

CYB650

Introduction to Cryptography

Public Key Algorithms



Prepared by Dr. Esraa Saleh Alomari
Lecture #10



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

The Part 2 of the module 5 covers the Public-Key Algorithms are as follows:

➤ ***Digital Signature Algorithm (DSA)***

Introduction

- ❑ The most important development from the work on public-key cryptography is the digital signature. The digital signature provides a set of security capabilities that would be difficult to implement in any other way.

- ❑ Figure 13.1 is a generic model of the process of making and using digital signatures. Bob can sign a message using a digital signature generation algorithm. The inputs to the algorithm are the message and Bob's private key. Any other user, say Alice, can verify the signature using a verification algorithm, whose inputs are the message, the signature, and Bob's public key. In simplified terms, the essence of the digital signature mechanism is shown in Figure 13.2.

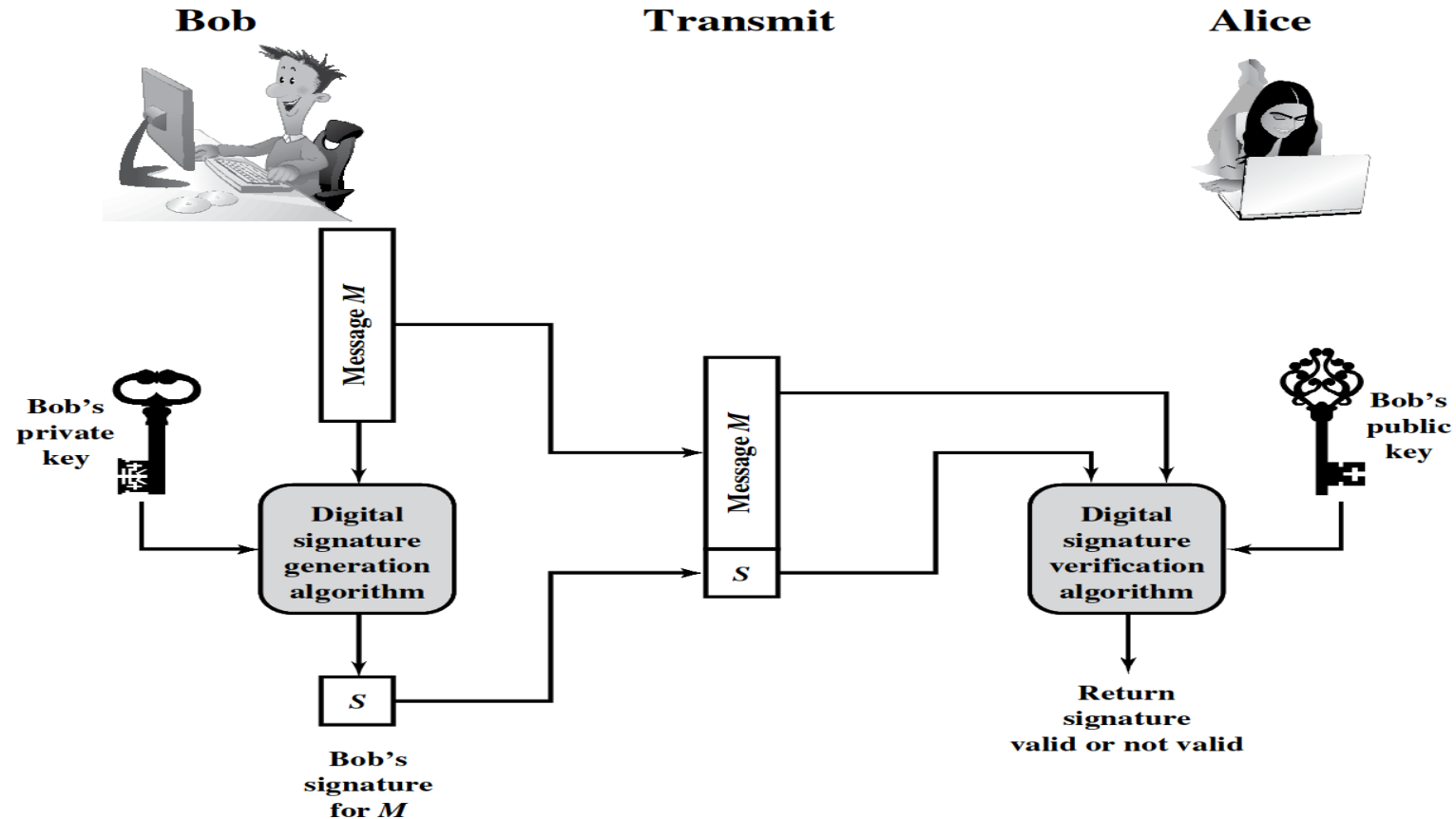


Figure 13.1 Generic Model of Digital Signature Process

- ❑ 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.

Digital Signature- Properties

- ❑ 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.

