Cryptography and Network Security (CS435/890BN)

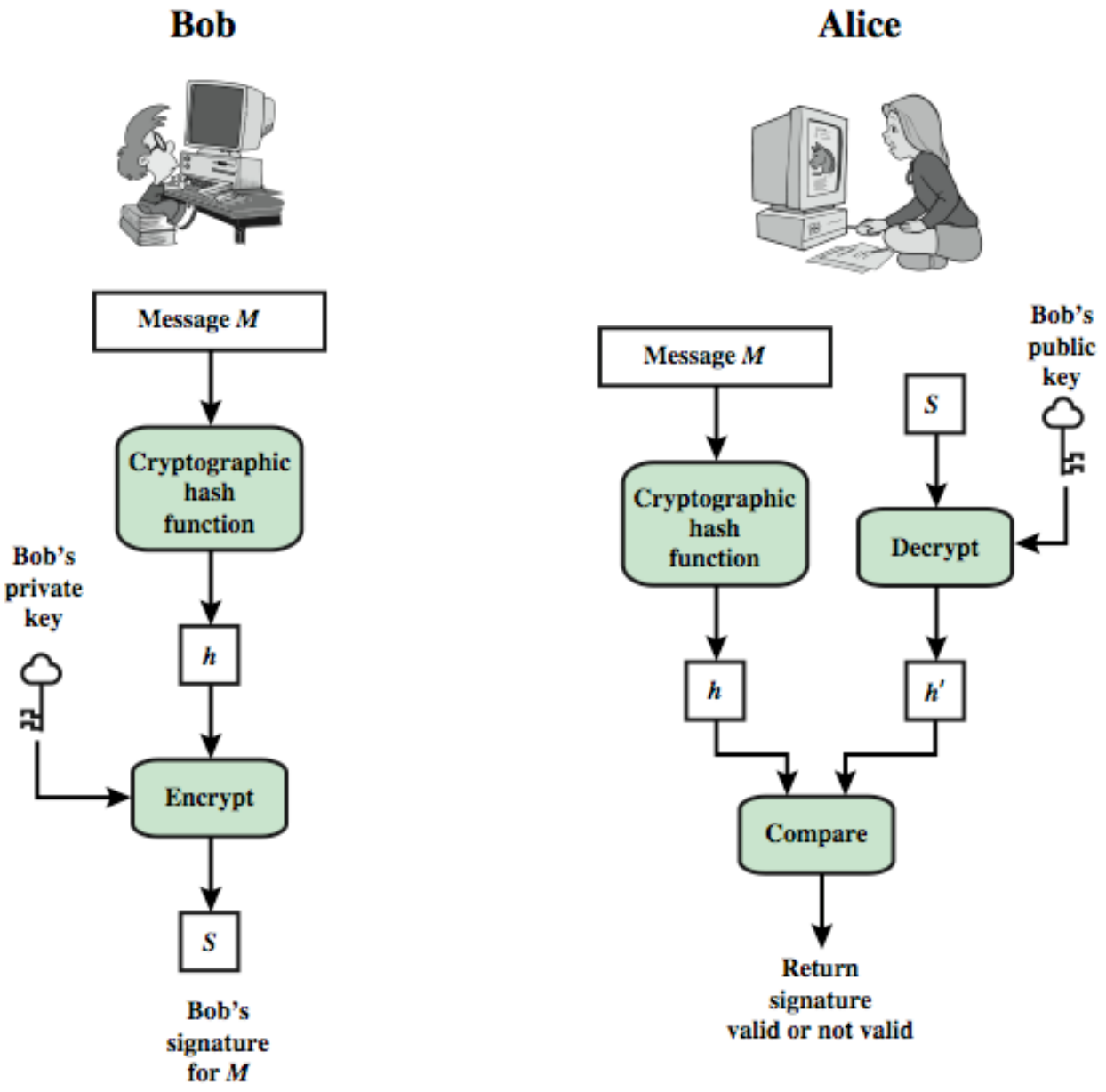
Part Eleven

(Digital Signatures)

