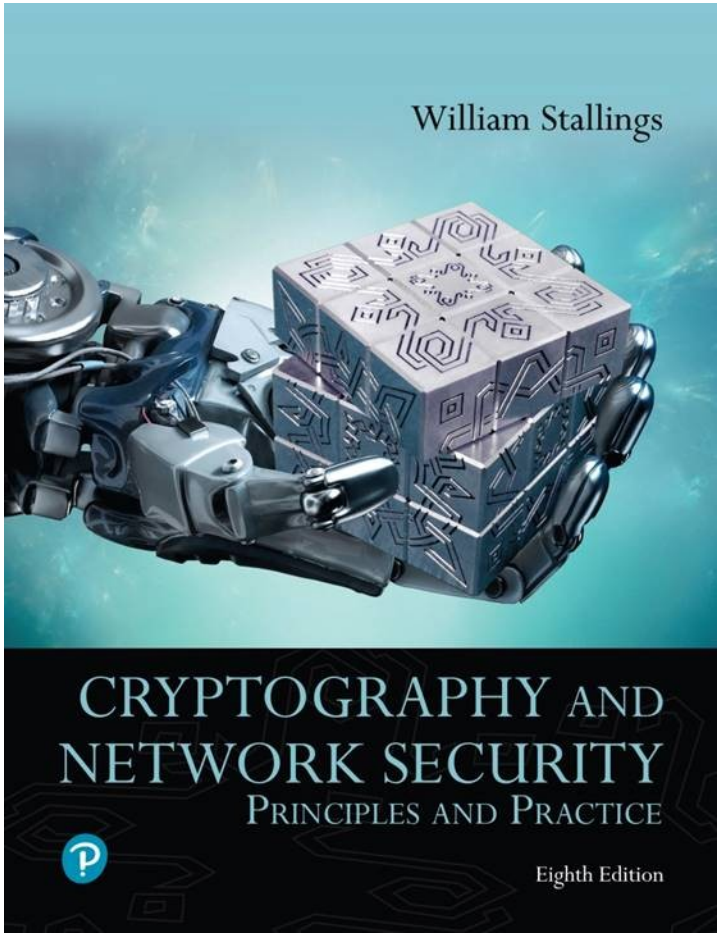
Cryptography and Network Security: Principles and Practice

Eighth Edition

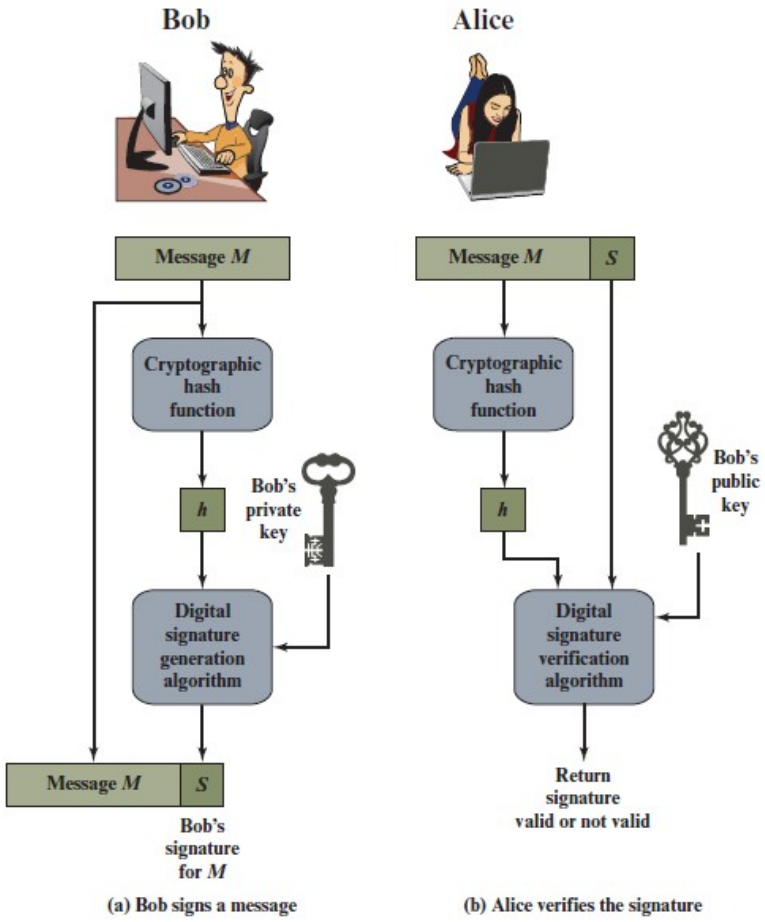


Chapter 13

Digital Signatures

Pearson

Figure 13.1 Simplified Depiction of Essential Elements of Digital Signature Process



Digital Signature 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

A secure digital signature scheme, like a secure conventional signature can provide message authentication.

Note

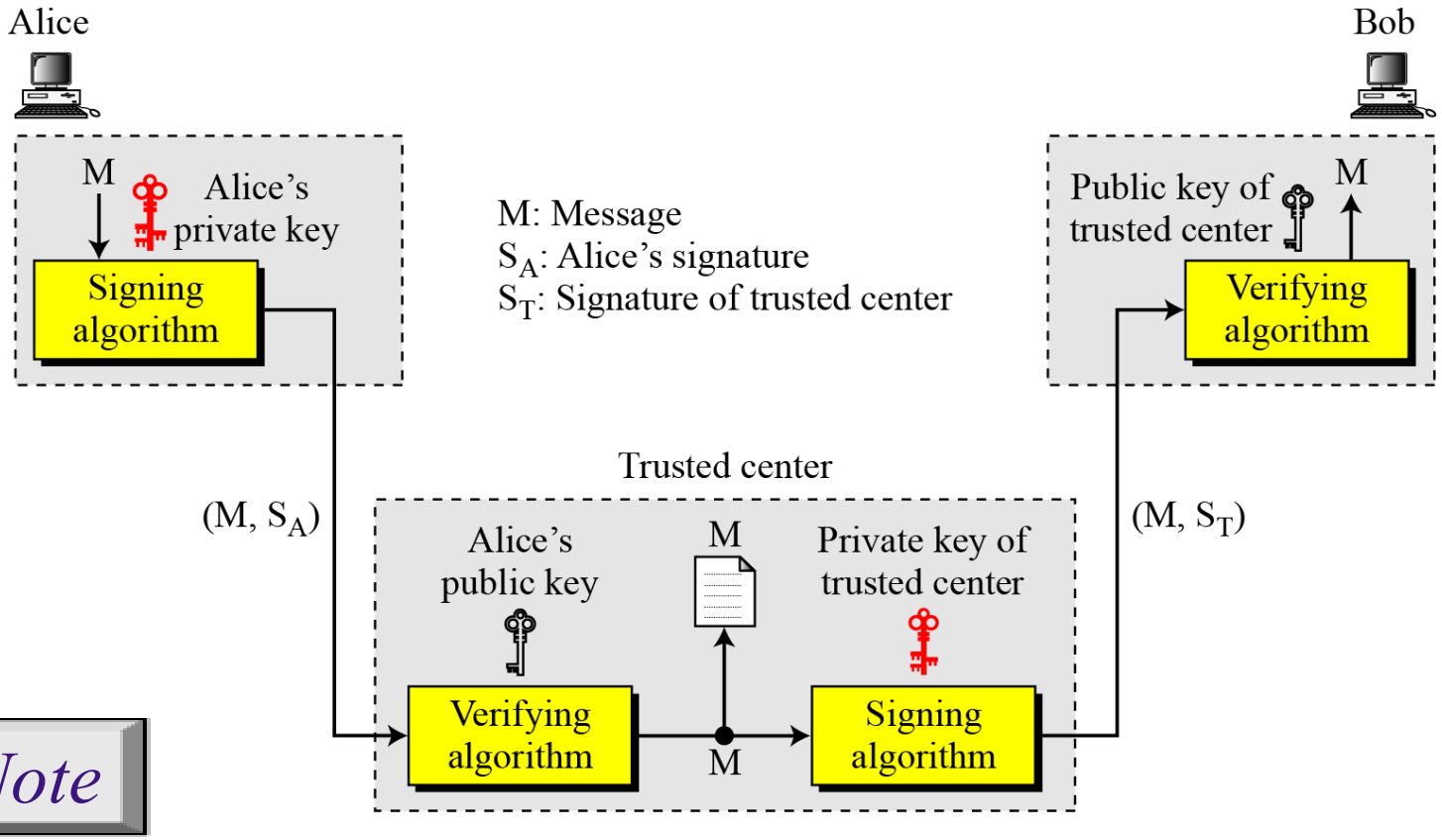
A digital signature provides message authentication.

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.

Note

A digital signature provides message integrity.