Digital Signature- Properties

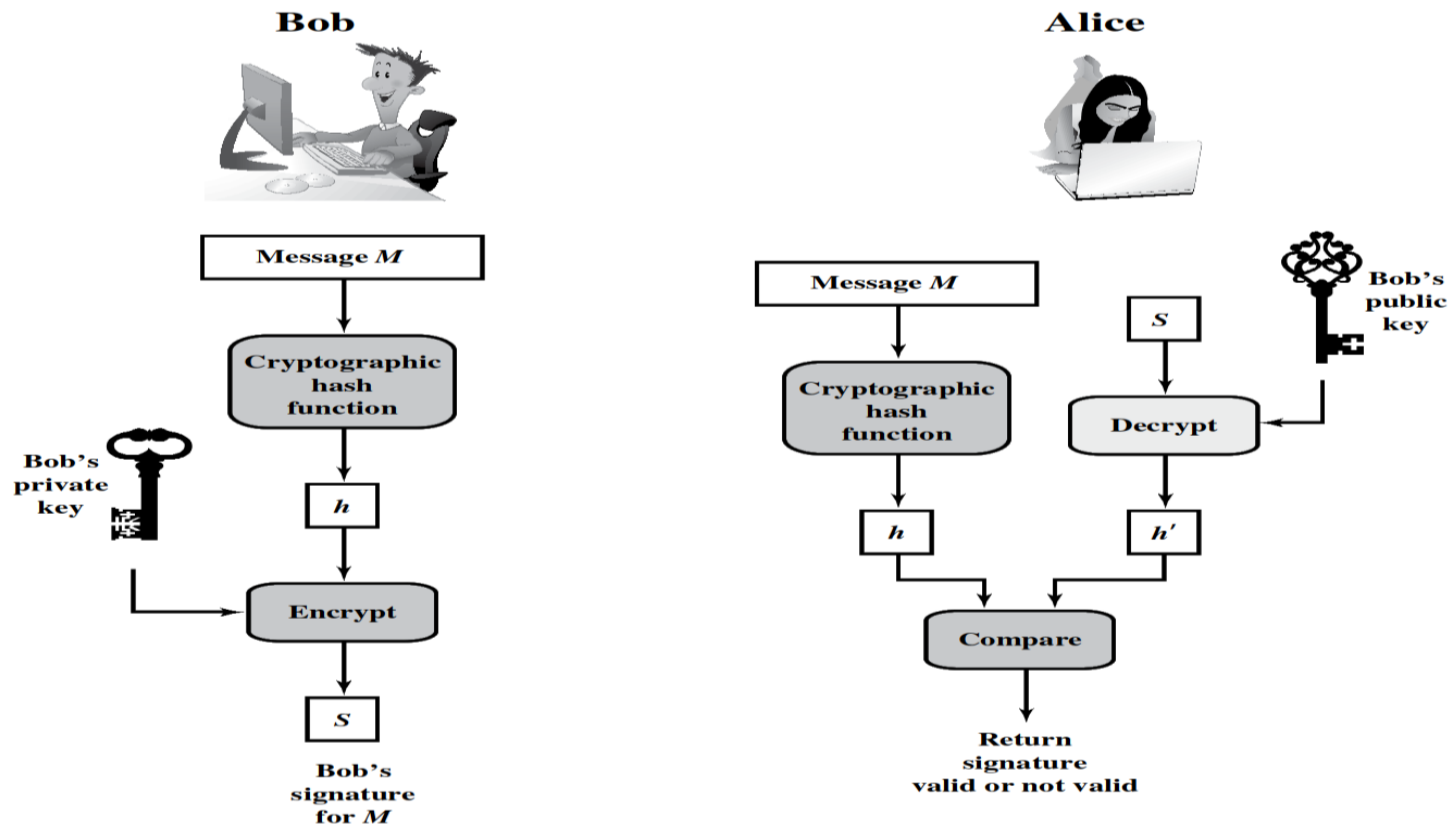


Figure 13.2 Simplified Depiction of Essential Elements of Digital Signature Process

Digital Signature- Properties

- ❑ 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.

Digital Signature- Properties

- ❑ 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.

- ❑ *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 break 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 that C may request from A 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.

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.

A secure hash function, embedded in a scheme such as that of Figure 13.2, provides a basis for satisfying these requirements. However, care must be taken in the design of the details of the scheme.

The term direct digital signature refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source. Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption).

Note that it is important to perform the signature function first and then an outer confidentiality function. In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.

The validity of the scheme just described depends on the security of the sender's private key. If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature. Administrative controls relating to the security of private keys can be employed to thwart or at least weaken this ploy, but the threat is still there, at least to some degree.

One example is to require every signed message to include a timestamp (date and time) and to require prompt reporting of compromised keys to a central authority.

Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

Federal Information Processing Standard FIPS 186, known as the Digital Signature Algorithm (DSA). The DSA makes use of the Secure Hash Algorithm (SHA). The DSA 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.

The DSA uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

Figure 13.3 contrasts the DSA approach for generating digital signatures to that used with RSA.



In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted.