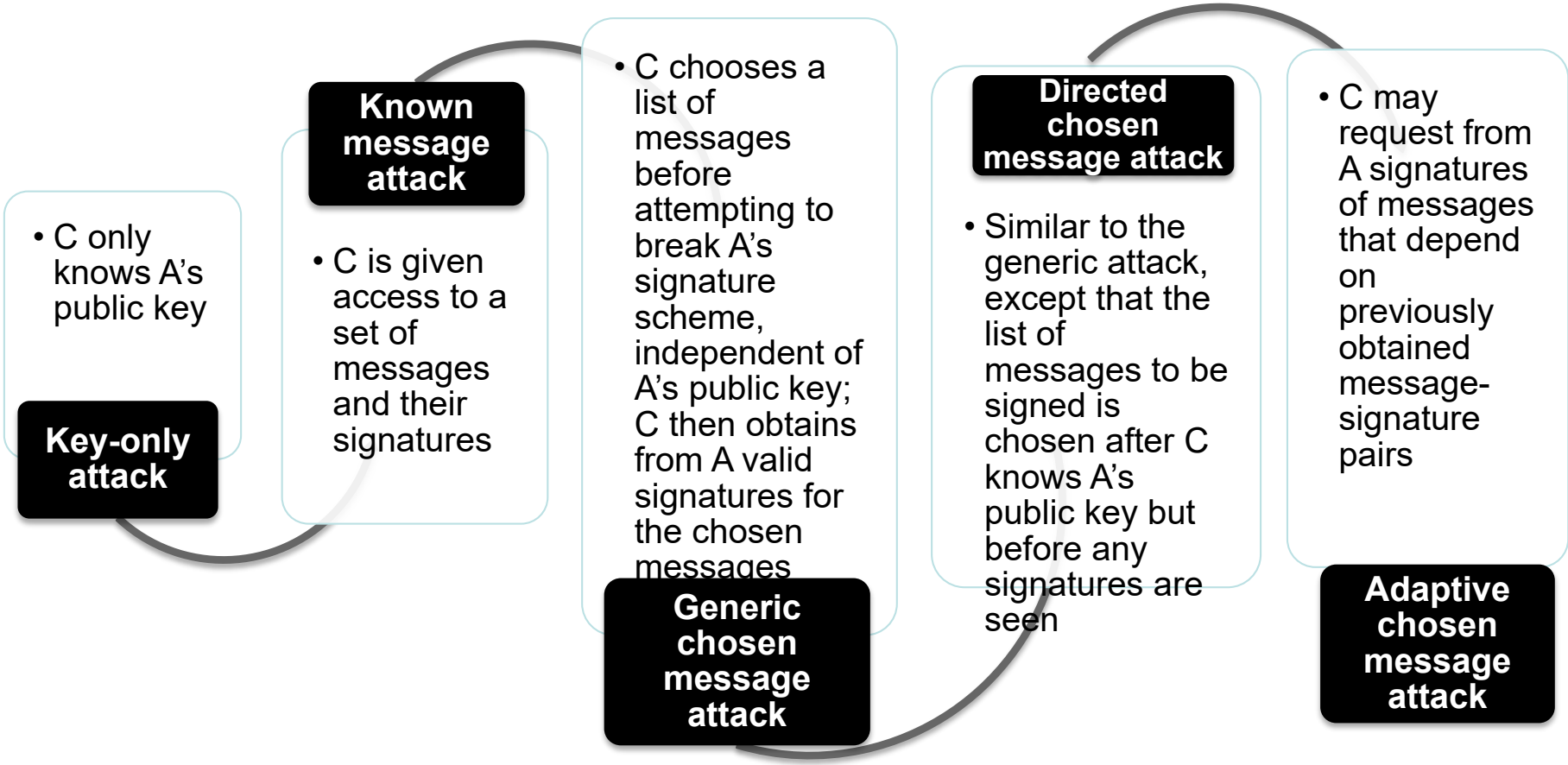
Digital Signatures

- 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

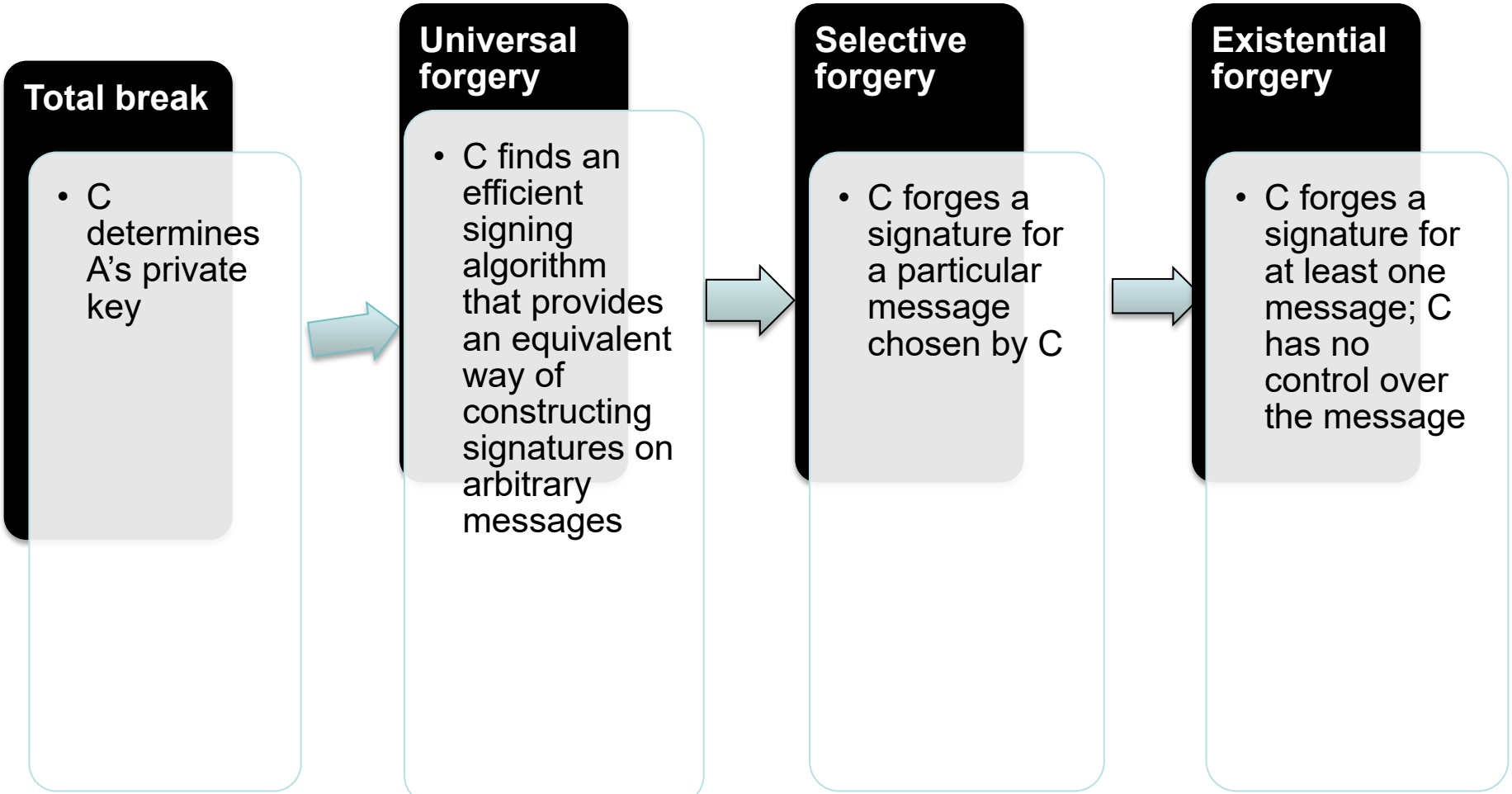
Digital Signature Model



Attacks


• C only knows A's public key
Key-only attack
Known message attack
• C is given access to a set of messages and their signatures
• 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
Generic chosen message attack
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
• C may request from A signatures of messages that depend on previously obtained message-signature pairs
Adaptive chosen message attack

