## Digital Signatures

Message authentication

- Does not address issues of trust between two parties
- The receiver can forge a message
- The sender can deny sending a message

Digital signatures provide the ability to:

- Verify the author and the date and time of the signature
- Authenticate the contents at the time of the signature
- Be verifiable by third parties, to resolve disputes
- Essentially authentication function with additional capabilities

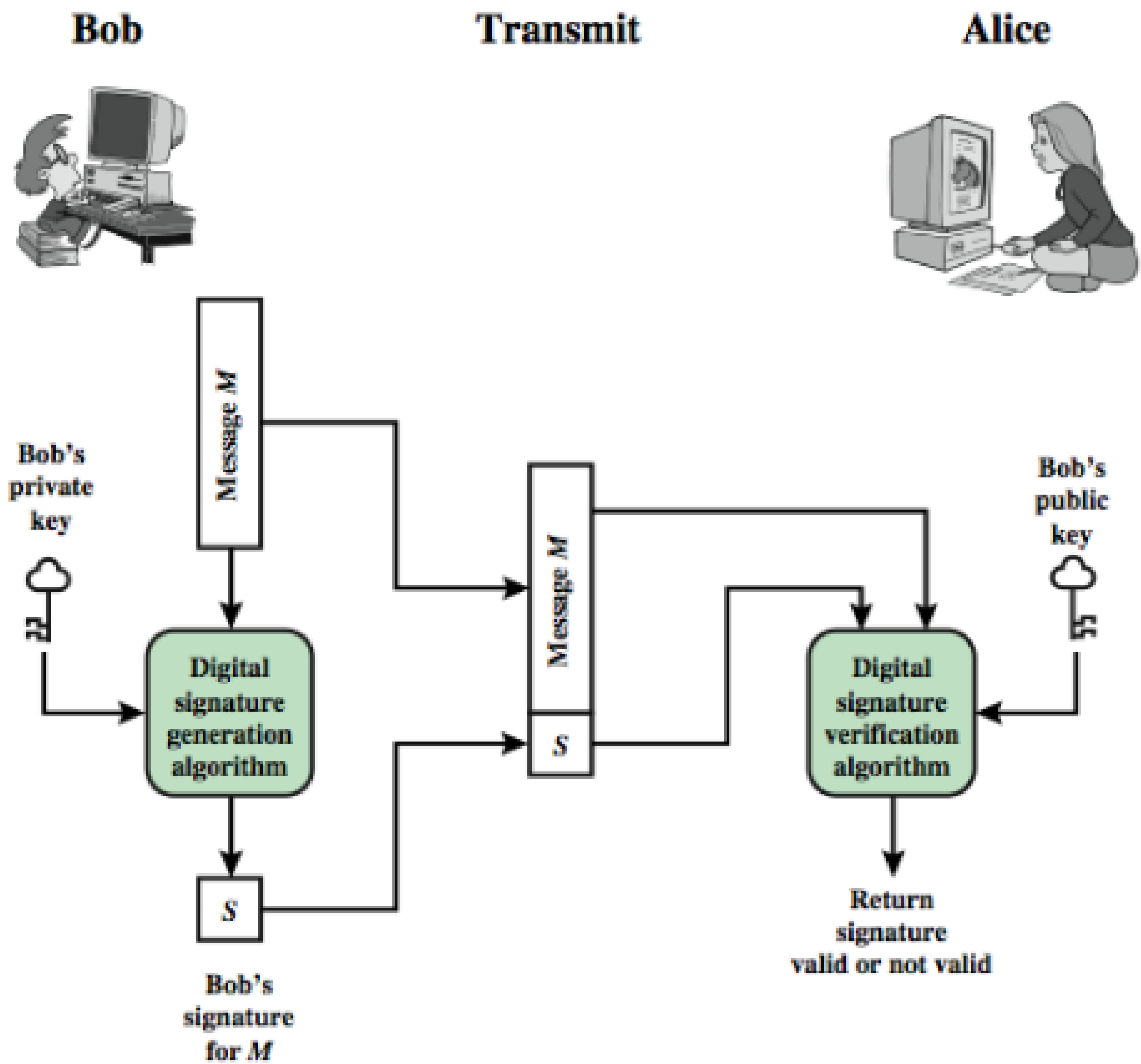


Figure 1: General Model of Digital Signature Process

### 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

Directed chosen message attack: Similar to 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

### Requirements for Digital Signatures

- Must depend on the message being signed
- Must use information unique to sender
  - This is to prevent both forgery and denial
- Must be relatively easy to recognize and verify
- Be computationally infeasible to forge...
  - New message for existing digital signature
  - Fraudulent digital signature for given message
- Be practical to retain a copy of in storage

## Direct Digital Signatures

- Involve only sender and receiver
  - Assumed receiver has sender's public-key
- Digital signature made by sender signing entire message or hash with private-key
- Can encrypt using receiver's public-key
- Should sign first then encrypt message & signature
  - The dispute resolver does not need to know the decryption key
- Security depends on sender's private-key

## ElGamal Digital Signature Setup

- Signature variant of ElGamal message exchange
- Depends on difficult of computing discrete logarithms
- Use private key for encryption (signing)
- Uses public key for decryption (verification)
- All users agree on global parameters
  - A large prime integer: q
  - A primitive root of q: a
- User A wants to sign and send a message to user B
  - User A: selects a random integer $X_A < q - 1$

### ElGamal Digital Signature

A signs a message M to B by computing:

- The hash $m = H(M)$, $0 \leq m \leq (q - 1)$
- Choose random integer k with $1 \leq k \leq (q\text{-}1)$ and $GCD(k,q\text{-}1) = 1$
- Compute temporary keys: $S_1 = a^k \bmod q$
- Compute $k^{-1}$ the multiplicative inverse of k mod (q-1)
- Compute the value: $S_2 = k^{-1} (m - X_A S_1) \bmod (q\text{-}1)$
- Signature is: $(S_1, S_2)$

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 $V_1 = V_2$

## Digital Signature Standard

- Uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- DSA only provides the digital signature function
- A public-key technique, difficulty of discrete logarithms
- Based on 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)
  - This is Schnorr Scheme
- Choose $g = h^{(p-1)/q} \bmod p$ (Schnorr Scheme)
  - Where $1 < h < p\text{-}1$ and $h^{(p-1)/q} \bmod p > 1$

User choose private and compute public key:

- Choose random private key: $0 < x < q$
- Compute public key: $y = g^x \bmod p$

## DSA Signature Creation

To sign a message M the sender:

- Generates a per-message secret number k, $0 < k < q$
- k must be random and be unique for each signing

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