The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

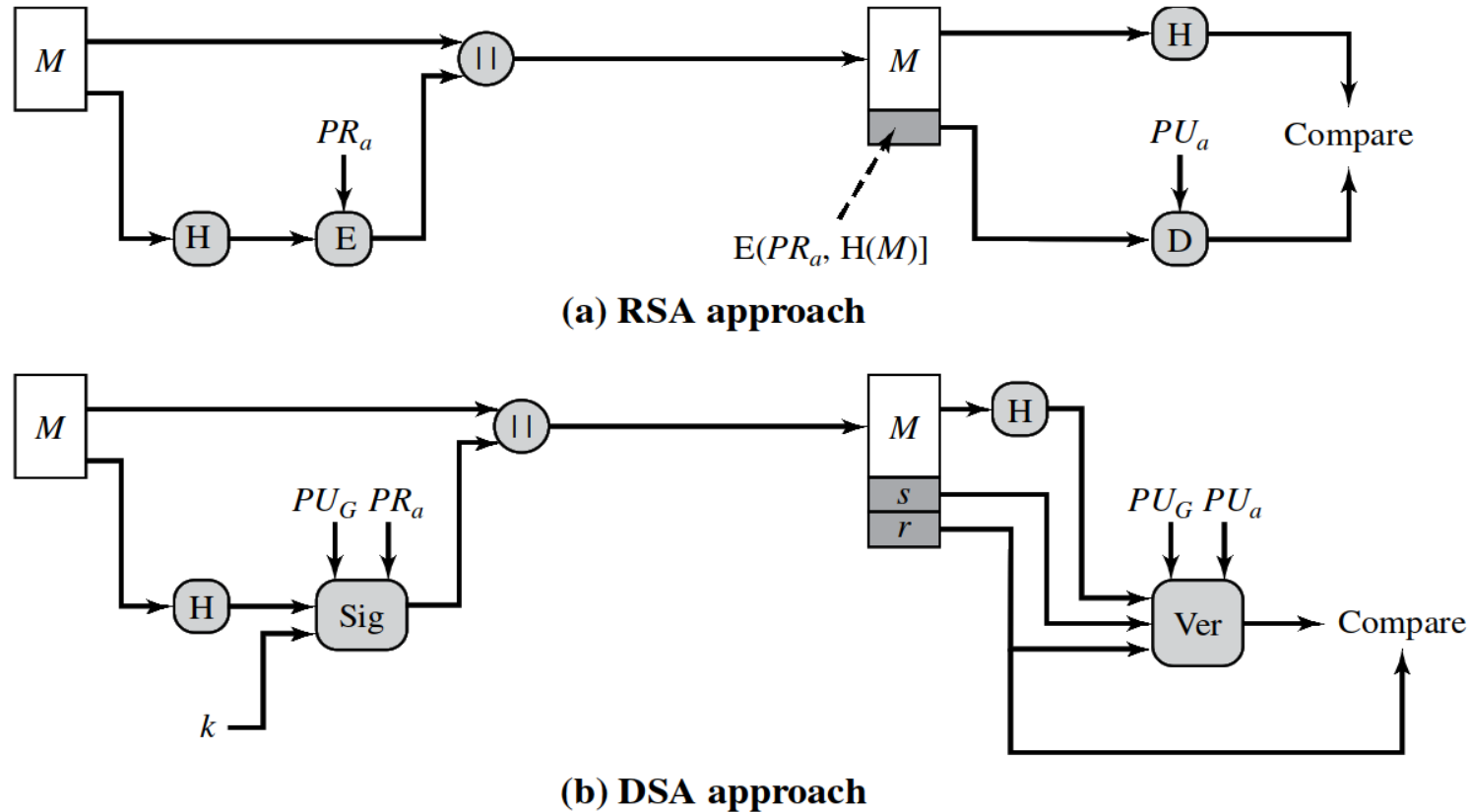


Figure 13.3 Two Approaches to Digital Signatures

The DSA approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number k generated for this particular signature. The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G).¹ The result is a signature consisting of two components, labeled s and r .

At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key. The output of the verification function is a value that is equal to the signature component r if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by Elgamal . Figure 13.4 summarizes the algorithm. There are three parameters that are public and can be common to a group of users. An N -bit prime number q is chosen.



Next, a prime number p is selected with a length between 512 and 1024 bits such that q divides $(p-1)$. Finally g is chosen to be of form $h^{(p-1)/q} \bmod p$ where h is an integer between 1 and $(p-1)$ with the restriction that must be greater than 1.2 Thus, the global public-key components of DSA are the same as in the Schnorr signature scheme.

With these numbers in hand, each user selects a private key and generates a public key. The private key x must be a number from 1 to $(q - 1)$ and should be chosen randomly or pseudorandomly. The public key is calculated from the private key as $y = g^x \bmod p$. The calculation of y given x is relatively straightforward. However, given the public key y , it is believed to be computationally infeasible to determine x , which is the discrete logarithm of y to the base g , mod p (see Chapter 8).

To create a signature, a user calculates two quantities, r and s , that are functions of the public key components (p, q, g) , the user's private key (x) , the hash code of the message $H(M)$, and an additional integer k that should be generated randomly or pseudorandomly and be unique for each signing.

Let M , r' , and s' be the received versions of M , r , and s , respectively. Verification is performed using the formulas shown in Figure 13.4. The receiver generates a quantity v that is a function of the public key components, the sender's public key, and the hash code of the incoming message. If this quantity matches the r component of the signature, then the signature is validated.