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

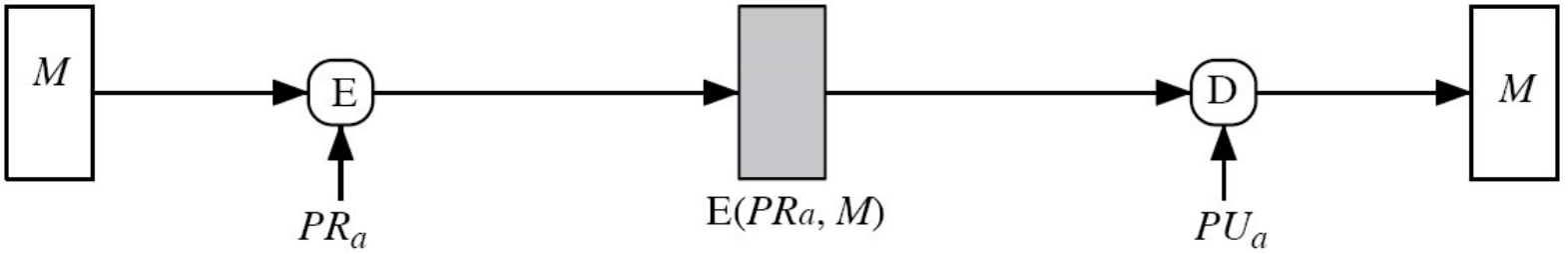
Digital Signature Requirements

- 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

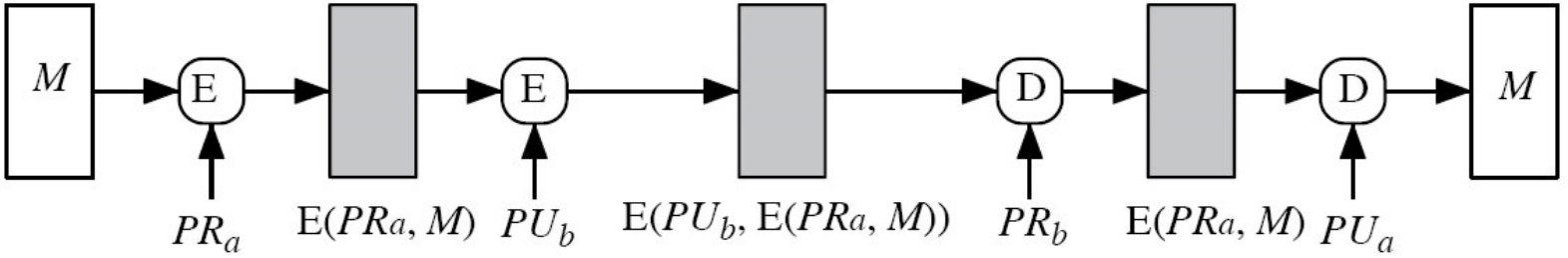
Direct Digital Signature

- 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

Direct Digital Signatures



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

Weakness: Security depends on sender's private-key

Arbitrated Digital Signatures

- involves use of arbiter A
  - validates any signed message
  - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message

Arbitrated Digital Signatures

(1) $X \rightarrow A: \ ID_X \| E(PR_x, [ID_X \| E(PU_y, E(PR_x, M))])$
(2) $A \rightarrow Y: E(PR_a, [ID_X \| E(PU_y, E(PR_x, M)) \| T])$

(c) Public-Key Encryption, Arbiter Does Not See Message

Notations:

X=sender
Y=recipient
A=Arbiter
ID_X=ID of X

M=message
T=time stamp
PR_X=X's private key
PU_Y=Y's public key
PR_A=A's private key

