Digital Signature

Sections

1. Digital signature overview
2. Hash functions
3. Digital signature algorithms

1. Digital signature overview

DIGITAL SIGNATURES

The most important development from the work on public-key cryptography is the digital signature.

A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. Typically the signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the source and integrity of the message. It must have the following properties:

It must verify the author and the date and time of the signature

It must to 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.

Informal definition

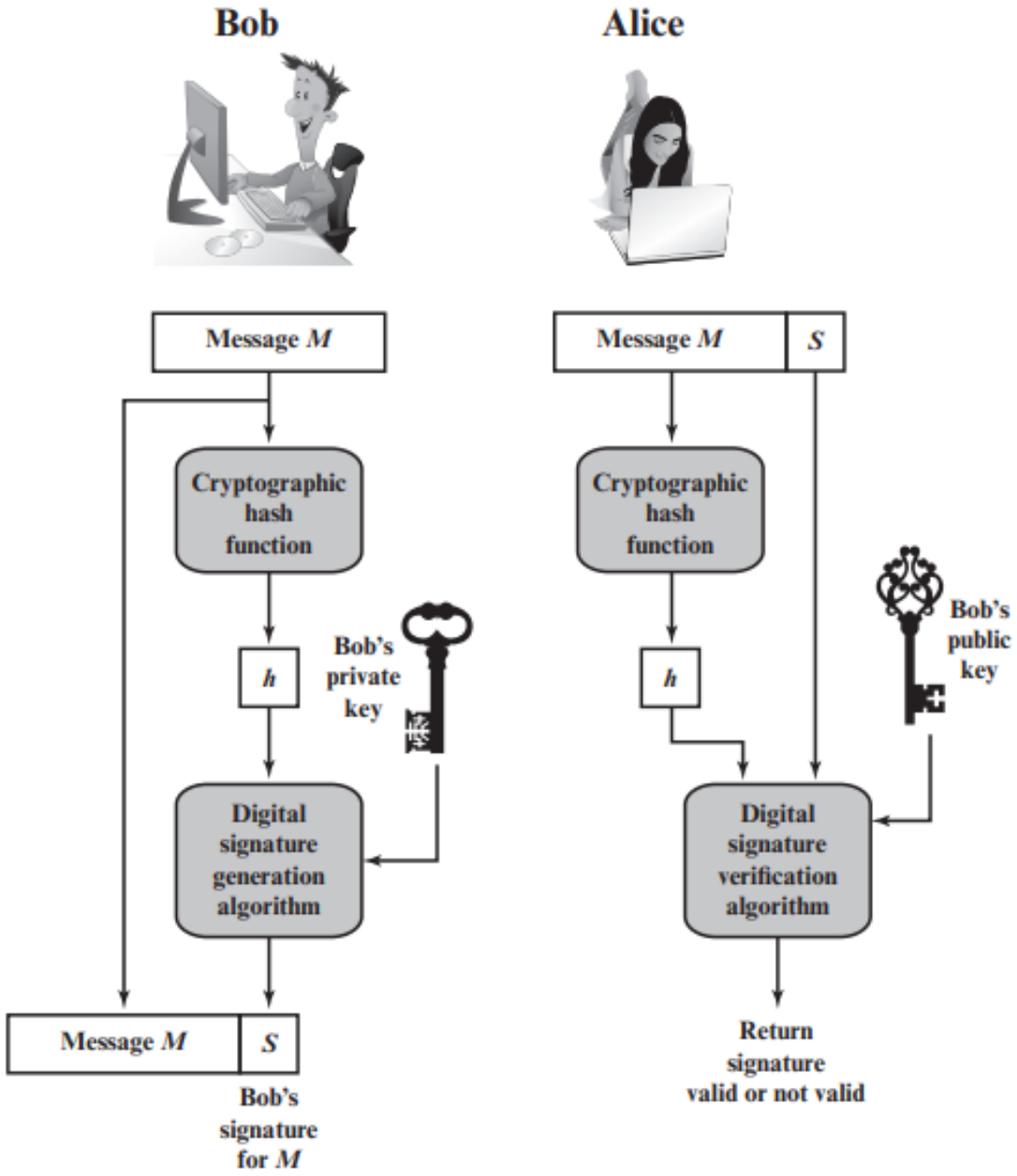
- Informally, a **digital signature** is a technique for establishing the origin of a particular message in order to settle later disputes about what message (if any) was sent.
- The purpose of a digital signature is thus for an entity to **bind its identity to a message**.
- We use the term **signer** for an entity who creates a digital signature, and the term **verifier** for an entity who receives a signed message and attempts to check whether the digital signature is “correct” or not.
- Digital signatures have many **attractive properties** and it is very important to understand exactly what **assurances** they provide and what their **limitations** are.

