



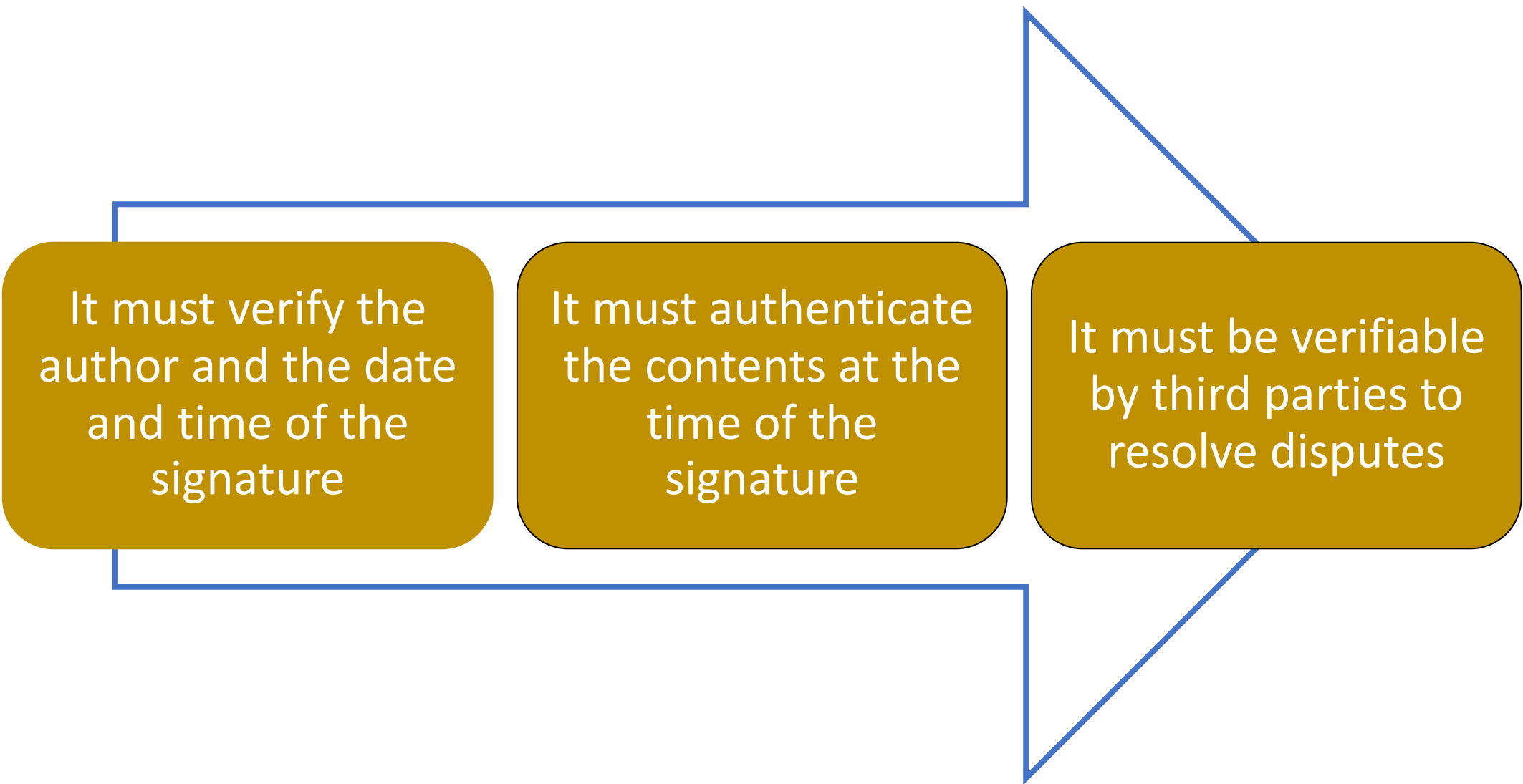
Chapter 13

Digital Signatures

Digital Signatures

- have looked at message authentication
 - but does not address issues of **lack of trust**
- digital signatures provide the ability to:
 - **verify** author, date & time of signature
 - **authenticate** message contents
 - be verified **by third parties** to resolve disputes
- hence include authentication function with additional capabilities