Weakness: multiple times public-key encryptions on the message

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

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

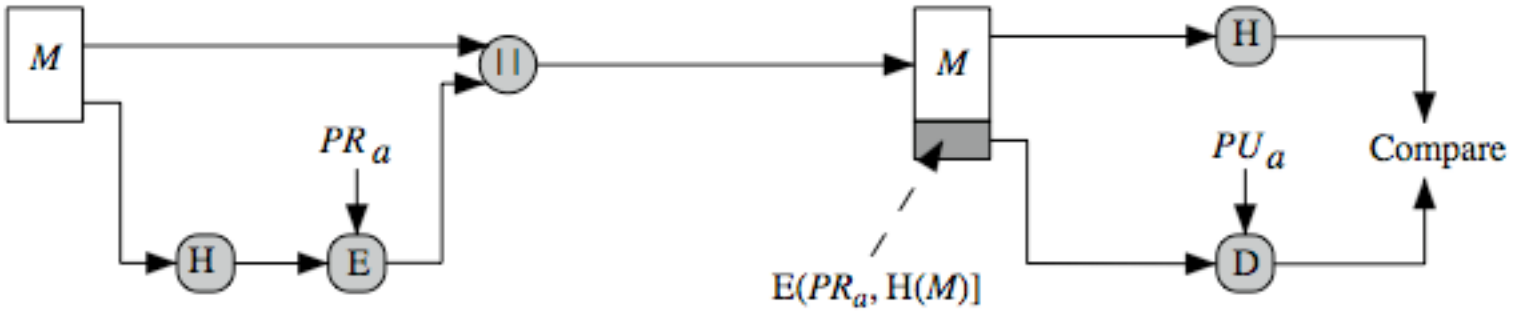
NIST Digital Signature Algorithm

- 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

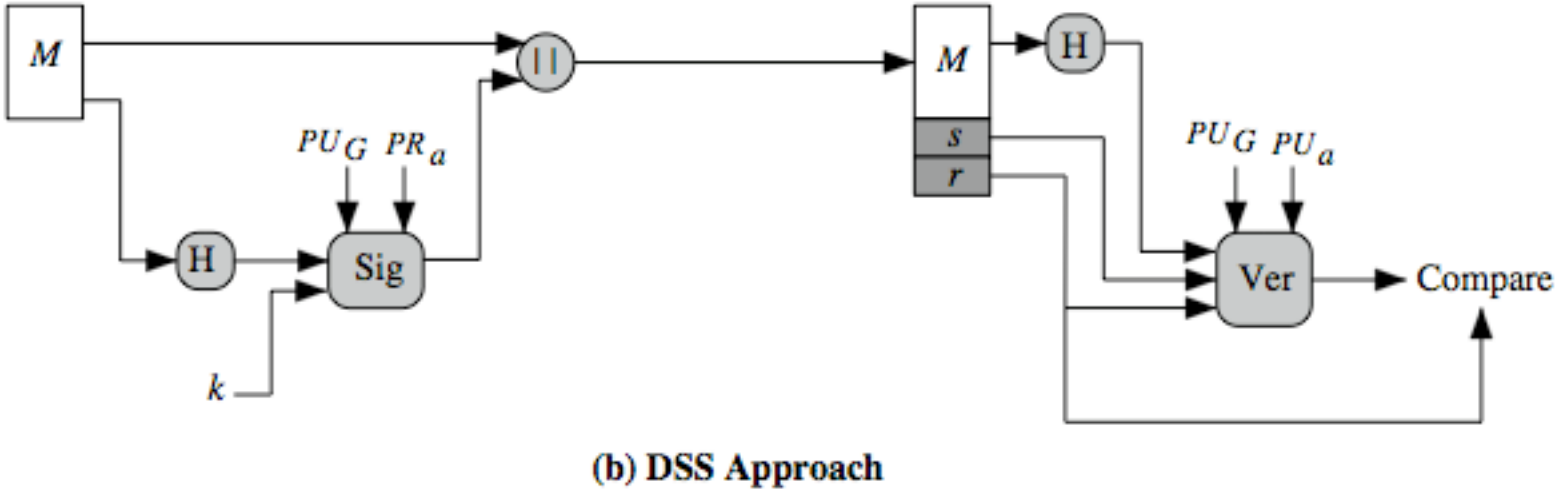
Digital Signature Algorithm (DSA)

- 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

Digital Signature Algorithm (DSA)



(a) RSA Approach



(b) DSS Approach

DSA Key Generation

- have shared global public key values (p,q,g):
  - choose a large prime p: $2^{L-1} < P < 2^{L}$ and q where L= 512 to 1024 bits and is a multiple of 64
    - and q is a prime factor of $(p-1)$: $2^{159} < q < 2^{160}$
  - commute $g = h^{(p-1)/q} \bmod p$
    - where $h<p-1$, $h^{(p-1)/q} \pmod p > 1$
- users choose private & compute public key:
  - choose $x<q$
  - compute $y = g^{x} \pmod p$

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 book web site for details of proof why

Global Public Key Components

p prime number where $2^{L-1} < p < 2^L$
for $512 \le L \le 1024$ 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

$r = (g^k \bmod p) \bmod q$

$s = \left[k^{-1}\left(\mathrm{H}(M) + xr\right)\right] \bmod q$

Signature = (r, s)

Verifying

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

$u_1 = \left[\mathrm{H}(M')w\right] \bmod q$

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

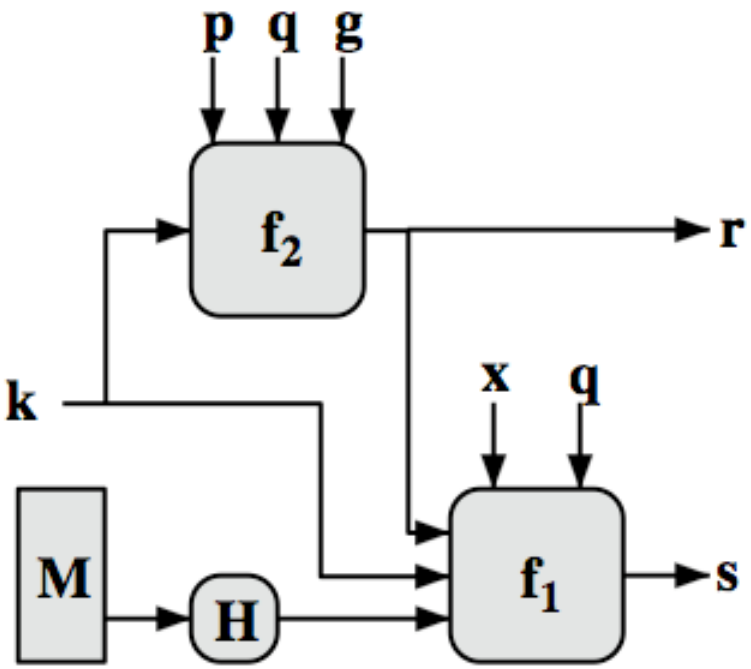
$v = \left[\left(g^{u1} y^{u2}\right) \bmod p\right] \bmod q$