Advanced electronic signatures

The European Community Directive on electronic signatures also refers to the concept of an **advanced electronic signature** as:

an electronic signature that is:

1. uniquely linked to the signatory
2. capable of identifying the signatory
3. created using means under the sole control of the signatory
4. linked to data to which it relates in such a way that subsequent changes in the data is detectable



- **Digital Signature Model**

- Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other either fraudulently creating, or denying creation, of a message.

Attacks and Forgeries

- 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 breaks 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 the A may request signatures of messages that depend on previously obtained message signature pairs.

- 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.

Security requirements

We will define a **digital signature** on a message to be some data that provides:

- **Data origin authentication of the signer**
  - A digital signature validates the message in the sense that assurance is provided about the integrity of the message and of the identity of the entity that signed the message.
- **Non-repudiation**
  - A digital signature can be stored by anyone who receives the signed message as evidence that the message was sent and of who sent it. This evidence could later be presented to a third party who could use the evidence to resolve any dispute that relates to the contents and/or origin of the message.

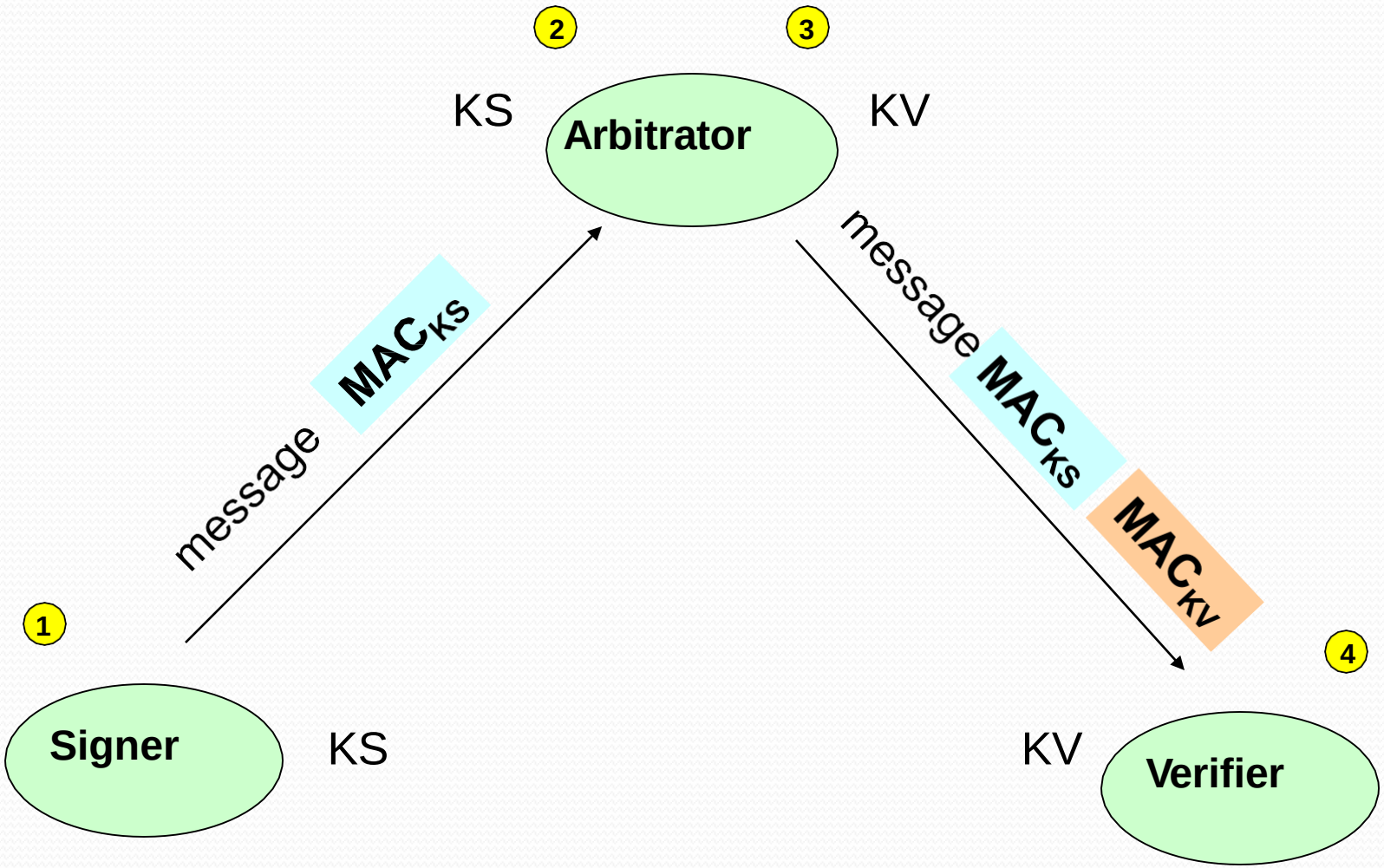
Input to a digital signature

- **The message**
  - Since a digital signature needs to offer data origin authentication (and non-repudiation) it is clear that the digital signature itself must be a piece of data that depends on the message, and cannot be a completely separate identifier.
  - It may be **sent** as a separate piece of data to the message, but its computation must involve the message.
- **A secret parameter known only by the signer**
  - Since a digital signature needs to offer non-repudiation, its calculation must involve a secret parameter that is known only by the signer.
  - The only possible exception to this rule is if the other entity is totally trusted
  - by all parties involved in the signing and verifying of digital signatures.

Properties of a digital signature

- **Easy for the signer to sign a message**
  - There is no point in having a digital signature scheme that involves the signer needing to use slow and complex operations to compute a digital signature.
- **Easy for anyone to verify a message**
  - Similarly we would like the verification of a digital signature to be as efficient as possible.
- **Hard for anyone to forge a digital signature**
  - It should be practically impossible for anyone who is not the legitimate signer to compute a digital signature on a message that appears to be valid. By “appears to be valid” we mean that anyone who attempts to verify the digital signature is led to believe that they have just successfully verified a valid digital signature on a message.