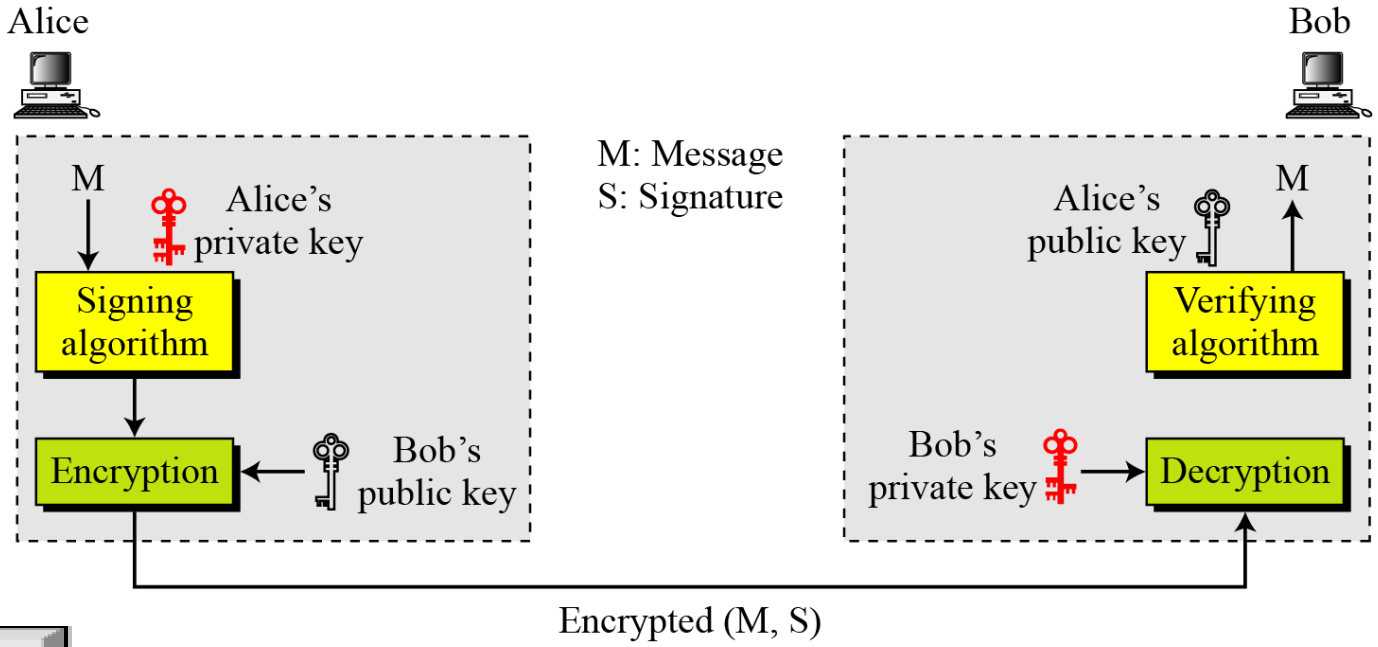
Figure 13.4 *Using a trusted center for nonrepudiation*



Note

Nonrepudiation can be provided using a trusted party.

Figure 13.5 *Adding confidentiality to a digital signature scheme*



Note

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

Attacks

- **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
- **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 may request from A signatures of messages that depend on previously obtained message-signature pairs

Forgeries

- **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

Digital Signature Requirements

- 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

Direct Digital Signature

- Refers to a digital signature scheme that involves only the communicating parties
  - 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
  - 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
- The validity of the scheme 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
  - One way to thwart or at least weaken this ploy is to require every signed message to include a timestamp and to require prompt reporting of compromised keys to a central authority

Key Generation

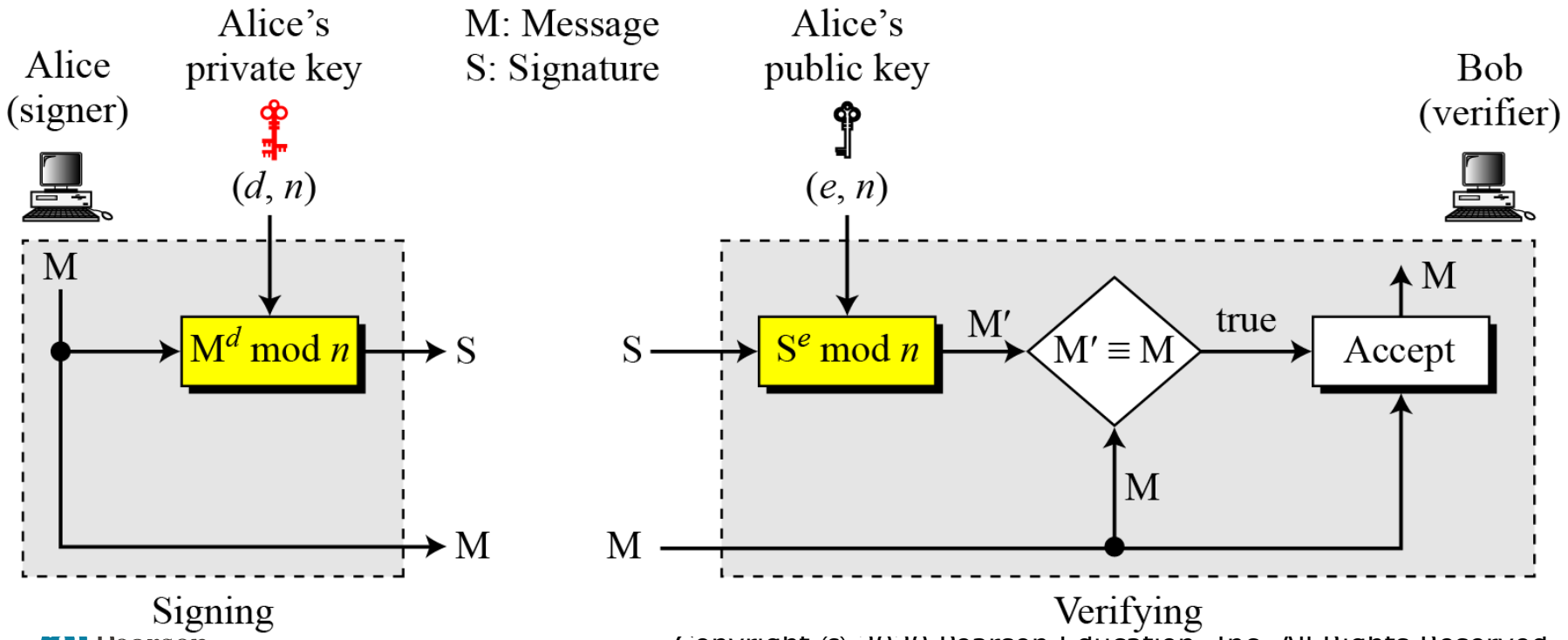
Key generation in the RSA digital signature scheme is exactly the same as key generation in the RSA

Note

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

Signing and Verifying

Figure 13.7 *RSA digital signature scheme*



Example 13.1

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:

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

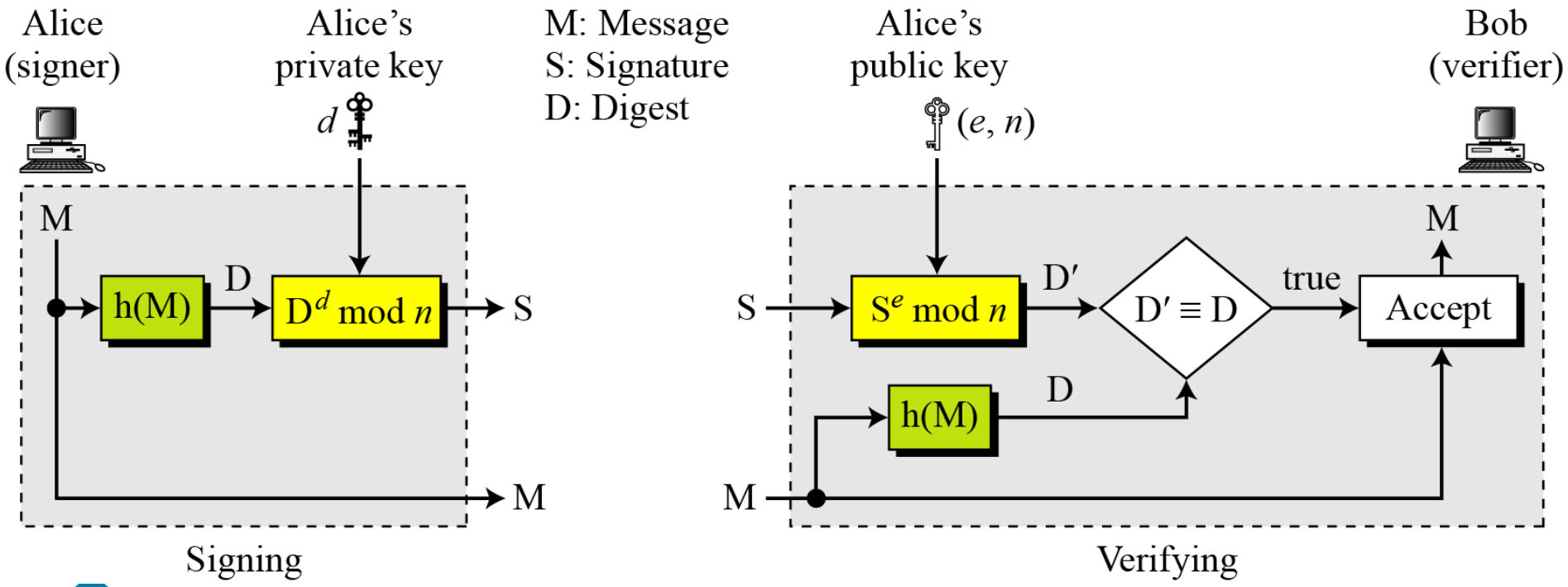
Alice sends the message and the signature to Bob. Bob receives the message and the signature. He calculates

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

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

RSA Signature on the Message Digest

Figure 13.8 *The RSA signature on the message digest*



Note

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.