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

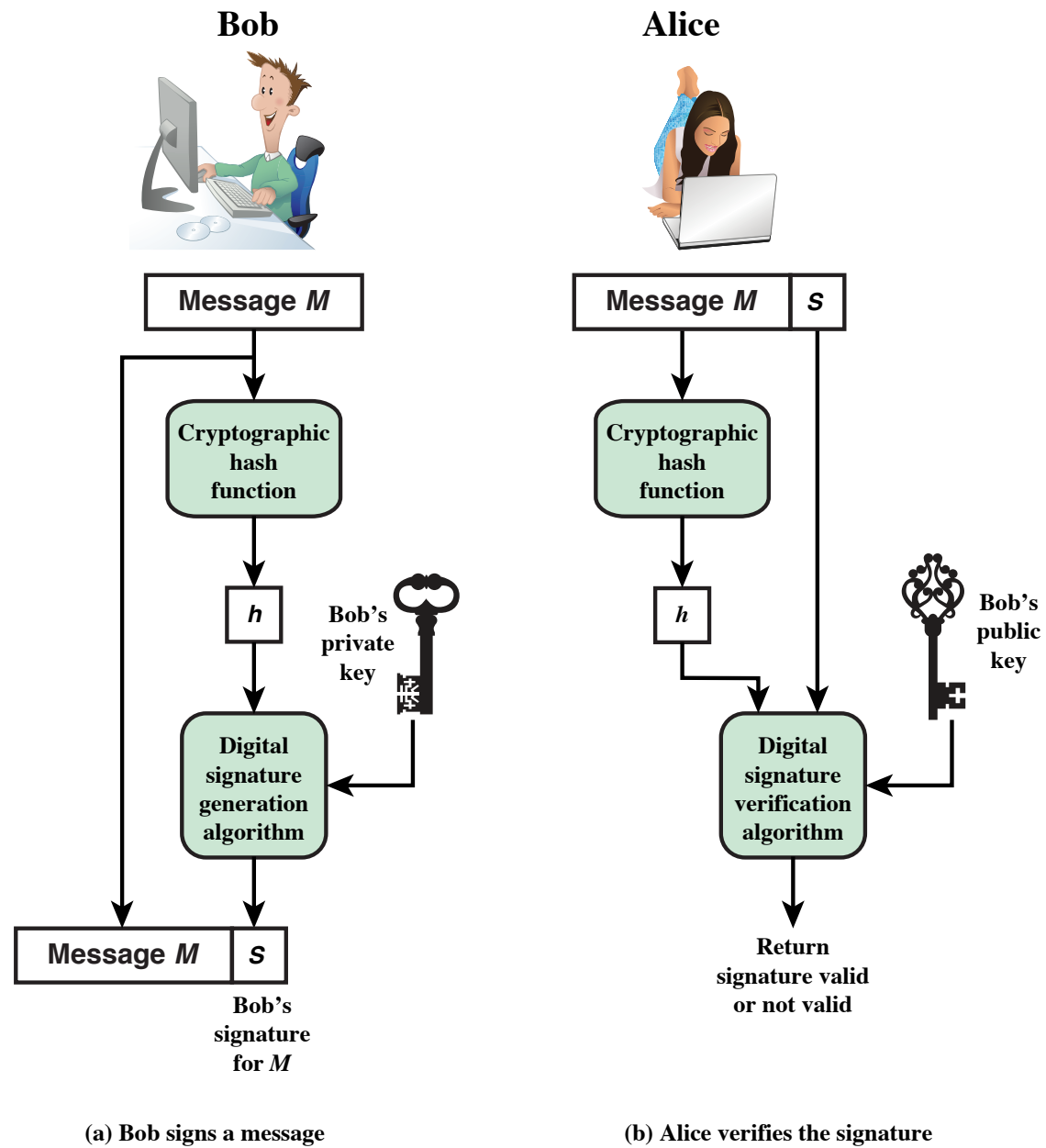
