CHAPTER 13

Digital Signatures
13.1 Digital Signatures

Properties
Attacks and Forgeries
Digital Signature Requirements
Direct Digital Signature

13.2 ElGamal Digital Signature Scheme

13.3 Schnorr Digital Signature Scheme

13.4 Digital Signature Standard

The DSS Approach
The Digital Signature Algorithm
KEY POINTS

◆ A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. Typically the signature is formed by taking the hash of the message and encrypting the message with the creator’s private key. The signature guarantees the source and integrity of the message.

◆ The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).
Figure 13.1  Generic Model of Digital Signature Process
Figure 13.2  Simplified Depiction of Essential Elements of Digital Signature Process
13.1 DIGITAL SIGNATURES

Properties

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two are possible.
For example, suppose that John sends an authenticated message to Mary, using one of the schemes of Figure 12.1. Consider the following disputes that could arise.

1. **Mary may forge a different message and claim that it came from John.** Mary would simply have to create a message and append an authentication code using the key that John and Mary share.

2. **John can deny sending the message.** Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

Both scenarios are of legitimate concern. Here is an example of the first scenario: An electronic funds transfer takes place, and the receiver increases the amount of funds transferred and claims that the larger amount had arrived from the sender. An example of the second scenario is that an electronic mail message contains instructions to a stockbroker for a transaction that subsequently turns out badly. The sender pretends that the message was never sent.
In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature. The digital signature must have the following properties:

- It must verify the author and the date and time of the signature.
- It must authenticate the contents at the time of the signature.
- It must be verifiable by third parties, to resolve disputes.

Thus, the digital signature function includes the authentication function.

**Attacks and Forgeries**

[GOLD88] lists the following types of attacks, in order of increasing severity. Here A denotes the user whose signature method is being attacked, and C denotes the attacker.
- **Key-only attack**: C only knows A’s public key.
- **Known message attack**: C is given access to a set of messages and their signatures.
- **Generic chosen message attack**: C chooses a list of messages before attempting to breaks A’s signature scheme, independent of A’s public key. C then obtains from A valid signatures for the chosen messages. The attack is generic, because it does not depend on A’s public key; the same attack is used against everyone.
- **Directed chosen message attack**: Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A’s public key but before any signatures are seen.
- **Adaptive chosen message attack**: C is allowed to use A as an “oracle.” This means the A may request signatures of messages that depend on previously obtained message–signature pairs.
[GOLD88] then defines success at breaking a signature scheme as an outcome in which C can do any of the following with a non-negligible probability:

- **Total break**: C determines A’s private key.
- **Universal forgery**: C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.
- **Selective forgery**: C forges a signature for a particular message chosen by C.
- **Existential forgery**: C forges a signature for at least one message. C has no control over the message. Consequently, this forgery may only be a minor nuisance to A.
Digital Signature Requirements

On the basis of the properties and attacks just discussed, we can formulate the following requirements for a digital signature.

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique to the sender to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.
13.2 ELGAMAL DIGITAL SIGNATURE SCHEME

Before examining the NIST Digital Signature standard, it will be helpful to understand the ElGamal and Schnorr signature schemes. Recall from Chapter 10, that the ElGamal encryption scheme is designed to enable encryption by a user’s public key with decryption by the user’s private key. The ElGamal signature scheme involves the use of the private key for encryption and the public key for decryption [Elg84, Elg85].

Before proceeding, we need a result from number theory. Recall from Chapter 8 that for a prime number $q$, if $\alpha$ is a primitive root of $q$, then

$$\alpha, \ \alpha^2, \ldots, \alpha^{q-1}$$

are distinct (mod $q$). It can be shown that, if $\alpha$ is a primitive root of $q$, then

1. For any integer $m$, $\alpha^m \equiv 1 \pmod{q}$ if and only if $m \equiv 0 \pmod{q-1}$.
2. For any integers, $i, j$, $\alpha^i \equiv \alpha^j \pmod{q}$ if and only if $i \equiv j \pmod{q-1}$.
As with ElGamal encryption, the global elements of ElGamal digital signature are a prime number \( q \) and \( \alpha \), which is a primitive root of \( q \). User A generates a private/public key pair as follows.