ElGamal Digital Signature

- Scheme involves the use of the private key for encryption and the public key for decryption
- Global elements are a prime number q and a, which is a primitive root of q
- Use private key for encryption (signing)
- Uses public key for decryption (verification)
- Each user generates their key
  - Chooses a secret key (number): $1 < x_A < q-1$
  - Compute their public key: $y_A = a^{xA} \bmod q$

10.4.2 Procedure

Figure 10.11 *Key generation, encryption, and decryption in ElGamal*

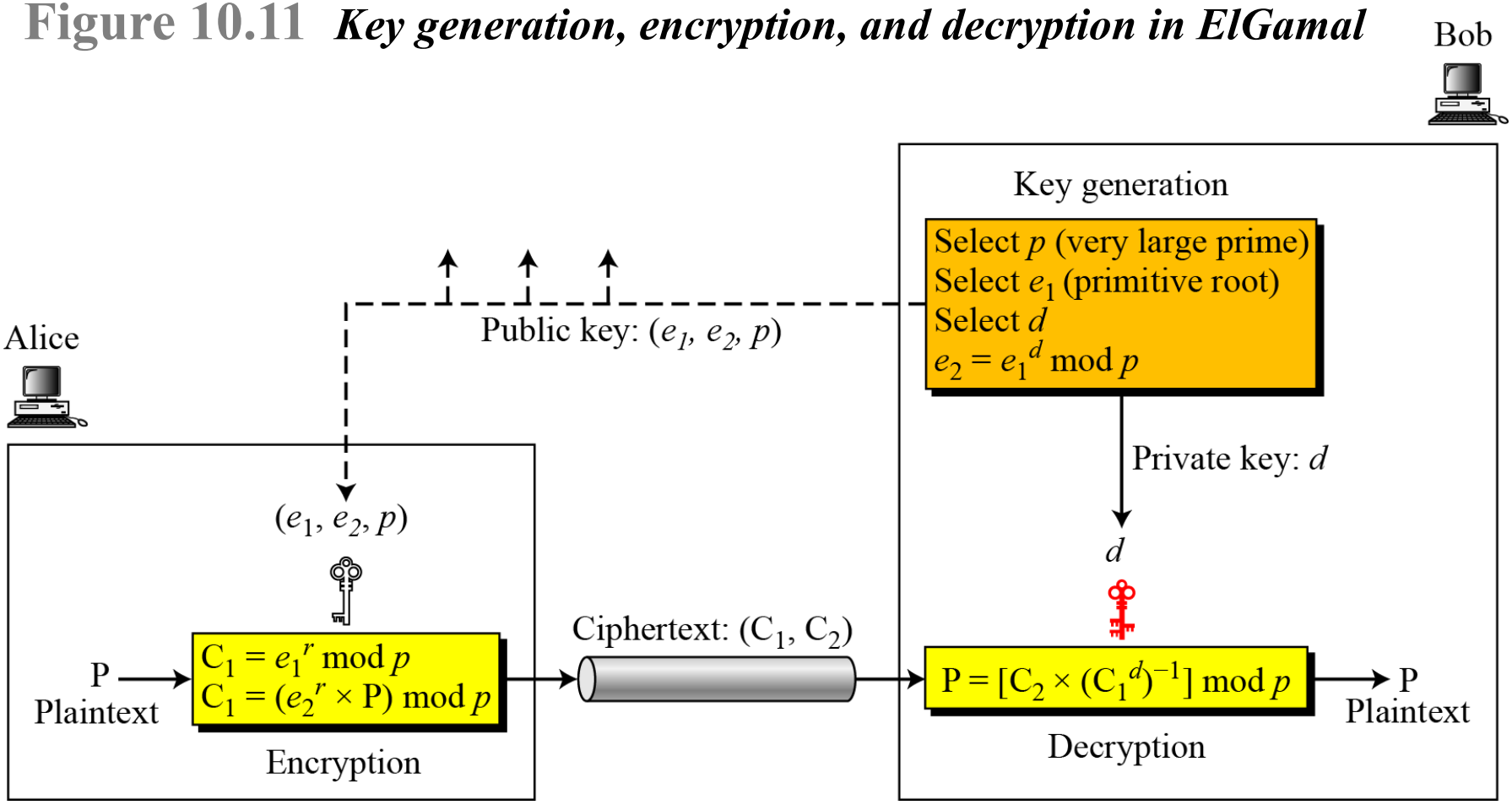
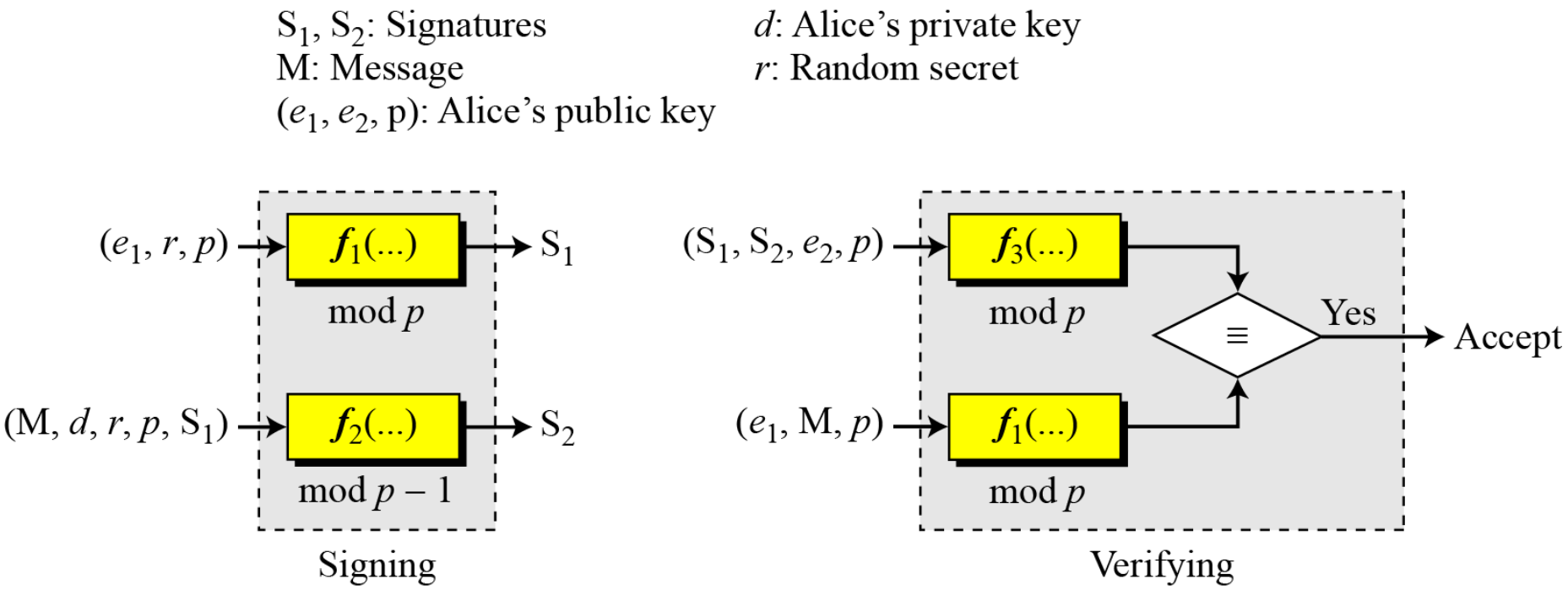


Figure 13.9 *General idea behind the ElGamal digital signature scheme*



Key Generation

The key generation procedure here is exactly the same as the one used in the cryptosystem.