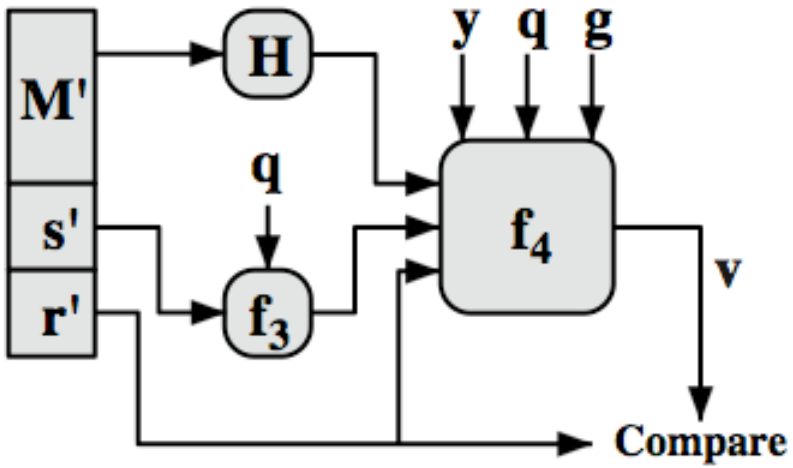
TEST: $v = r'$

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

Figure 13.3 The Digital Signature Algorithm (DSS)