1. Generate a random integer \( X_A \), such that \( 1 < X_A < q - 1 \).
2. Compute \( Y_A = \alpha^{X_A} \mod q \).
3. A’s private key is \( X_A \); A’s public key is \( q, \alpha, Y_A \).

To sign a message \( M \), user A first computes the hash \( m = H(M) \), such that \( m \) is an integer in the range \( 0 \leq m \leq q - 1 \). A then forms a digital signature as follows.

1. Choose a random integer \( K \) such that \( 1 \leq K \leq q - 1 \) and \( \gcd(K, q - 1) = 1 \). That is, \( K \) is relatively prime to \( q - 1 \).
2. Compute \( S_1 = \alpha^K \mod q \). Note that this is the same as the computation of \( C_1 \) for ElGamal encryption.
3. Compute \( K^{-1} \mod (q - 1) \). That is, compute the inverse of \( K \) modulo \( q - 1 \).
4. Compute \( S_2 = K^{-1}(m - X_A S_1) \mod (q - 1) \).
5. The signature consists of the pair \( (S_1, S_2) \).
Any user B can verify the signature as follows.

1. Compute $V_1 = \alpha^m \mod q$.
2. Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \mod q$.

The signature is valid if $V_1 = V_2$. Let us demonstrate that this is so. Assume that the equality is true. Then we have

\[
\alpha^m \mod q = (Y_A)^{S_1}(S_1)^{S_2} \mod q
\]

\[
\alpha^m \mod q = \alpha^{X_A S_1} \alpha^{K S_2} \mod q
\]

\[
\alpha^{m - X_A S_1} \mod q = \alpha^{K S_2} \mod q
\]

\[
m - X_A S_1 \equiv K S_2 \mod (q - 1)
\]

\[
m - X_A S_1 \equiv K K^{-1}(m - X_A S_1) \mod (q - 1)
\]

assuming $V_1 = V_2$

substituting for $Y_A$ and $S_1$

rearranging terms

property of primitive roots

substituting for $S_2$
For example, let us start with the prime field $GF(19)$; that is, $q = 19$. It has primitive roots $\{2, 3, 10, 13, 14, 15\}$, as shown in Table 8.3. We choose $\alpha = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 16$.
2. Then $Y_A = \alpha^{X_A} \mod q = \alpha^{16} \mod 19 = 4$.
3. Alice’s private key is 16; Alice’s public key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.

Suppose Alice wants to sign a message with hash value $m = 14$.

1. Alice chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
2. $S_1 = \alpha^K \mod q = 10^5 \mod 19 = 3$ (see Table 8.3).
3. \( K^{-1} \mod (q - 1) = 5^{-1} \mod 18 = 11. \)

4. \( S_2 = K^{-1}(m - X_A S_1) \mod (q - 1) = 11(14 - (16)(3)) \mod 18 = -374 \mod 18 = 4. \)

Bob can verify the signature as follows.

1. \( V_1 = \alpha^m \mod q = 10^{14} \mod 19 = 16. \)

2. \( V_2 = (Y_A)^{S_1}(S_1)^{S_2} \mod q = (4^3)(3^4) \mod 19 = 5184 \mod 19 = 16. \)

Thus, the signature is valid.
13.3 SCHRÖR DIGITAL SIGNATURE SCHEME

As with the ElGamal digital signature scheme, the Schnorr signature scheme is based on discrete logarithms [SCHN89, SCHN91]. The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature. The main work for signature generation does not depend on the message and can be done during the idle time of the processor. The message-dependent part of the signature generation requires multiplying a $2n$-bit integer with an $n$-bit integer.

The scheme is based on using a prime modulus $p$, with $p - 1$ having a prime factor $q$ of appropriate size; that is, $p - 1 \equiv 0 \pmod{q}$. Typically, we use $p \approx 2^{1024}$ and $q \approx 2^{160}$. Thus, $p$ is a 1024-bit number, and $q$ is a 160-bit number, which is also the length of the SHA-1 hash value.
The first part of this scheme is the generation of a private/public key pair, which consists of the following steps.