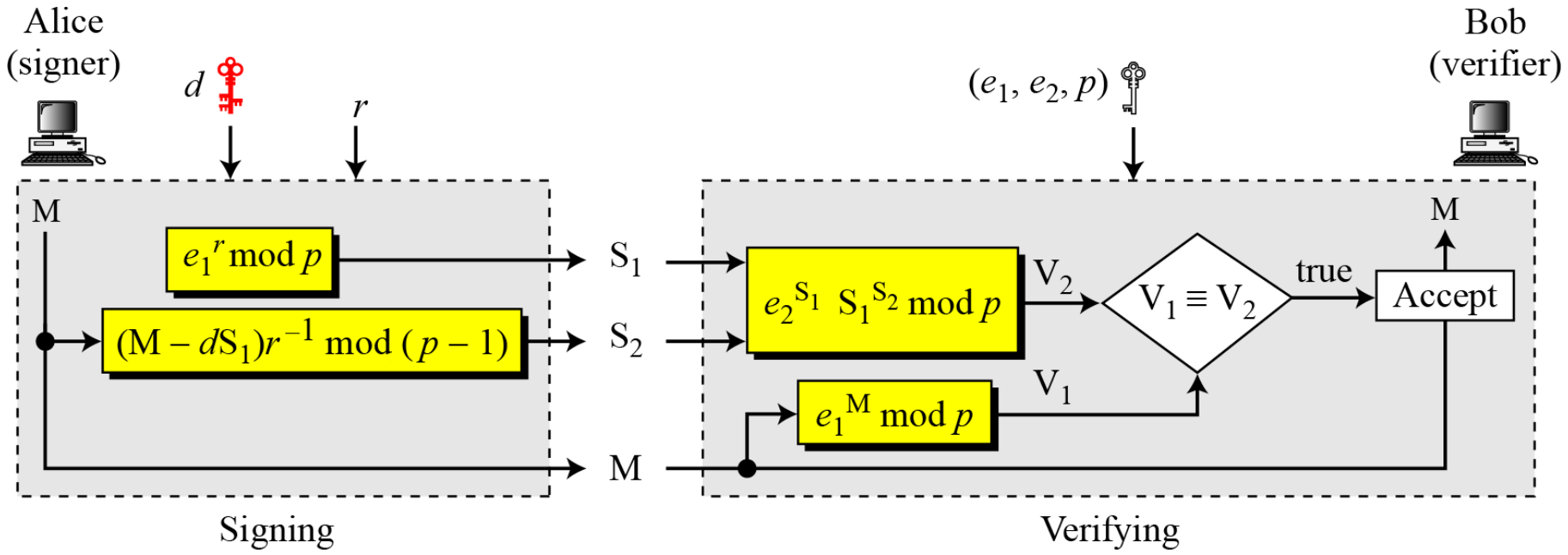
Note

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 *ElGamal digital signature scheme*

M: Message
S_1, S_2: Signatures
V_1, V_2: Verifications
r: Random secret
d: Alice's private key
(e_1, e_2, p): Alice's public key



Example 13.2

Here is a trivial example. Alice chooses $p = 3119$, $e_1 = 2$, $_d = 127$ and calculates $e_2 = 2^{127} \bmod 3119 = 1702$. She also chooses r to be 307. She announces e*1*, e*2*, and p publicly; she keeps d secret. The following shows how Alice can sign a message.

$$M = 320$$

$$S_1 = e_1{}^r = 2^{307} = 2083 \bmod 3119$$

$$S_2 = (M - d \times S_1) \times r^{-1} = (320 - 127 \times 2083) \times 307^{-1} = 2105 \bmod 3118$$

Alice sends M, S_1, and S_2 to Bob. Bob uses the public key to calculate V_1 and V_2.

$$V_1 = e_1{}^M = 2^{320} = 3006 \bmod 3119$$

$$V_2 = d^{S_1} \times S_1{}^{S_2} = 1702^{2083} \times 2083^{2105} = 3006 \bmod 3119$$

Example 13.3

Now imagine that Alice wants to send another message, M = 3000, to Ted. She chooses a new *r*, 107. Alice sends M, S_1, and S_2 to Ted. Ted uses the public keys to calculate V_1 and V_2.

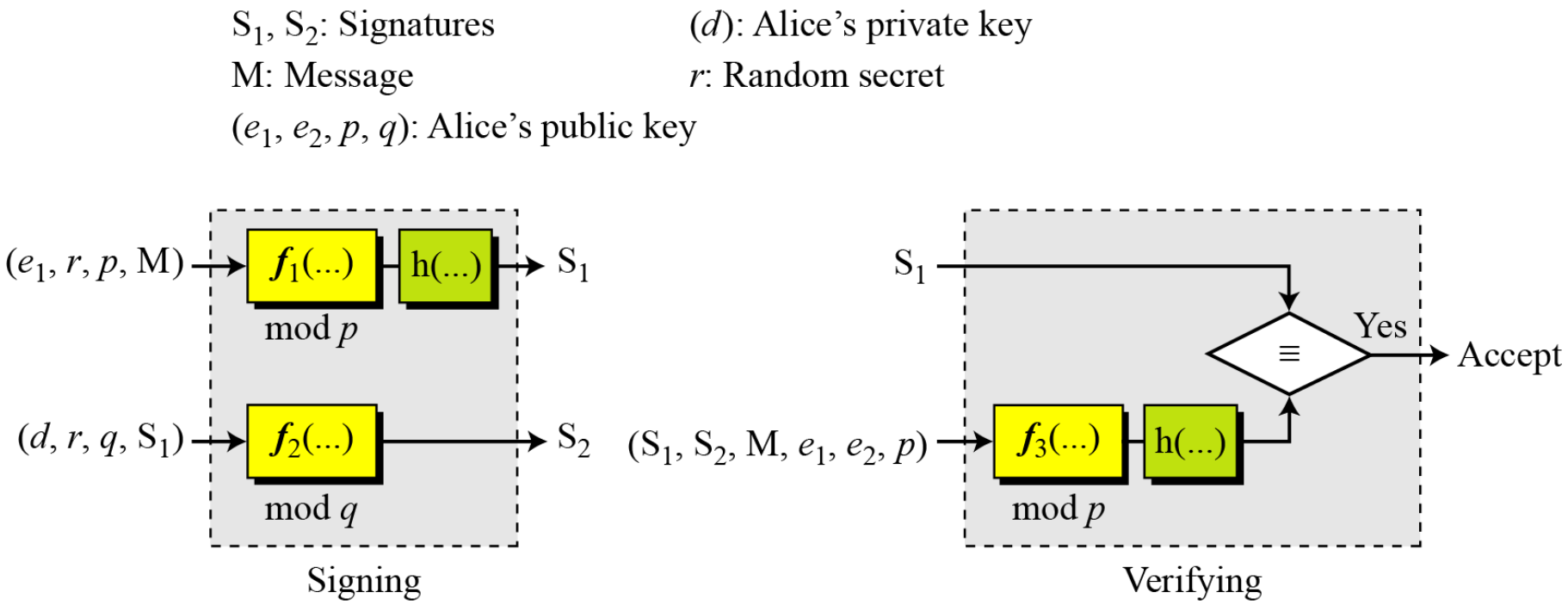
$$M = 3000$$
$$S_1 = e_1{}^r = 2^{107} = 2732 \bmod 3119$$
$$S_2 = (M - d \times S_1)\, r^{-1} = (3000 - 127 \times 2083) \times 107^{-1} = 2526 \bmod 3118$$

$$V_1 = e_1{}^M = 2^{3000} = 704 \bmod 3119$$
$$V_2 = d^{S_1} \times S_1{}^S = 1702^{2732} \times 2083^{2526} = 704 \bmod 3119$$

Schnorr Digital Signature

- Scheme is based on discrete logarithms
- Minimizes the message-dependent amount of computation required to generate a signature
  - Multiplying a $2n$-bit integer with an n-bit integer
- Main work can be done during the idle time of the processor
- Based on using a prime modulus p, with $p - 1$ having a prime factor q of appropriate size
  - Typically p is a 1024-bit number, and q is a 160-bit number

Figure 13.11 *General idea behind the Schnorr digital signature scheme*



Key Generation

1) Alice selects a prime p, which is usually 1024 bits in length.
2) Alice selects another prime q.
3) Alice chooses e_1 to be the qth root of 1 modulo p.
4) Alice chooses an integer, d, as her private key.
5) Alice calculates $e_2 = e_1^d \bmod p$.
6) Alice's public key is (e_1, e_2, p, q); her private key is (d).

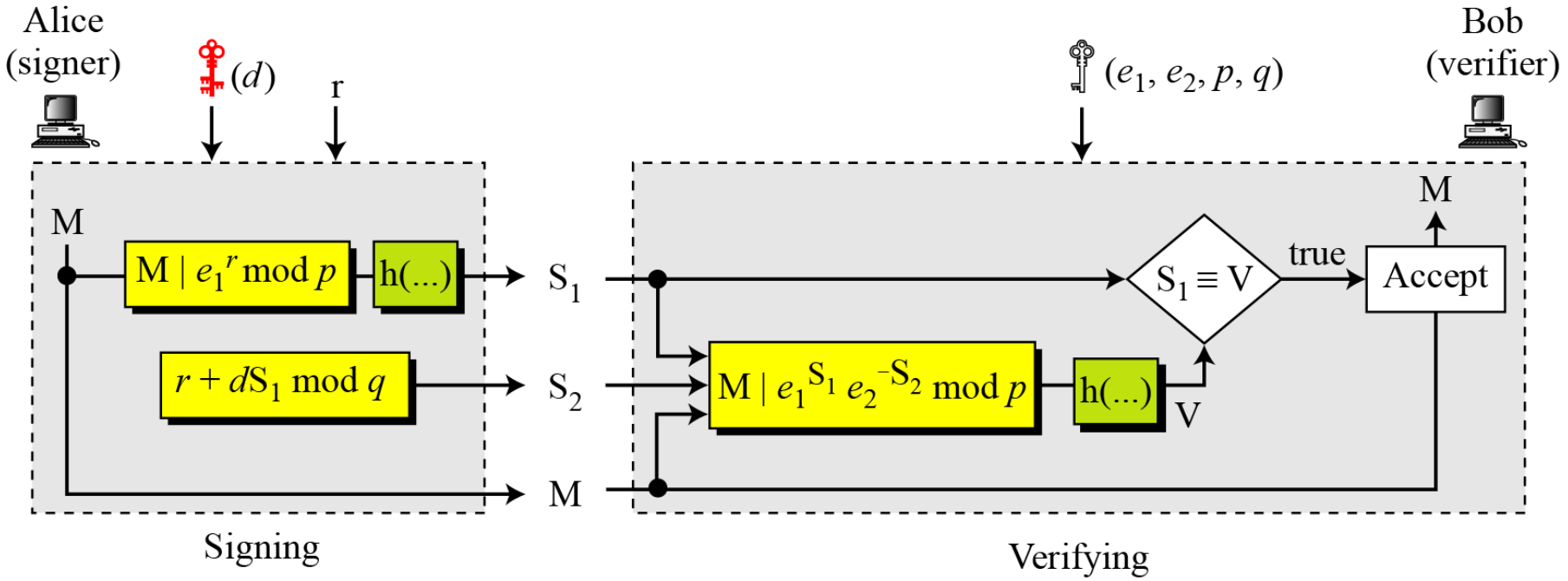
Note

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

Signing and Verifying

Figure 13.12 *Schnorr digital signature scheme*

M: Message
S_1, S_2: Signatures
V: Verification
r: Random secret
(d): Alice's private key
(e_1, e_2, p, q): Alice's public key
| : Concatenation
h(...): Hash algorithm