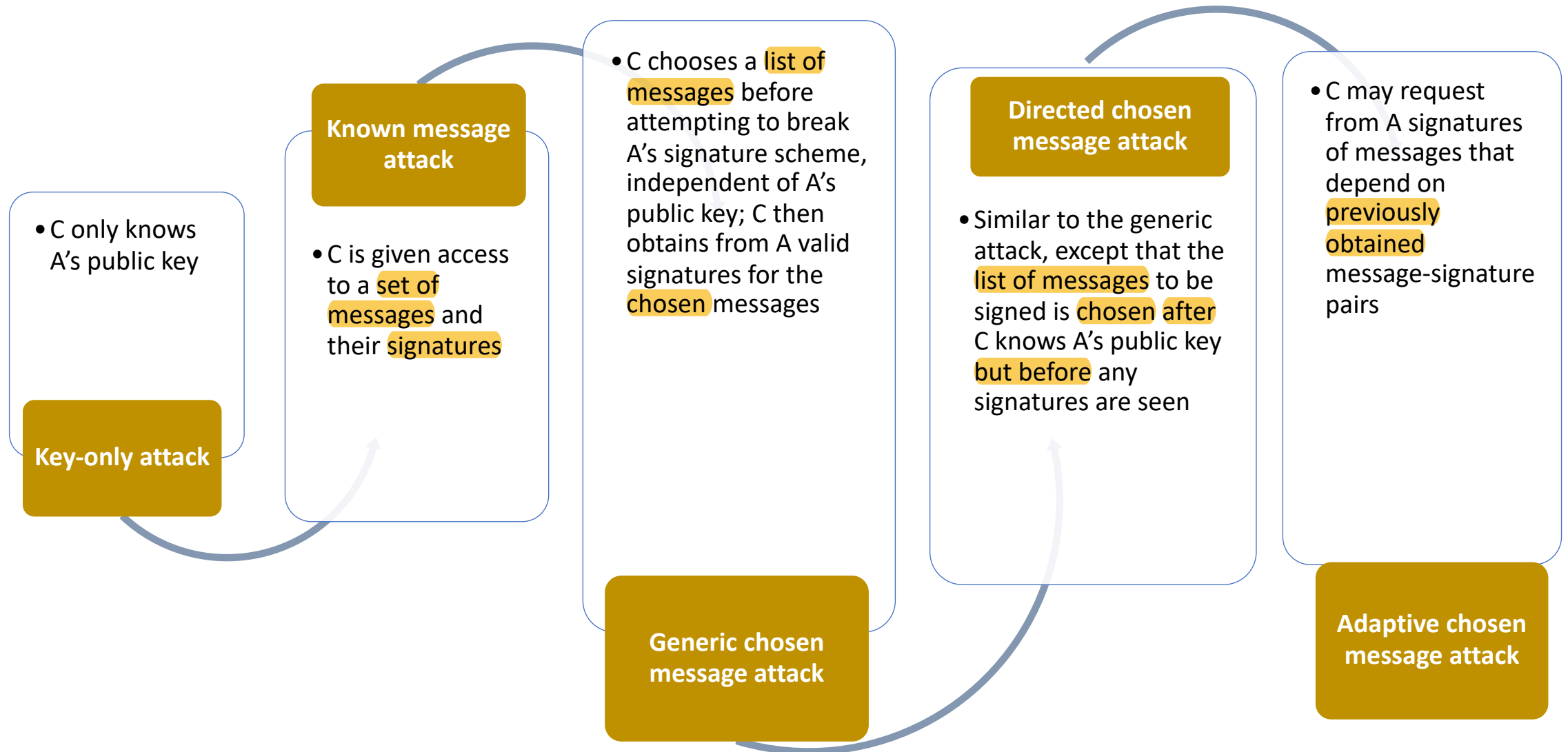
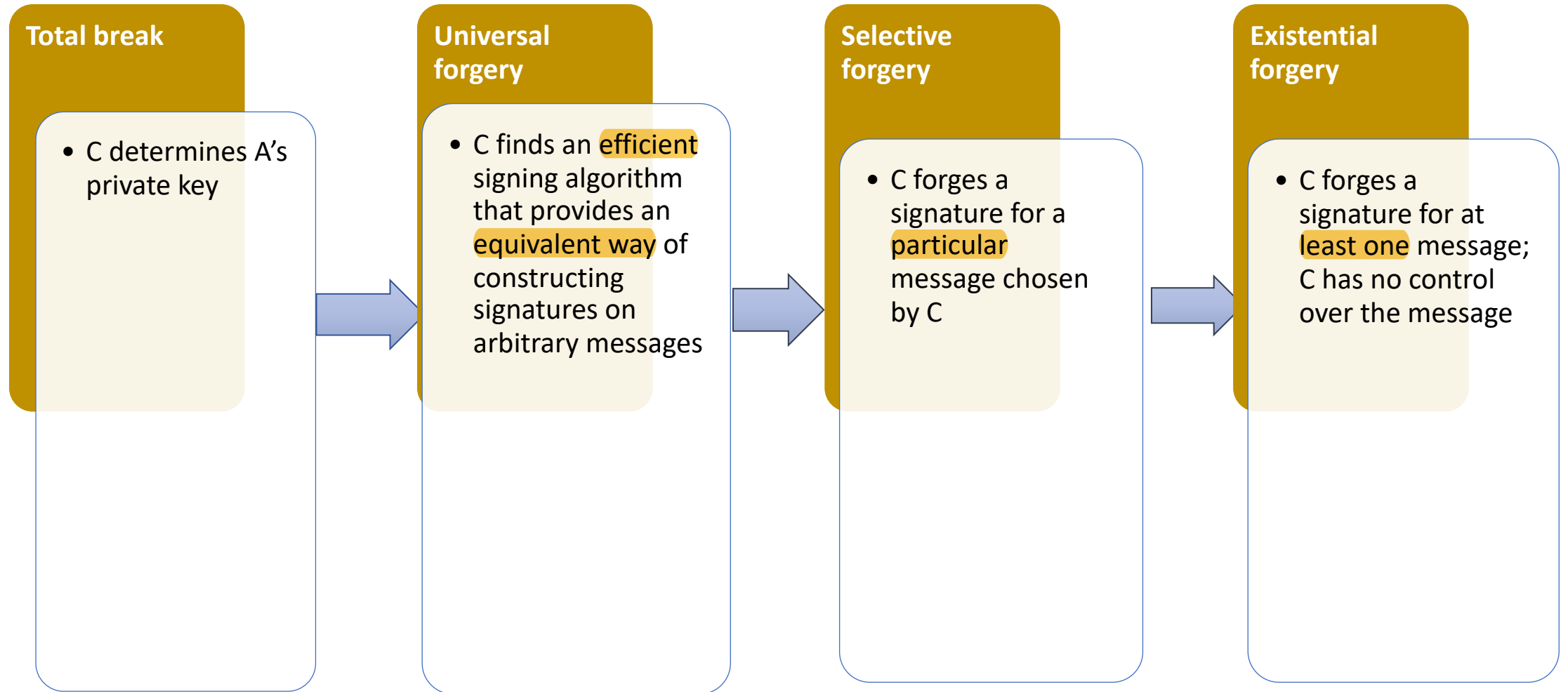


