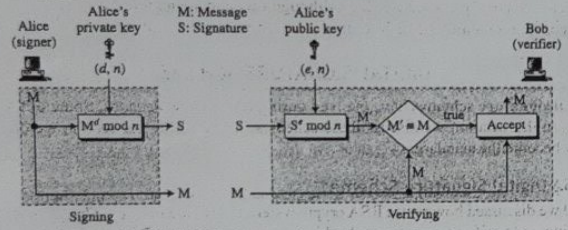


Fig. 13.7 *RSA digital signature scheme*

Signing Alice creates a signature out of the message using her private exponent, $S = M^d \bmod n$ and sends the message and the signature to Bob.

Verifying Bob receives M and S. Bob applies Alice's public exponent to the signature to create a copy of the message $M' = S^e \bmod n$. Bob compares the value of M' with the value of M. If the two values are congruent, Bob accepts the message. To prove this, we start with the verification criteria:

$$M' \equiv M \pmod n \rightarrow S^e \equiv M \pmod n \rightarrow M^{d \times e} \equiv M \pmod n$$

The last congruent holds because $d \times e = 1 \bmod \phi(n)$ (see Euler's theorem in Chapter 9).

Example 13.1 For the security of the signature, the value of p and q must be very large. As a trivial example, suppose that Alice chooses $p = 823$ and $q = 953$, and calculates $n = 784319$. The value of $\phi(n)$ is 782544. Now she chooses $e = 313$ and calculates $d = 160009$. At this point key generation is complete. Now imagine that Alice wants to send a message with the value of M = 19070 to Bob. She uses her private exponent, 160009, to sign the message:

$$M: 19070 \rightarrow S = (19070^{160009}) \bmod 784319 = 210625 \bmod 784319$$

Alice sends the message and the signature to Bob. Bob receives the message and the signature. He calculates

$$M' = 210625^{313} \bmod 784319 = 19070 \bmod 784319 \rightarrow M = M' \bmod n$$

Bob accepts the message because he has verified Alice's signature.

13.5.2 Attacks on RSA Signature

There are some attacks that Eve can apply to the RSA digital signature scheme to forge Alice's signature.

Key-Only Attack Eve has access only to Alice's public key. Eve intercepts the pair (M, S) and tries to create another message M' such that $M' \equiv S^e \pmod n$. This problem is as difficult to solve as the discrete logarithm problem we saw in Chapter 9. Besides, this is an existential forgery and normally is useless to Eve.

Known-Message Attack Here Eve uses the *multiplicative property* of RSA. Assume that Eve has intercepted two message-signature pairs (M_1, S_1) and (M_2, S_2) that have been created using the same private key. If $M = (M_1 \times M_2) \bmod n$, then $S = (S_1 \times S_2) \bmod n$. This is simple to prove because we have

$$S = (S_1 \times S_2) \bmod n = (M_1^d \times M_2^d) \bmod n = (M_1 \times M_2)^d \bmod n = M^d \bmod n$$

Eve can create $M = (M_1 \times M_2) \bmod n$, and she can create $S = (S_1 \times S_2) \bmod n$, and fool Bob into believing that S is Alice's signature on the message M. This attack, which is sometimes referred to as *multiplicative attack*, is easy to launch. However, this is an existential forgery as the message M is a multiplication of two previous messages created by Alice, not Eve; M is normally useless.

Chosen-Message Attack This attack also uses the multiplicative property of RSA. Eve can somehow ask Alice to sign two legitimate messages, M_1 and M_2, for her and later creates a new message $M = M_1 \times M_2$. Eve can later claim that Alice has signed M. The attack is also referred to as *multiplicative attack*. This is a very serious attack on the RSA digital signature scheme because it is a selective forgery (Eve can manipulate M_1 and M_2 to get a useful M).

RSA Signature on the Message Digest As we discussed before, signing a message digest using a strong hash algorithm has several advantages. In the case of RSA, it can make the signing and verifying processes much faster because the RSA digital signature scheme is nothing other than encryption with the private key and decryption with the public key. The use of a strong cryptographic hashing function also makes the attack on the signature much more difficult as we will explain shortly. Figure 13.8 shows the scheme.

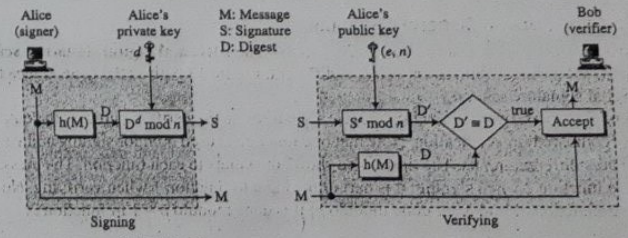


Fig. 13.8 *The RSA signature on the message digest*

Alice, the signer, first uses an agreed-upon hash function to create a digest from the message: D = h(M). She then signs the digest, $S = D^d \bmod n$. The message and the signature are sent to Bob. Bob, the verifier, receives the message and the signature. He first uses Alice's public exponent to retrieve the digest, $D' = S^e \bmod n$. He then applies the hash algorithm to the message received to obtain D′ = h(M). Bob now compares the two digests, D and D′. If they are congruent to modulo *n*, he accepts the message.