Signing

1. Alice chooses a random number r.
2. Alice calculates $S_1 = h(M|e_1^r \bmod p)$.
3. Alice calculates $S_2 = r + d \times S_1 \bmod q$.
4. Alice sends M, S_1, and S_2.

Verifying Message

1. Bob calculates $V = h\,(M \mid e_1^{S2}\, e_2^{-S1} \bmod p)$.
2. If S_1 is congruent to V modulo p, the message is accepted;

Example 13.4

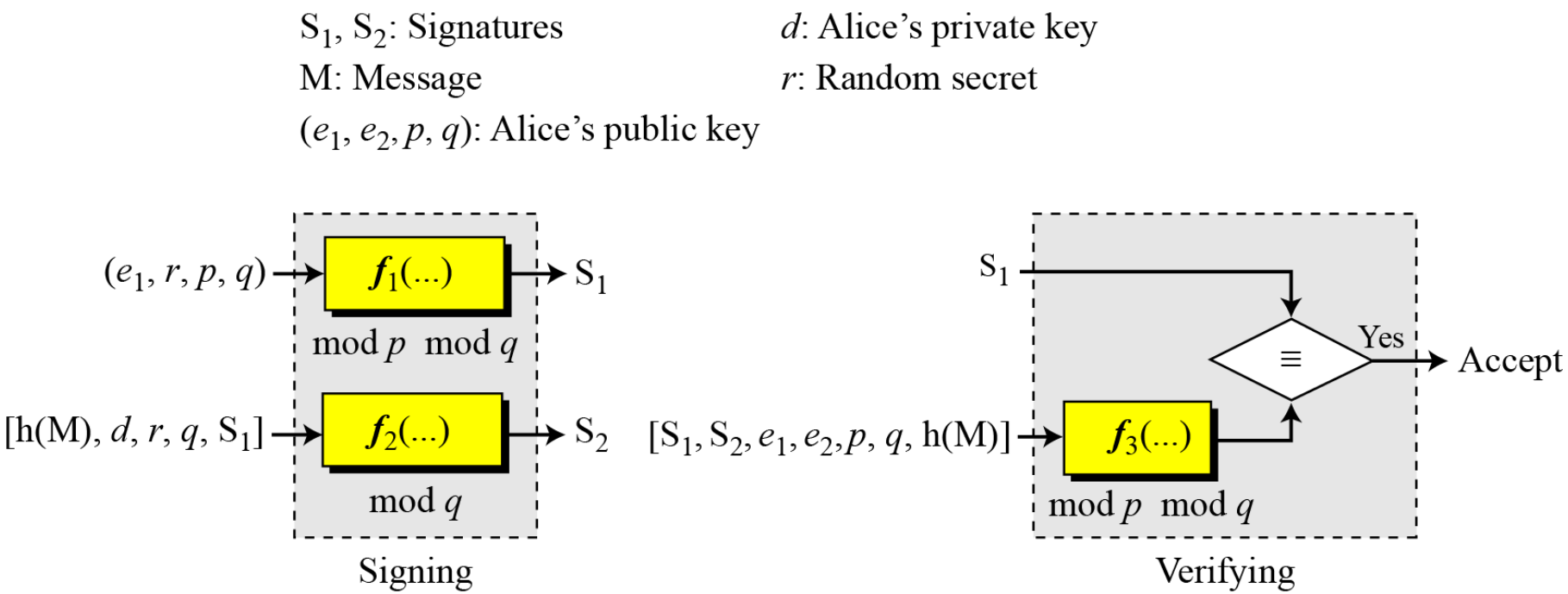
Here is a trivial example. Suppose we choose $q = 103$ and $p = 2267$. Note that $p = 22 \times q + 1$. We choose $e_0 = 2$, which is a primitive in Z_{2267}^*. Then $(p - 1) / q = 22$, so we have $e_1 = 2^{22} \bmod 2267 = 354$. We choose $d = 30$, so $e_2 = 354^{30} \bmod 2267 = 1206$. Alice's private key is now (d); her public key is (e_1, e_2, p, q).

Alice wants to send a message M. She chooses $r = 11$ and calculates $e_2{}^r = 354^{11} = 630 \bmod 2267$. Assume that the message is 1000 and concatenation means 1000630. Also assume that the hash of this value gives the digest h(1000630) = 200. This means S1 = 200. Alice calculates $S2 = r + d \times S_1 \bmod q = 11 + 1026 \times 200 \bmod 103 = 35$. Alice sends the message M =1000, $S_1 = 200$, and $S_2 = 35$. The verification is left as an exercise.

NIST Digital Signature Algorithm

- Published by NIST as Federal Information Processing Standard FIPS 186
- Makes use of the Secure Hash Algorithm (SHA)
- The latest version, FIPS 186-3, also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography

Figure 13.13 *General idea behind DSS scheme*



Key Generation.

1) *Alice chooses primes p and q.*

2) *Alice uses* $<Z_p^*, \times>$ *and* $<Z_q^*, \times>$.

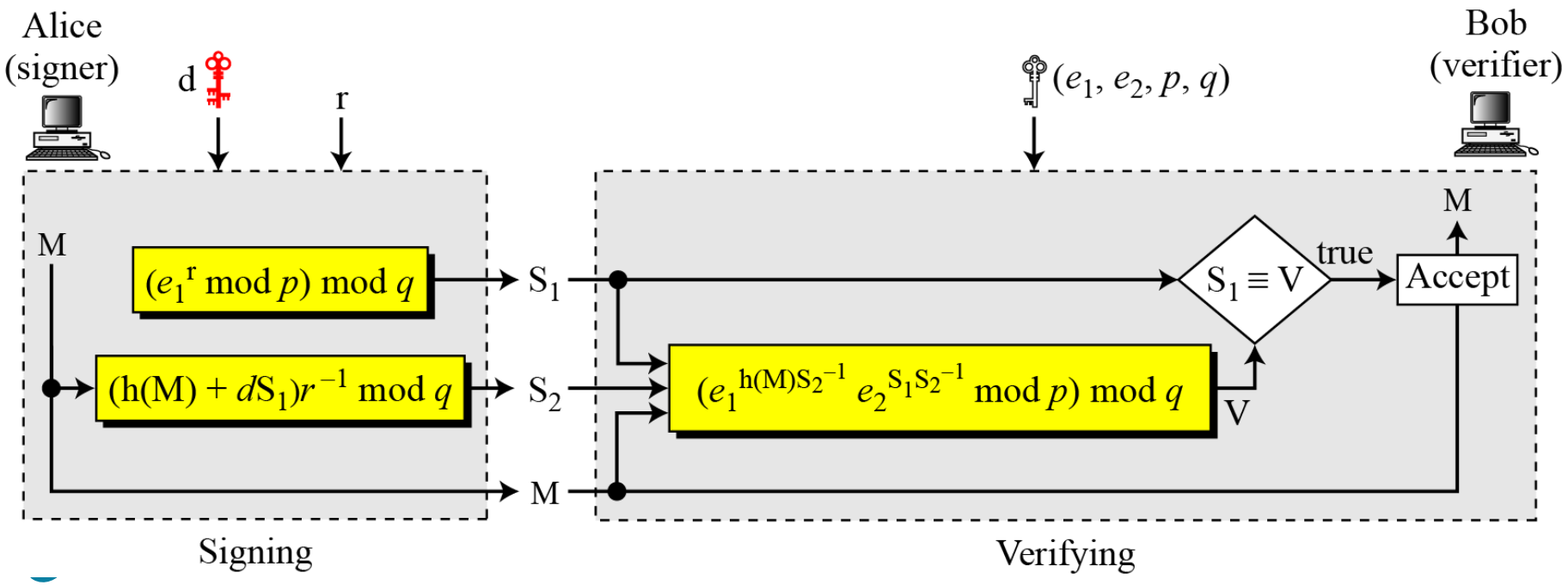
3) *Alice creates* e_1 *to be the qth root of 1 modulo p.*

4) *Alice chooses d and calculates* $e_2 = e_1^d$.