Figure 13.1 Simplified Depiction of Essential Elements of Digital Signature Process

Attacks



Forgeries



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

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: $y_A = a^{x_A} \bmod q$

ElGamal Digital Signature

- Alice signs a message M to Bob by computing
 - the hash $m = H(M)$, $0 \leq m \leq (q-1)$
 - chose random integer K with $1 \leq K \leq (q-1)$ and $\gcd(K, q-1) = 1$
 - compute temporary key: $S_1 = a^K \bmod q$
 - compute K^{-1} the inverse of $K \bmod (q-1)$
 - compute the value: $S_2 = K^{-1}(m - x_A S_1) \bmod (q-1)$
 - signature is: (S_1, S_2)
- any user B can verify the signature by computing
 - $V_1 = a^m \bmod q$
 - $V_2 = Y_A^{S_1} S_1^{S_2} \bmod q$
 - signature is valid if $V_1 = V_2$

Proof: ElGamal Digital Signature

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

$$\alpha^m \bmod q = (Y_A)^{S_1} (S_1)^{S_2} \bmod q$$

$$\alpha^m \bmod q = \alpha^{X_A S_1} \alpha^{K S_2} \bmod q$$

$$\alpha^{m - X_A S_1} \bmod q = \alpha^{K S_2} \bmod q$$

$$m - X_A S_1 \equiv K S_2 \bmod (q - 1)$$

$$m - X_A S_1 \equiv K K^{-1} (m - X_A S_1) \bmod (q - 1)$$

assume $V_1 = V_2$

substituting for Y_A and S_1

rearranging terms

property of primitive roots

substituting for S_2

ElGamal Signature Example

- use field $GF(19)$ $q=19$ and $a=10$
- Alice computes her key:
 - A chooses $x_A=16$ & computes $y_A=10^{16} \bmod 19 = 4$
- Alice signs message with hash $m=14$ as $(3, 4)$:
 - choosing random $K=5$ which has $\gcd(18, 5)=1$
 - computing $S_1 = 10^5 \bmod 19 = 3$
 - finding $K^{-1} \bmod (q-1) = 5^{-1} \bmod 18 = 11$
 - computing $S_2 = 11(14-16.3) \bmod 18 = 4$
- any user B can verify the signature by computing
 - $V_1 = 10^{14} \bmod 19 = 16$
 - $V_2 = 4^3 \cdot 3^4 = 5184 = 16 \bmod 19$
 - since $16 = 16$ signature is valid

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

Schnorr Key Setup

- choose suitable primes p , q
- choose a such that $a^q = 1 \pmod p$
- (a, p, q) are global parameters for all
- each user (eg. A) generates a key
 - chooses a secret key (number): $0 < s < q$
 - compute their **public key**: $v = a^{-s} \pmod q$

Schnorr Signature

- user signs message by
 - choosing **random** r with $0 < r < q$ and computing $x = a^r \bmod p$
 - concatenate message with x and hash result to computing:
 $e = H(M \parallel x)$
 - computing: $y = (r + se) \bmod q$
 - **signature** is pair (e, y)
- any other user can verify the signature as follows:
 - computing: $x' = a^y v^e \bmod p$
 - **verifying** that: $e = H(M \parallel x')$

Proof: Schnorr Signature

To see that the verification works, observe that

$$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod{p}$$

Hence, $H(M \| x') = H(M \| x)$.

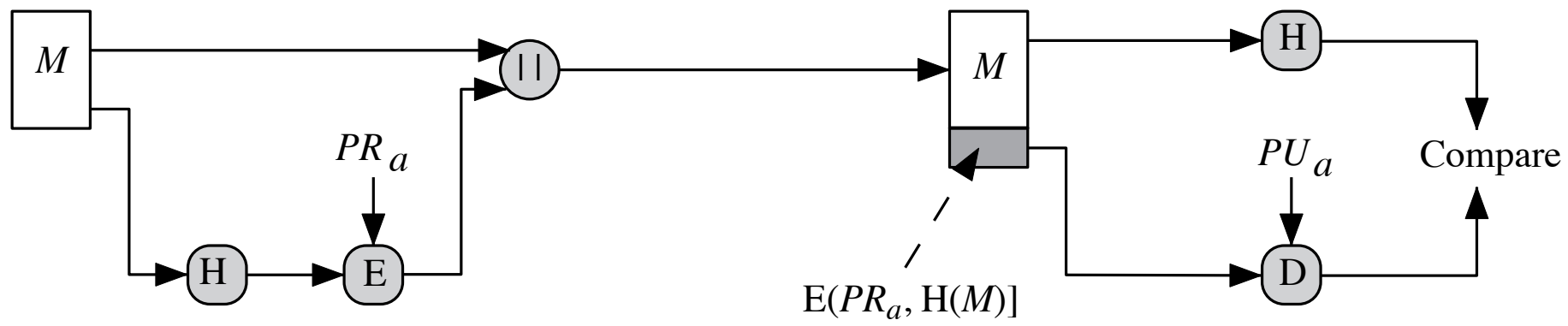
Digital Signature Standard (DSS)

- US Govt approved signature scheme
- designed by NIST & NSA in early 90's
- published as FIPS-186 in 1991
- revised in 1993, 1996 & then 2000
 - uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
 - FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants
 - DSA is digital signature only unlike RSA
 - is a public-key technique