The Digital Signature Algorithm

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 $2^{N-1} < q < 2^N$
i.e., bit length of N bits
- $g = h(p - 1)/q \bmod p$,
where h is any integer with $1 < h < (p - 1)$
such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

- x random or pseudorandom integer with $0 < x < q$

User's Public Key

- $y = g^x \bmod p$

User's Per-Message Secret Number

- k random or pseudorandom integer with $0 < k < q$

Signing

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

Verifying

- $= (s')^{-1} \bmod q$
- $u_1 = [H(M') \] \bmod q$
- $u_2 = (r')w \bmod q$
- $= [(g^{u_1} y^{u_2}) \bmod p] \bmod q$
- TEST: $v = r'$

M = message to be signed

$H(M)$ = hash of M using SHA-1

M', r', s' = received versions of M, r, s

Figure 13.4 The Digital Signature Algorithm (DSA)

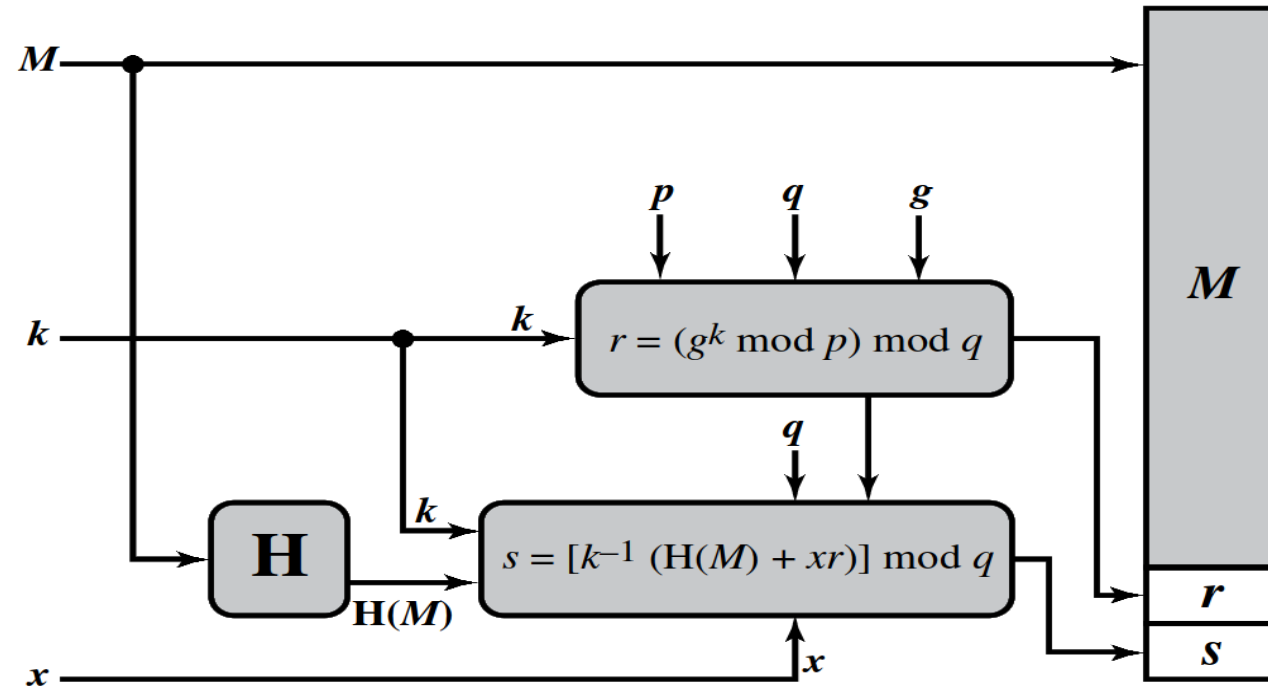
Figure 13.5 depicts the functions of signing and verifying.

The structure of the algorithm, as revealed in Figure 13.5, is quite interesting. Note that the test at the end is on the value r , which does not depend on the message at all. Instead, r is a function of k and the three global public-key components. The multiplicative inverse of $k \pmod{q}$ is passed to a function that also has as inputs the message hash code and the user's private key. The structure of this function is such that the receiver can recover r using the incoming message and signature, the public key of the user, and the global public key. It is certainly not obvious from Figure 13.4 or Figure 13.5 that such a scheme would work. A proof is provided in Appendix K.