5) *Alice's public key is* (e_1, e_2, p, q)*; her private key is* (d).

Verifying and Signing

Figure 13.14 *DSS scheme*



Example 13.5

Alice chooses $q = 101$ and p = 8081. Alice *selects* $e_0 = 3$ and calculates $e^1 = e_0^{(p-1)/q} \bmod p = 6968$. Alice chooses $d = 61$ as the private key and calculates $e_2 = e_1^{d} \bmod p = 2038$. Now Alice can send a message to Bob. Assume that h(M) = 5000 and Alice chooses $r = 61$:

$$\text{h(M)} = 5000 \qquad r = 61$$
$$S_1 = (e_1^{\,r} \bmod p) \bmod q = 54$$
$$S_2 = ((\text{h(M)} + d\,S_1)\,r^{-1}) \bmod q = 40$$

Alice sends M, S_1, and S_2 to Bob. Bob uses the public keys to calculate V.

$$S_2^{\,-1} = 48 \bmod 101$$
$$V = [(6968^{5000 \times 48} \times 2038^{54 \times 48}) \bmod 8081] \bmod 101 = 54$$

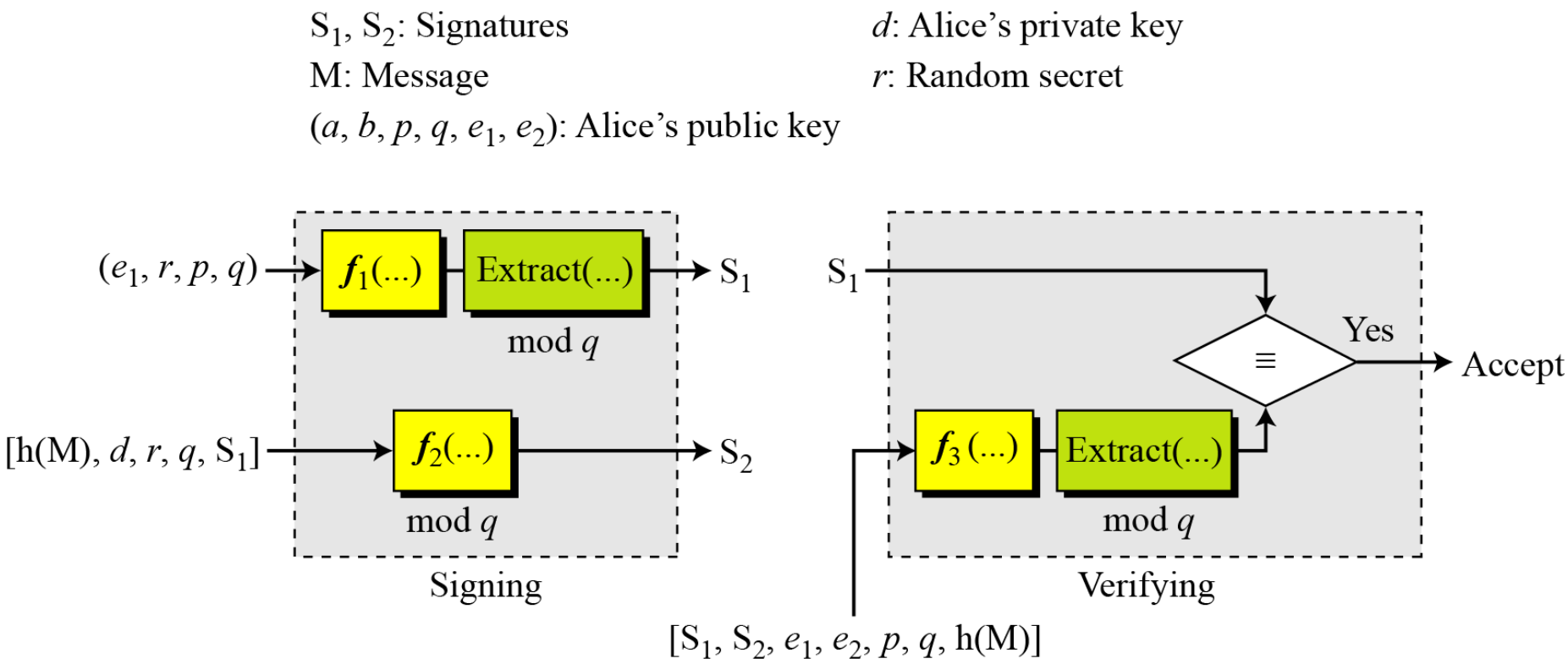
DSS Versus RSA

Computation of DSS signatures is faster than computation of RSA signatures when using the same p.

DSS Versus ElGamal

DSS signatures are smaller than ElGamal signatures because q is smaller than p.

Figure 13.15 *General idea behind the ECDSS scheme*



Key Generation

Key generation follows these steps:

1) *Alice chooses an elliptic curve $E_p(a, b)$.*

2) *Alice chooses another prime q the private key d.*

3) *Alice chooses $e_1(..., ...)$, a point on the curve.*

4) *Alice calculates $e_2(..., ...) = d \times e_1(..., ...)$.*

5) *Alice's public key is (a, b, p, q, e1, e2); her private key is d.*

Signing and Verifying

Figure 13.16 *The ECDSS scheme*

M: Message
S_1, S_2: Signatures
V: Verification

r: Random secret
d: Alice's private key
(a, b, p, q, e_1, e_2): Alice's public key

P(*u*, *v*), T(*x*, *y*): Points on the curve
h(M): Message digest
A, B: Intermediate results