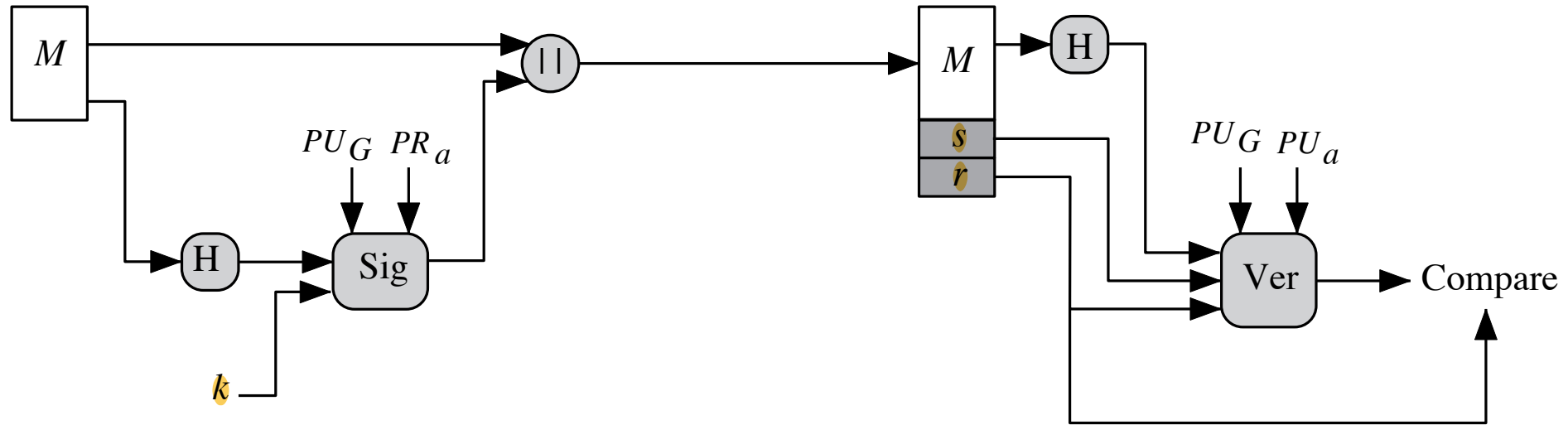
NIST Digital Signature Algorithm (DSA)

- 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





(a) RSA Approach



(b) DSA Approach

Figure 13.2 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 $2^{N-1} < q < 2^N$
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

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

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

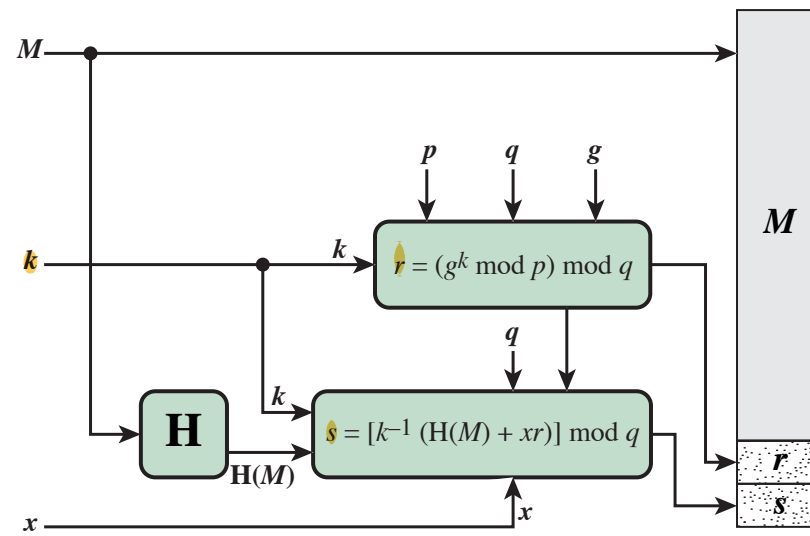
$u_2 = (r')w \bmod q$

$v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$

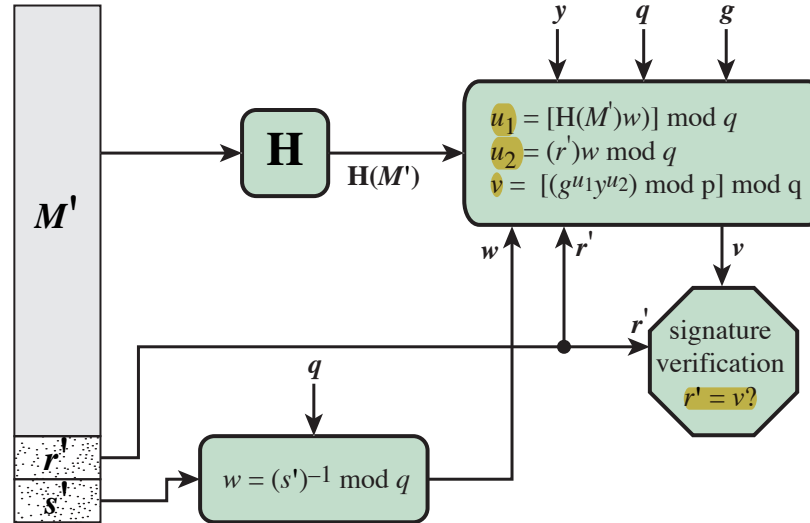
TEST: $v = r'$

M = message to be signed
 $H(M)$ = hash of M using SHA-1
 M', r', s' = received versions of M, r, s

Figure 13.3 The Digital Signature Algorithm (DSS)



(a) Signing



(b) Verifying

Figure 13.4 DSS Signing and Verifying

Digital Signature Algorithm (DSA)