DSS Overview





$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

(b) Verifying

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*

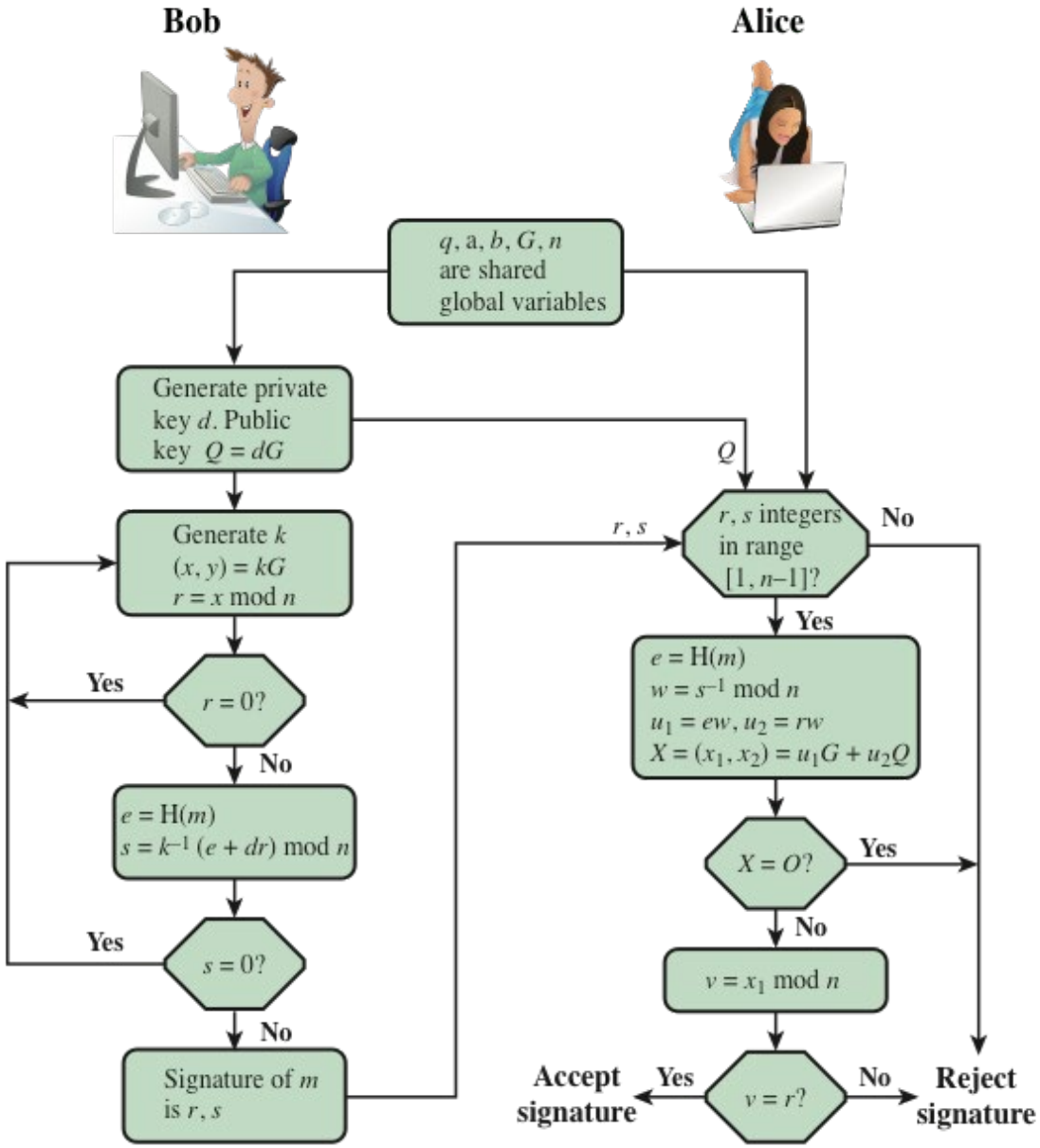


Figure 13.5 ECDSA Signing and Verifying

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

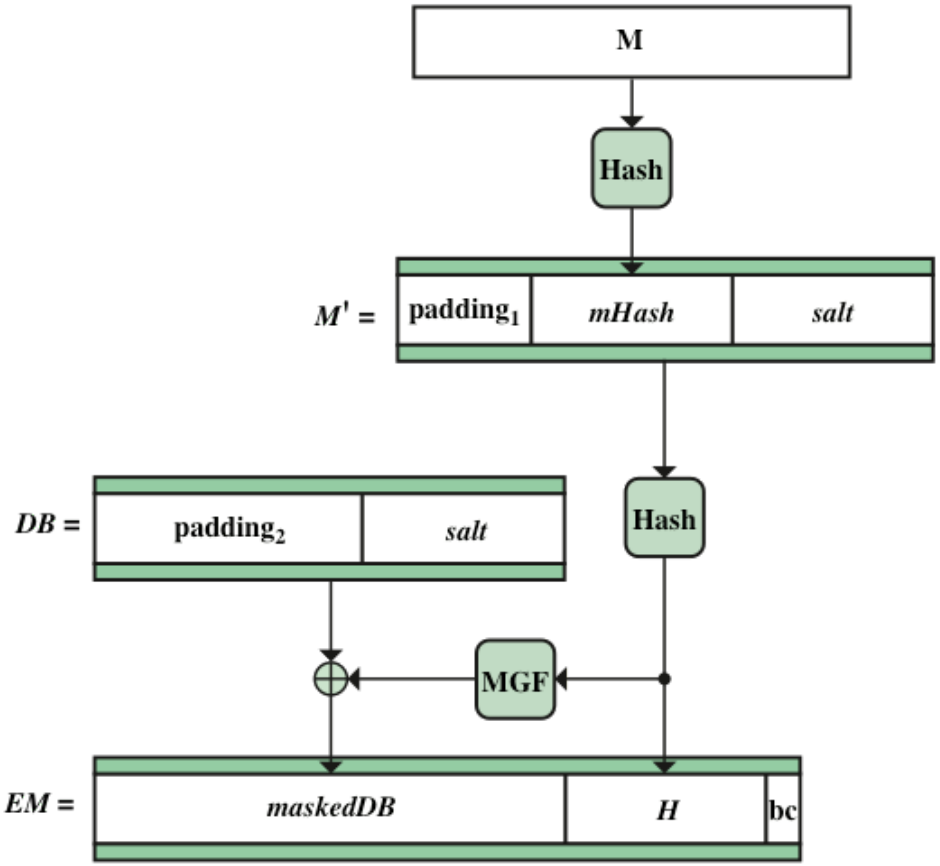


Figure 13.6 RSA-PSS Encoding

The first stage in generating an RSA-PSS signature of a message *M* is to generate from *M* a fixed-length message digest, called an encoded message *(EM*). Figure 13.6 illustrates this process.