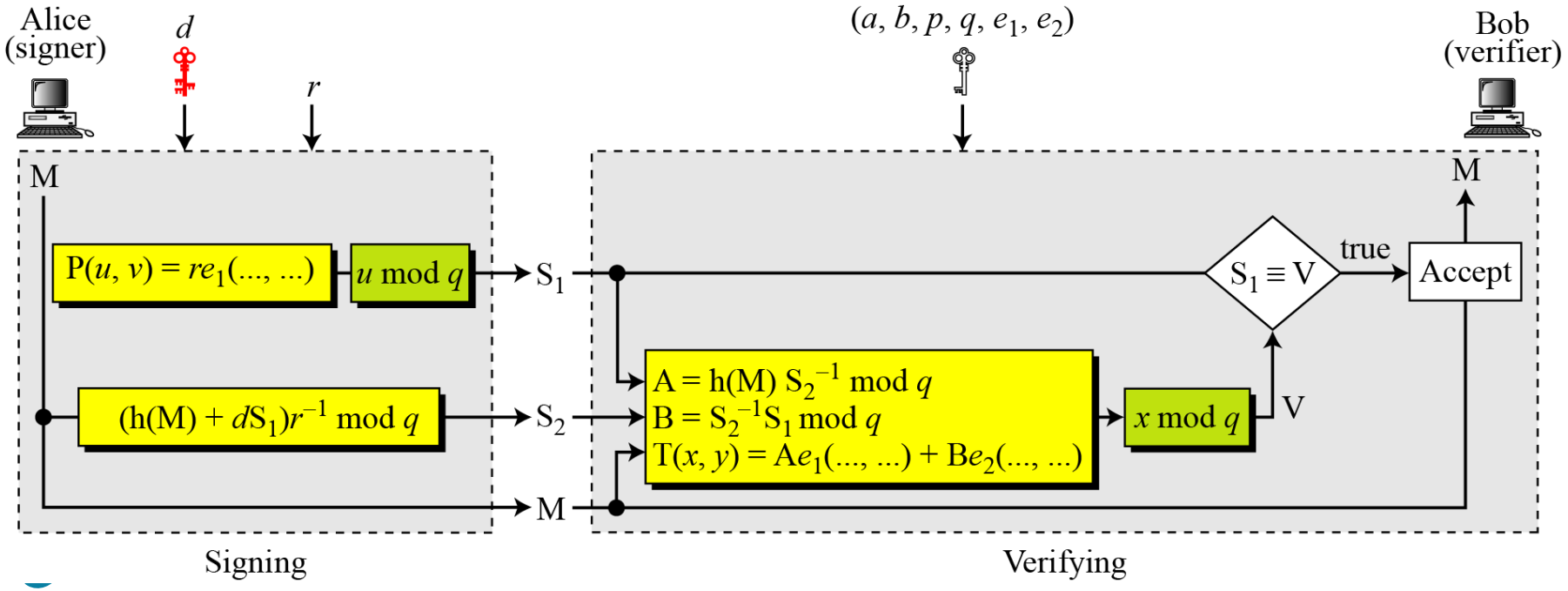


Figure 13.2 Two Approaches to Digital Signatures

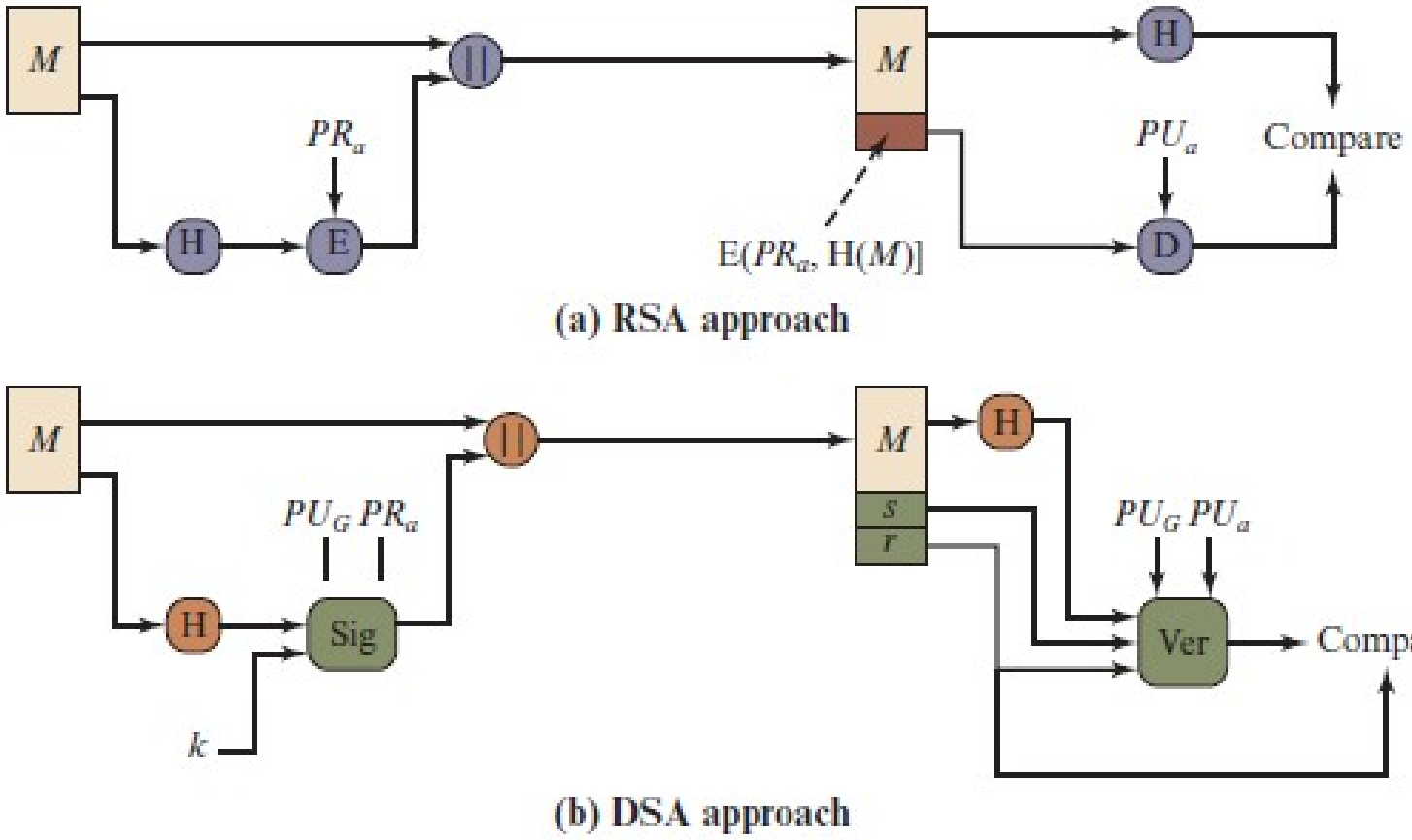


Figure 13.3 The Digital Signature Algorithm (DSA)

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 L 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 $= h(p-1)/q$ is an exponent mod 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

$w = (s')^{-1} \bmod q$

$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u1}y^{u2}) \bmod p] \bmod q$

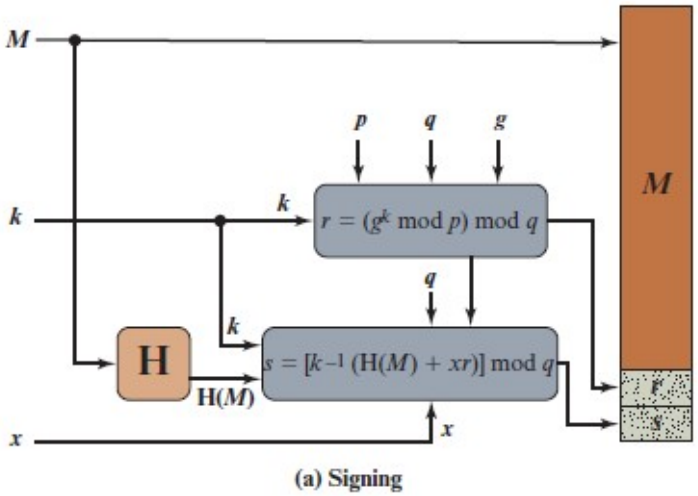
TEST: $v = r'$

M = message to be signed

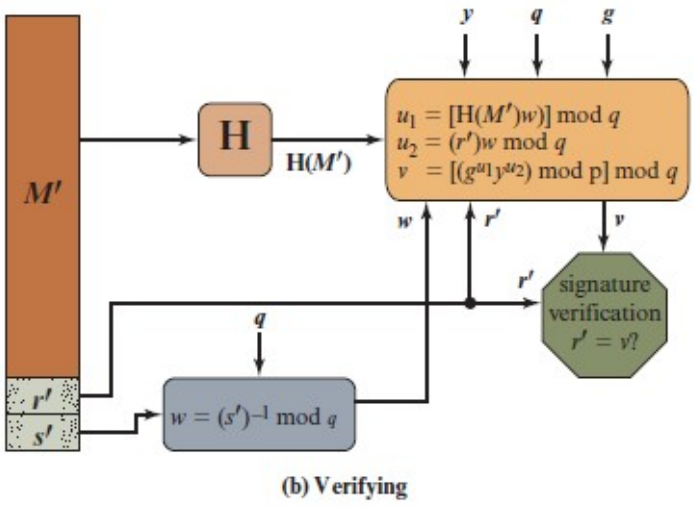
$H(M)$ = hash of M using SHA-1

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

Figure 13.4 DS A Signing and Verifying



(a) Signing

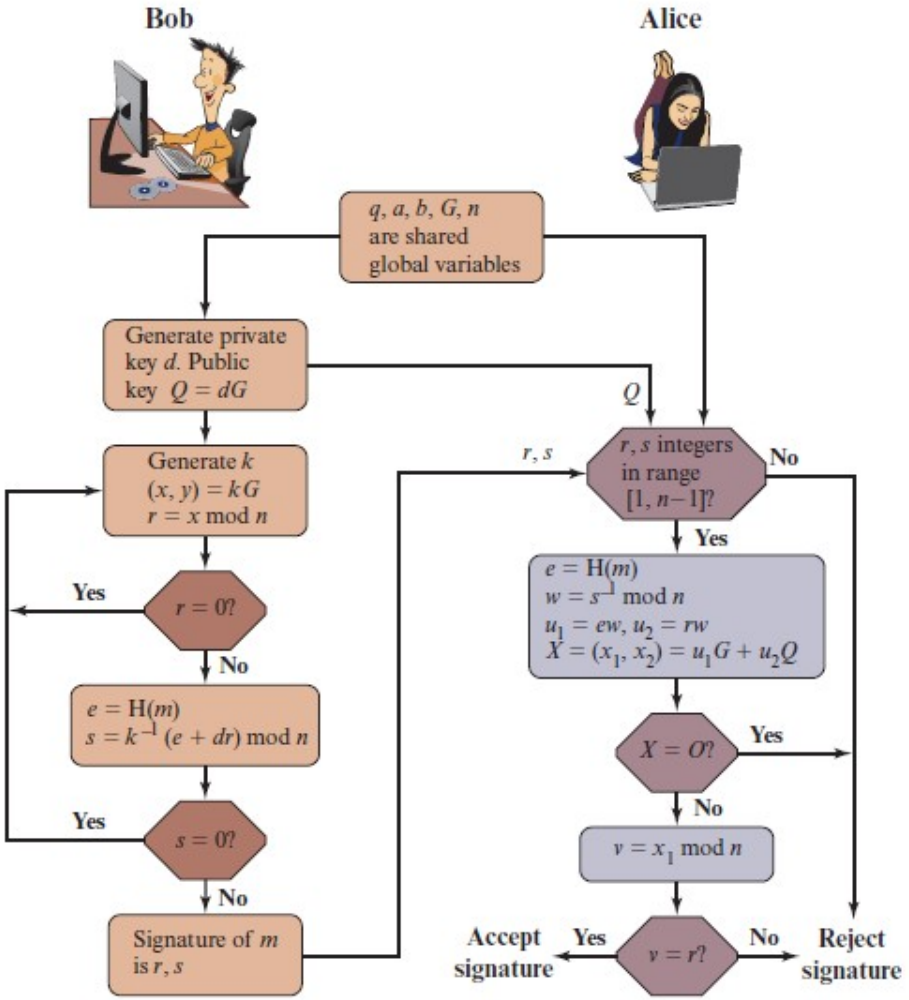


(b) Verifying

Elliptic Curve Digital Signature Algorithm (ECDSA)

- Four elements are involved:
  - All those participating in the digital signature scheme use the same global domain parameters, which define an elliptic curve and a point of origin on the curve
  - A signer must first generate a public, private key pair
  - A hash value is generated for the message to be signed; using the private key, the domain parameters, and the hash value, a signature is generated
  - To verify the signature, the verifier uses as input the signer's public key, the domain parameters, and the integer *s;* the output is a value *v* that is compared to *r* ; the signature is verified if the *v* = *r*