1. Choose primes \( p \) and \( q \), such that \( q \) is a prime factor of \( p - 1 \).

2. Choose an integer \( a \), such that \( a^q \equiv 1 \pmod{p} \). The values \( a, p, \) and \( q \) comprise a global public key that can be common to a group of users.

3. Choose a random integer \( s \) with \( 0 < s < q \). This is the user’s private key.

4. Calculate \( v = a^{-s} \pmod{p} \). This is the user’s public key.

A user with private key \( s \) and public key \( v \) generates a signature as follows.

1. Choose a random integer \( r \) with \( 0 < r < q \) and compute \( x = a'^r \pmod{p} \). This computation is a preprocessing stage independent of the message \( M \) to be signed.

\[
3' \quad 3^2 \quad 3^3 \equiv 5 \quad 3^4 = 4 \quad 3^5 = 1 \pmod{p=11}
\]
2. Concatenate the message with \( x \) and hash the result to compute the value \( e \):

\[
e = H(M \parallel x)
\]

3. Compute \( y = (r + se) \mod q \). The signature consists of the pair \((e, y)\).

Any other user can verify the signature as follows.

1. Compute \( x' = a^y v^e \mod p \).

2. Verify that \( e = H(M \parallel x') \).

To see that the verification works, observe that

\[
x' = a^y v^e = a^y a^{-se} = a^{y-se} = a^r = x \pmod{p}
\]

Hence, \( H(M \parallel x') = H(M \parallel x) \).
The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS). The DSS makes use of the Secure Hash Algorithm (SHA) described in Chapter 12 and presents a new digital signature technique, the Digital Signature Algorithm (DSA). The DSS was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2, subsequently updated to FIPS 186-3 in 2009. This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography. In this section, we discuss the original DSS algorithm.
Figure 13.3  Two Approaches to Digital Signatures
Global Public-Key Components

- **p**  
  prime number where \(2^{L-1} < p < 2^L\)  
  for \(512 \leq L \leq 1024\) and \(L\) a multiple of 64;  
  i.e., bit length of between 512 and 1024 bits  
  in increments of 64 bits

- **q**  
  prime divisor of \((p - 1)\), where \(2^{159} < q < 2^{160}\);  
  i.e., bit length of 160 bits

- **g**  
  \(g = h^{(p-1)/q} \mod p\),  
  where \(h\) is any integer with \(1 < h < (p - 1)\)  
  such that \(h^{(p-1)/q} \mod p > 1\)

User’s Private Key

- **x**  
  random or pseudorandom integer with \(0 < x < q\)

Signing

- \(r = (g^k \mod p) \mod q\)
- \(s = [k^{-1} (H(M) + xr)] \mod q\)
- Signature = \((r; s)\)

Verifying

- \(w = (s')^{-1} \mod q\)
- \(u_1 = [H(M')w] \mod q\)
- \(u_2 = (r')w \mod q\)
- \(v = [(g^{u_1} y^{u_2}) \mod p] \mod q\)
- TEST: \(v = r'\)
User’s Public Key

\[ y = g^x \mod p \]

User’s Per-Message Secret Number

\[ k = \text{random or pseudorandom integer with } 0 < k < q \]

\[ M \quad = \text{message to be signed} \]
\[ H(M) \quad = \text{hash of } M \text{ using SHA-1} \]
\[ M', r', s' \quad = \text{received versions of } M, r, s \]

Figure 13.4 The Digital Signature Algorithm (DSA)
\[ s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \mod q \]

\[ r = f_2(k, p, q, g) = (g^k \mod p) \mod q \]

\[ w = f_3(s', q) = (s')^{-1} \mod q \]

\[ v = f_4(y, q, g, H(M'), w, r') \]

\[ = ((g^{H(M')w} \mod q \cdot y'^w \mod q) \mod p) \mod q \]

(a) Signing

(b) Verifying

**Figure 13.5** DSS Signing and Verifying