We make several comments about the complex nature of this message digest algorithm. All of the RSA-based standardized digital signature schemes involve appending one or more constants (e.g., padding_1 and padding_2) in the process of forming the message digest. 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. RSA-PSS also incorporates a pseudorandom number, namely the salt. Because the salt changes with every use, signing the same message twice using the same private key will yield two different signatures. This is an added measure of security.

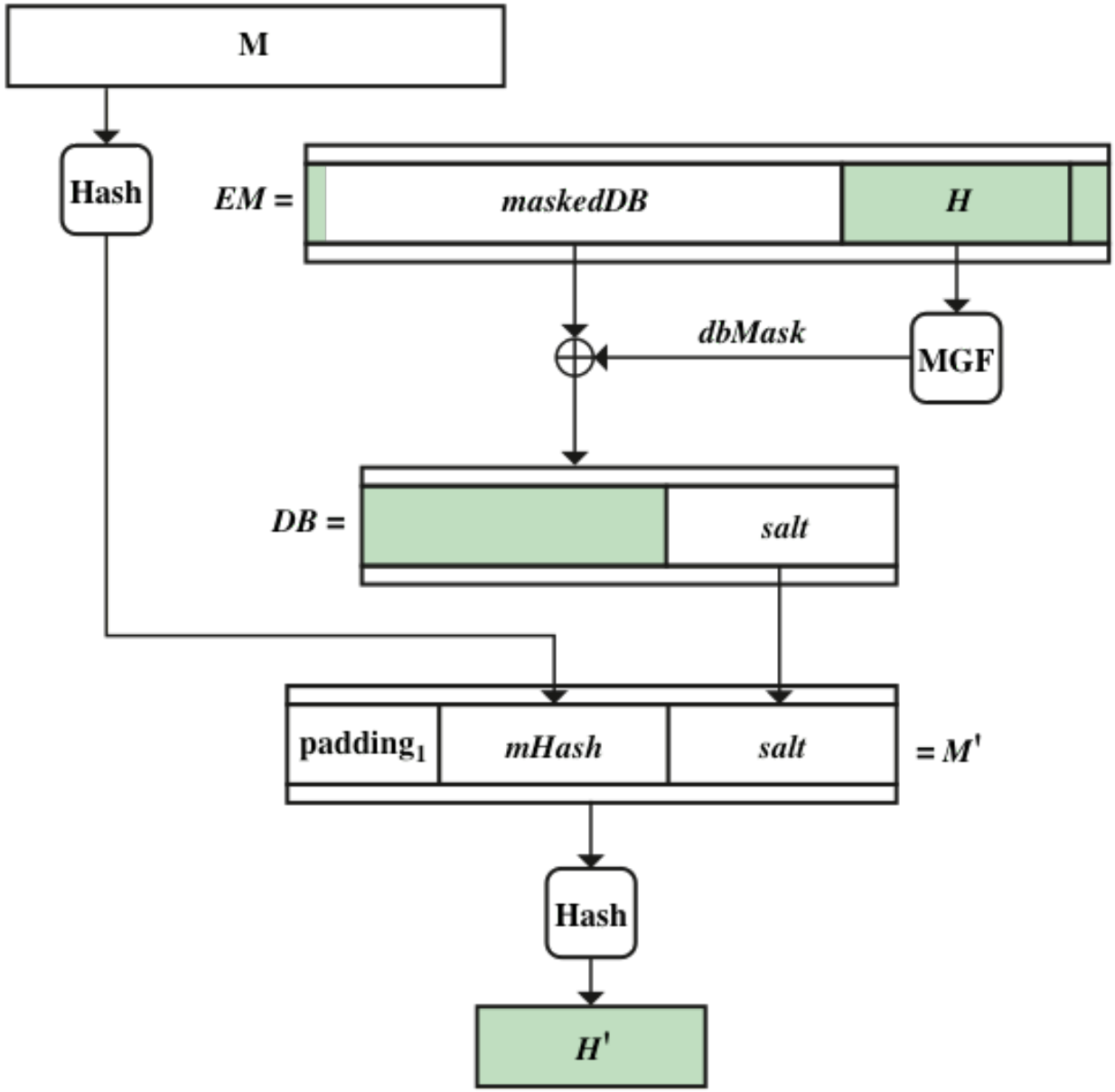


Figure 13.7 RSA-PSS EM Verification