- creates a 320 bit signature
- with 512-1024 bit security
- smaller and faster than RSA
- a digital signature scheme only
- security depends on difficulty of computing discrete logarithms
- variant of ElGamal & Schnorr schemes

DSA Key Generation

- have shared global public key values (p, q, g) :
 - choose 160-bit prime number q
 - choose a large prime p with $2^{L-1} < p < 2^L$
 - where $L = 512$ to 1024 bits and is a multiple of 64
 - such that q is a 160 bit prime divisor of $(p-1)$
 - choose $g = h^{(p-1)/q}$
 - where $1 < h < p-1$ and $h^{(p-1)/q} \bmod p > 1$
- users choose private & compute public key:
 - choose random private key: $x < q$
 - compute public key: $y = g^x \bmod p$

DSA Signature Creation

- to **sign** a message M the sender:
 - generates a random signature key k , $k < q$
 - nb. k must be random, be destroyed after use, and never be reused
- then computes signature pair:
$$r = (g^k \bmod p) \bmod q$$
$$s = [k^{-1} (H(M) + xr)] \bmod q$$
- sends signature (r, s) with message M

DSA Signature Verification

- having received M & signature (r, s)
- to **verify** a signature, recipient computes:

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

$$u1 = [H(M)w] \bmod q$$

$$u2 = (rw) \bmod q$$

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

- if $v=r$ then signature is verified
- see Appendix A for details of proof why

Elliptic Curve Digital Signature Algorithm (ECDSA)

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**

Four elements are involved:

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**

k is a pseudorandom number generated by the signer, resulting in different signatures from multiple signings of the same message with the same private key. A verifier **does not** and **need not** know the value of k

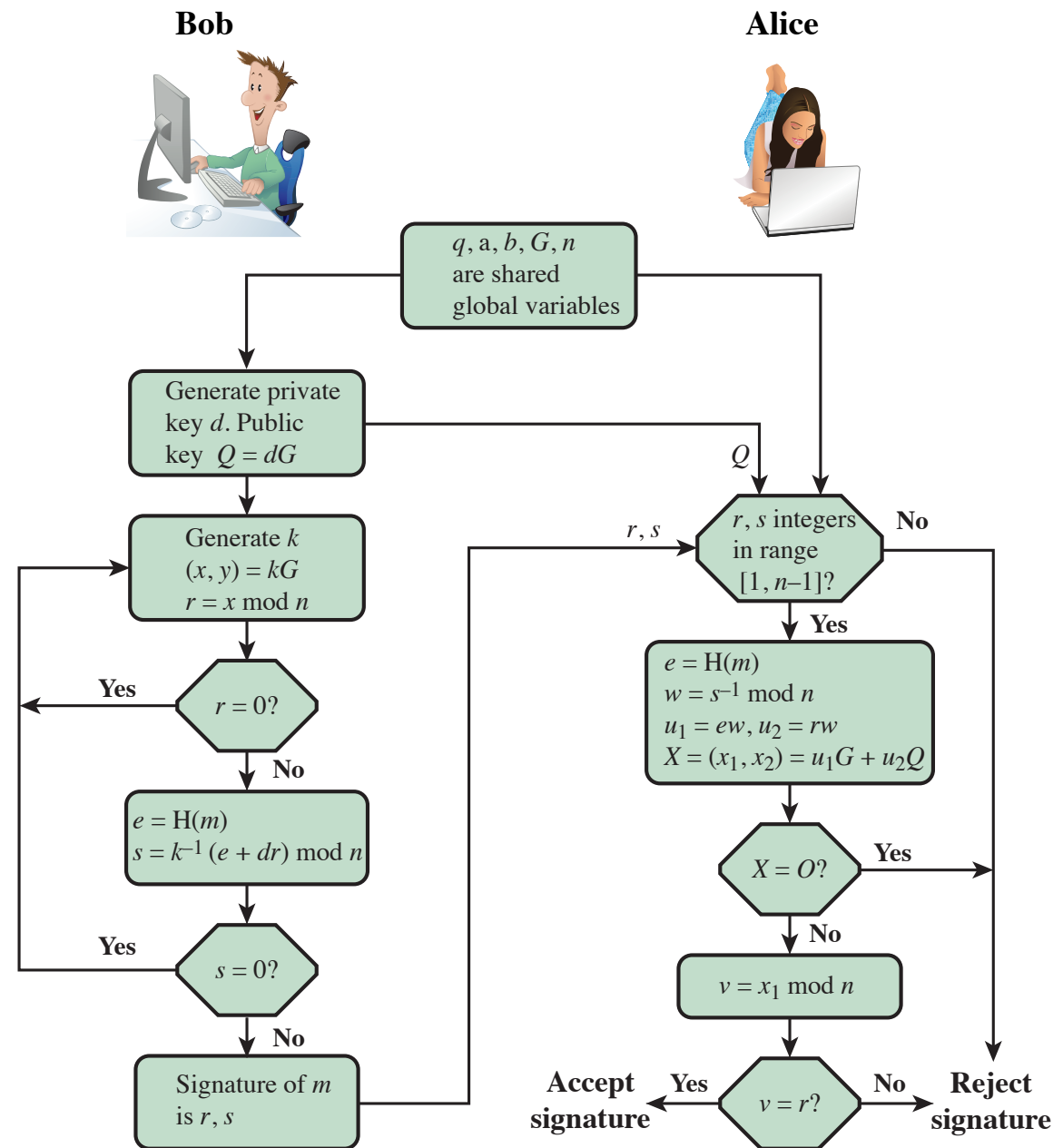


Figure 13.5 ECDSA Signing and Verifying

RSA-PSS

- **RSA Probabilistic Signature Scheme**

- First proposed by Bellare and Rogaway
 - 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
- 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
 - The objective is to make it **more difficult** for an adversary to find another message that maps to the same message digest as a given message or to find two messages that map to the same message digest.