Attacks on RSA Signed Digests

How susceptible to attack is the RSA digital signature scheme when the digest is signed?

Key-Only Attack We can have three cases of this attack:

a. Eve intercepts the pair (S, M) and tries to find another message M′ that creates the same digest: h(M) = h(M′). As we learned in Chapter 11, if the hash algorithm is *second preimage resistant*, this attack is very difficult.
b. Eve finds two messages M and M′ such that h(M) = h(M′). She lures Alice to sign h(M) to find S. Now Eve has a pair (M′, S) which passes the verifying test, but it is the forgery. We learned in Chapter 11 that if the hash algorithm is *collision resistant*, this attack is very difficult.
c. Eve may randomly find message digest D, which may match with a random signature S. She then finds a message M such that D = h(M). As we learned in Chapter 11, if the hash function is *preimage resistant*, this attack is very difficult to launch.

Known-Message Attack Let us assume Eve has two message-signature pairs (M_1, S_1) and (M_2, S_2) which have been created using the same private key. Eve calculates $S \equiv S_1 \times S_2$. If she can find a message M such that $h(M) \equiv h(M_1) \times h(M_2)$, she has forged a new message. However, finding M given h(M) is very difficult if the hash algorithm is *preimage resistant*.

Chosen-Message Attack Eve can ask Alice to sign two legitimate messages M_1 and M_2 for her. Eve then creates a new signature $S \equiv S_1 \times S_2$. Since Eve can calculate $h(M) \equiv h(M_1) \times h(M_2)$, if she can find a message M given h(M), the new message is a forgery. However, finding M given h(M) is very difficult if the hash algorithm is *preimage resistant*.

When the digest is signed instead of the message itself, the susceptibility of the RSA digital signature scheme depends on the strength of the hash algorithm.

13.5.3 ElGamal Digital Signature Scheme

The ElGamal cryptosystem was discussed in Chapter 10. The **ElGamal digital signature scheme** uses the same keys, but the algorithm, as expected, is different. Figure 13.9 gives the general idea behind the ElGamal digital signature scheme.

In the signing process, two functions create two signatures; in the verifying process the outputs of two functions are compared for verification. Note that one function is used both for signing and verifying but the function uses different inputs. The figure also shows the inputs to each function. The message is part of the input to function 2 when signing; it is part of the input to function 1 when verifying. Note that the calculations in functions 1 and 3 are done modulo *p*; it is done modulo $p - 1$ in function 2.

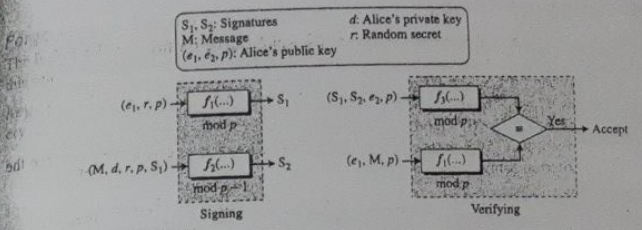


Fig. 13.9 *General idea behind the ElGamal digital signature scheme*

Key Generation The key generation procedure here is exactly the same as the one used in the cryptosystem. Let *p* be a prime number large enough that the discrete log problem is intractable in $\mathbf{Z}_p^*$. Let e_1 be a primitive element in $\mathbf{Z}_p^*$. Alice selects her private key *d* to be less than $p - 1$. She calculates $e_2 = e_1^d$. Alice's public key is the tuple (e_1, e_2, p); Alice's private key is *d*.

In ElGamal digital signature scheme, (e_1, e_2, p) is Alice's public key; *d* is her private key.

Verifying and Signing

Figure 13.10 shows the ElGamal digital signature scheme.

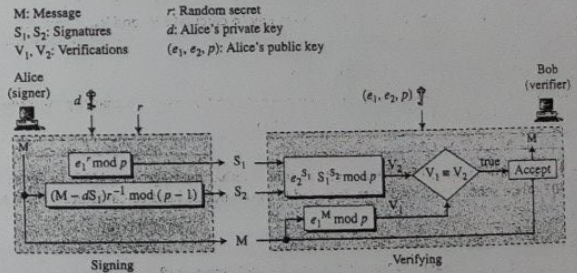


Fig. 13.10 *ElGamal digital signature scheme*

Signing Alice can sign the digest of a message to any entity, including Bob:

1. Alice chooses a secret random number *r*. Note that although public and private keys can be used repeatedly, Alice needs a new *r* each time she signs a new message.
2. Alice calculates the first signature $S_1 = e_1^r \bmod p$.
3. Alice calculates the second signature $S_2 = (M - d \times S_1) \times r^{-1} \bmod (p - 1)$, where r^{-1} is the multiplicative inverse of *r* modulo *p*.