Given the difficulty of taking discrete logarithms, it is infeasible for an opponent to recover k from r or to recover x from s .

Another point worth noting is that the only computationally demanding task in signature generation is the exponential calculation $g^k \bmod p$. Because this value does not depend on the message to be signed, it can be computed ahead of time. Indeed, a user could precalculate a number of values of r to be used to sign documents as needed. The only other somewhat demanding task is the determination of a multiplicative inverse, k^{-1} . Again, a number of these values can be precalculated.



(a) Signing

The Digital Signature Algorithm

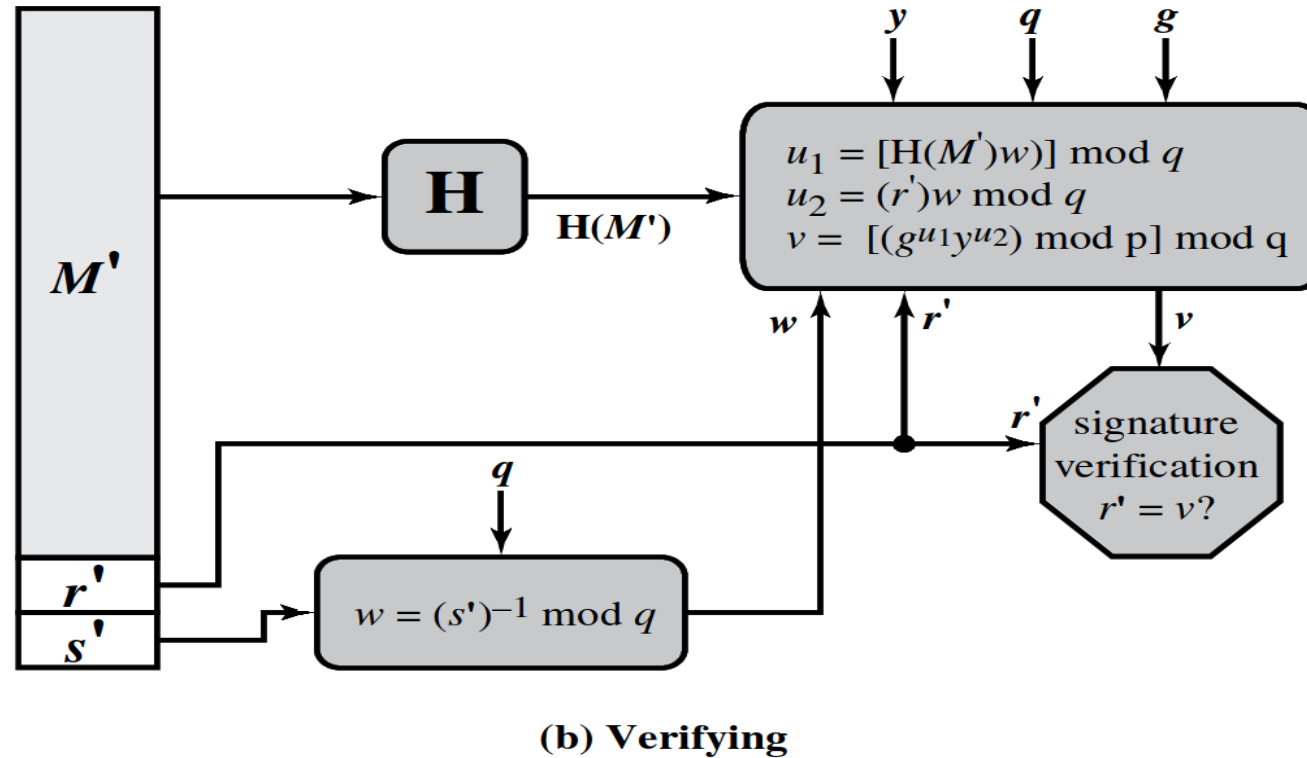


Figure 13.5 DSA Signing and Verifying

Thank You

ANY
QUESTIONS?