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

padding₁ hexadecimal string 00 00 00 00 00 00 00 00; that is, a string of 64 zero bits.

padding₂ hexadecimal string of 00 octets with a length ($emLen - sLen - hLen - 2$) octets, followed by the hexadecimal octet with value 01.

bc the hexadecimal value BC.

Added security measure:

salt (a pseudorandom number)

changes with every use,

signing the same message twice using the same private key will yield two different signatures.

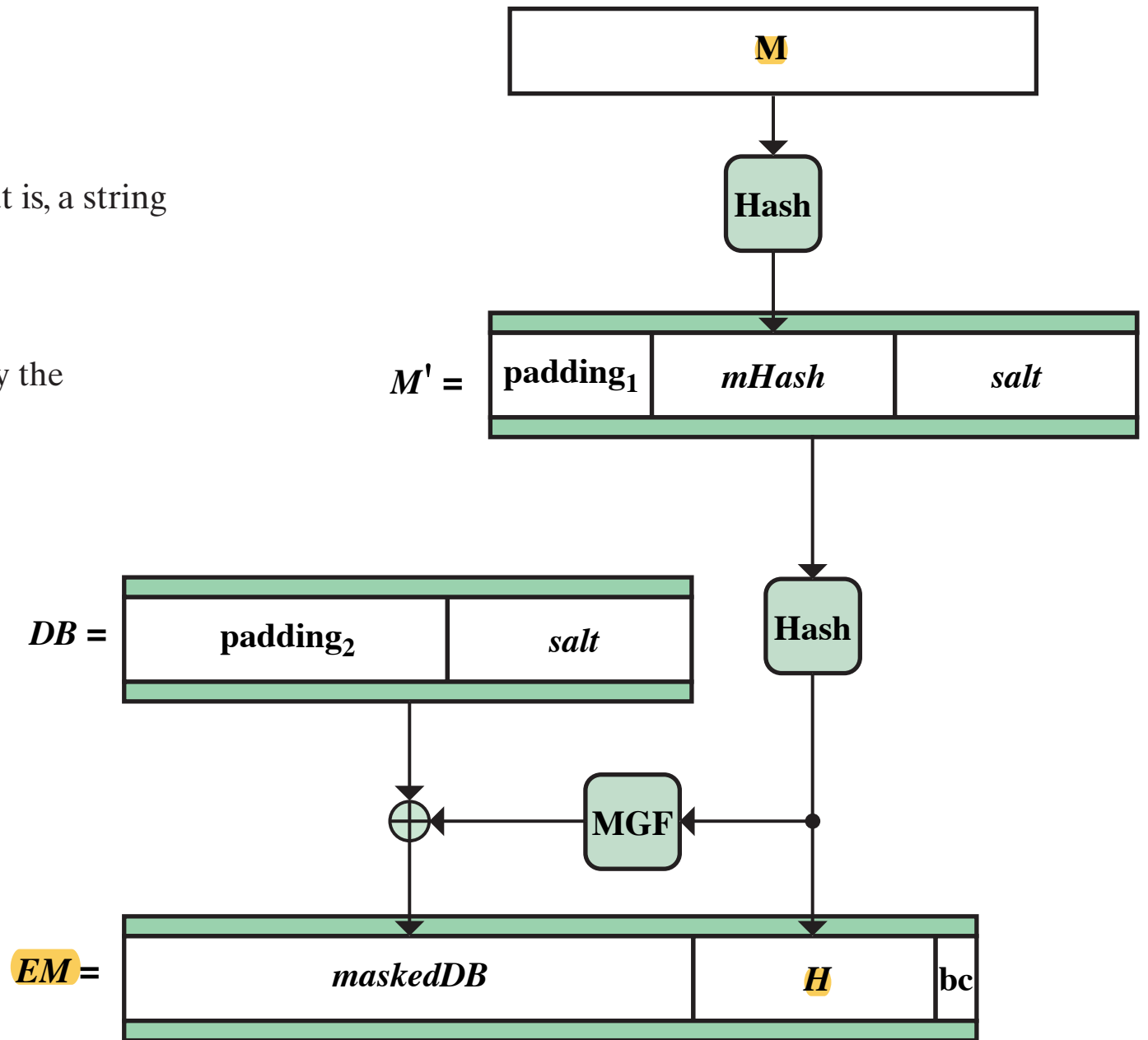


Figure 13.6 RSA-PSS Encoding

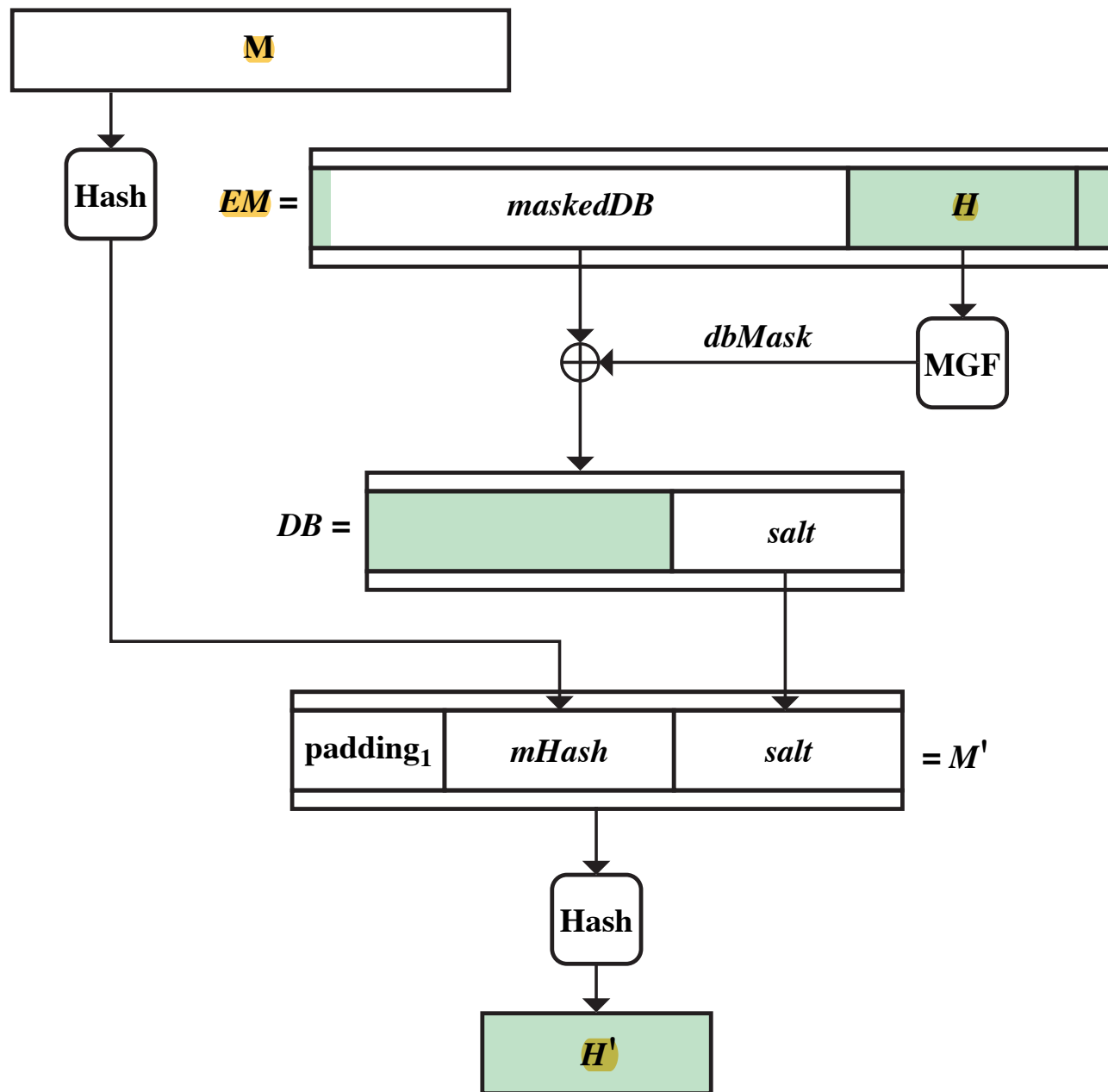


Figure 13.7 RSA-PSS EM Verification

FORMING THE SIGNATURE We now show how the signature is formed by a signer with private key $\{d, n\}$ and public key $\{e, n\}$ (see Figure 9.5). Treat the octet string EM as an unsigned, nonnegative binary integer m . The signature s is formed by encrypting m as follows:

$$s = m^d \bmod n$$

Let k be the length in octets of the RSA modulus n . For example if the key size for RSA is 2048 bits, then $k = 2048/8 = 256$. Then convert the signature value s into the octet string S of length k octets.

Signature Verification

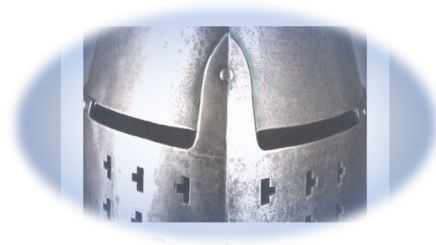
DECRYPTION For signature verification, treat the signature S as an unsigned, nonnegative binary integer s . The message digest m is recovered by decrypting s as follows:

$$m = s^e \bmod n$$

Then, convert the message representative m to an encoded message EM of length $emLen = \lceil (modBits - 1)/8 \rceil$ octets, where $modBits$ is the length in bits of the RSA modulus n .

Summary

- Digital signatures
 - Properties
 - Attacks and forgeries
 - Digital signature requirements
 - Direct digital signature
- Elgamal digital signature scheme
- RSA-PSS Digital Signature Algorithm
 - Mask generation function
 - The signing operation
 - Signature verification



- NIST digital signature algorithm
 - The DSA approach
 - The digital signature algorithm
- Elliptic curve digital signature algorithm
 - Global domain parameters
 - Key generation
 - Digital signature generation and authentication
- Schnorr digital signature scheme