Figure 13.5 ECDS A Signing and Verifying



RSA-PSS

- RSA Probabilistic Signature Scheme
- Included in the 2009 version of FIPS 186
- Latest of the RSA schemes and the one that RSA Laboratories recommends as the most secure of the RSA schemes
- For all schemes developed prior to PSS it has not been possible to develop a mathematical proof that the signature scheme is as secure as the underlying RSA encryption/decryption primitive
- The PSS approach was first proposed by Bellare and Rogaway
- This approach, unlike the other RSA-based schemes, introduces a randomization process that enables the security of the method to be shown to be closely related to the security of the RSA algorithm itself

Mask Generation Function (MGF)

- Typically based on a secure cryptographic hash function such as SHA-1
  - Is intended to be a cryptographically secure way of generating a message digest, or hash, of variable length based on an underlying cryptographic hash function that produces a fixed-length output

Figure 13.6 RSA-PSS Encoding

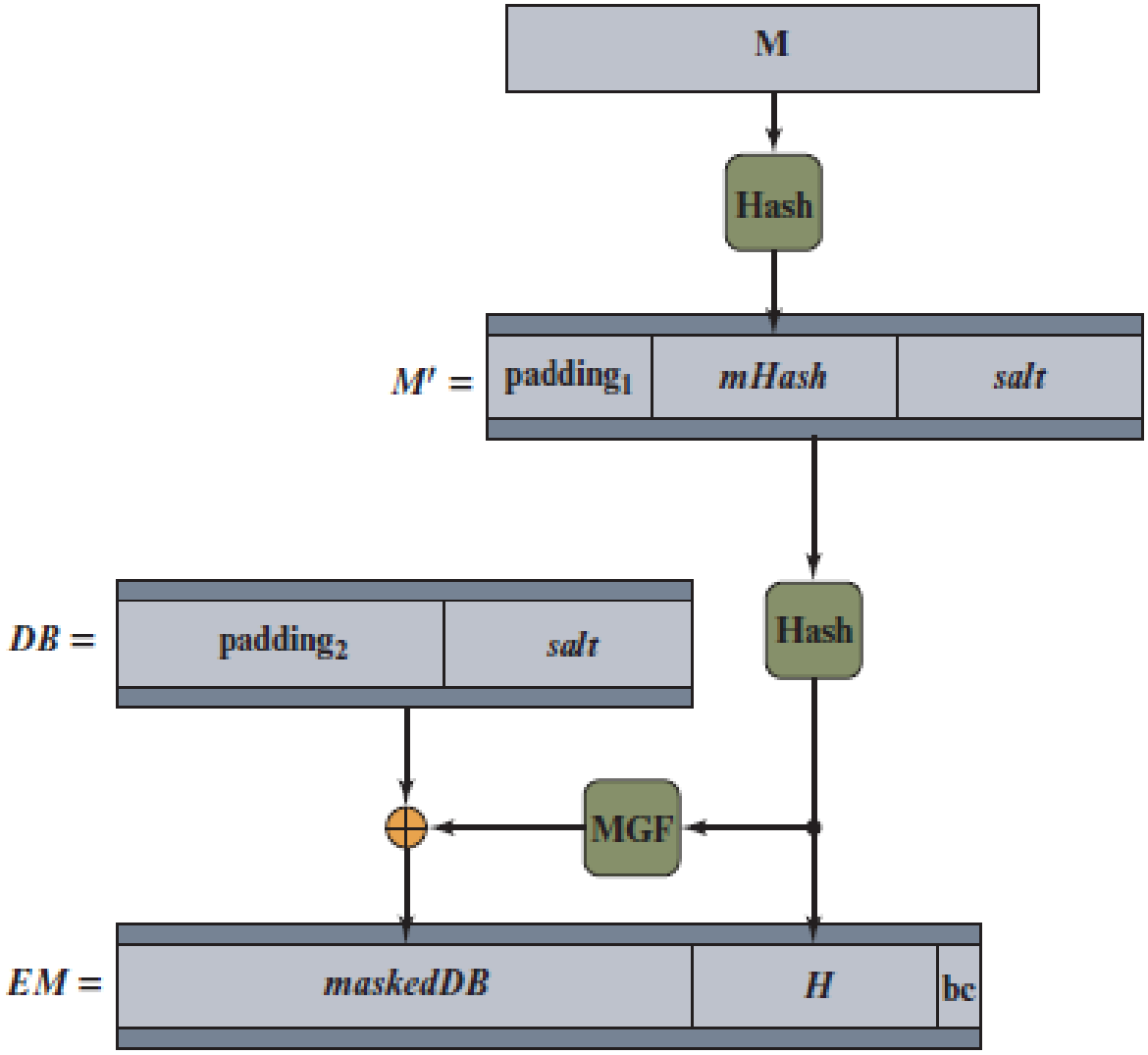
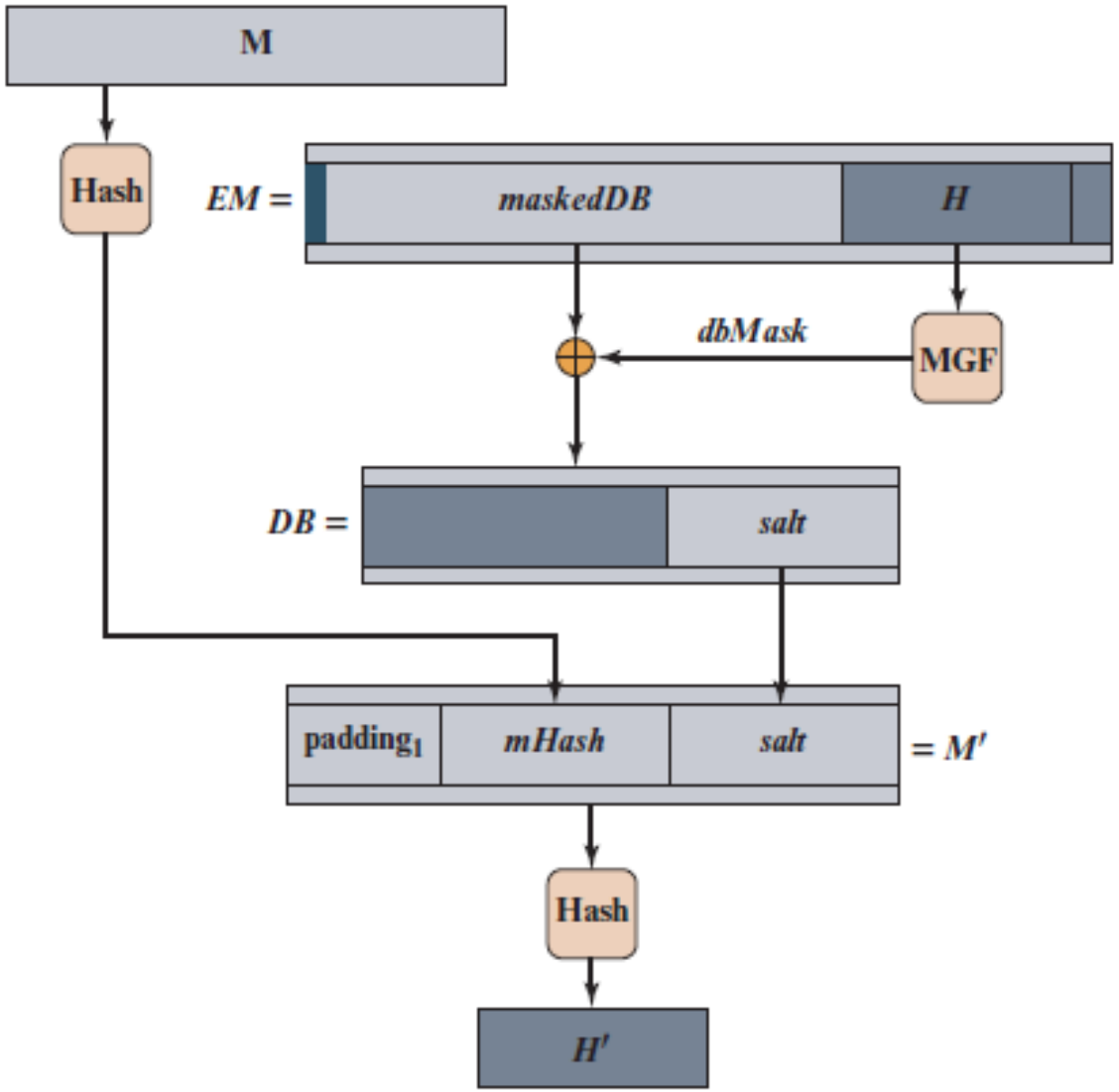


Figure 13.7 RSA-PSS EM Verification



Summary

- Present an overview of the digital signature process
- Understand the ElGamal digital signature scheme
- Understand the Schnorr digital signature scheme
- Understand the NIST digital signature scheme
- Compare and contrast the NIST digital signature scheme with the ElGamal and Schnorr digital signature schemes
- Understand the elliptic curve digital signature scheme
- Understand the RSA-PSS digital signature scheme

Copyright