Arbitrated digital signatures



True digital signatures

- The vast majority of digital signature techniques do not involve having to communicate through a trusted arbitrator.
- A **true digital signature** is one that can be sent directly from the signer to the verifier. For the rest of this unit when we say “digital signature” we mean “true digital signature”.

True digital signature requirements	Public key encryption requirements
Only the holder of some secret data can sign a message	“Anyone” can encrypt a message
“Anyone” can verify that a signature is valid	Only the holder of some secret data can decrypt a message

- **DIRECT DIGITAL SIGNATURE**

- 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.

- Direct Digital Signatures involve the direct application of public-key algorithms involving only the communicating parties.

- A digital signature may be formed by encrypting the entire message with the sender's private key, or by encrypting a hash code of the message with the sender's private key.

- **Confidentiality** can be provided by further encrypting the entire message plus signature using either public or private key schemes.

- It is important to perform the signature function first and then an outer confidentiality function, since in case of dispute, some third party must view the message and its signature.

- But these approaches are dependent on the security of the sender's private-key. Will have problems if it is lost/stolen and signatures forged. The universally accepted technique for dealing with these threats is the use of a digital certificate and certificate authorities.

2. Hash functions

Hash functions

A **hash function** is a mathematical function that generally has the following three properties:

1. Condenses arbitrary long inputs into a fixed length output

- You stuff as much data as you want into the function, and it churns out an output (or **hash**) that is always the same fixed length.
- In general this hash is much smaller than the data that was put into the function.
- Because the hash is a smaller thing that represents a larger thing, it sometimes referred to as a **digest**, and the hash function as a **message digest function**.

Hash functions

2. Is one-way

- The hash function should be easy to compute, but given the hash of some data it should be very hard to recover the original data from the hash.

3. It is hard to find two inputs with the same output

- Weak collision resistance :

  Assume you have a message M with hash value h(M).

  Then it should be hard to find a different message M' such that h(M) = h(M').

- Strong collision resistance :

  It should be hard to find two different message M1 and M2 such that h(M1) = h(M2).

- Strong collisions resistance is hard to prove.

Hash functions and data integrity

- **A hash function provides a weak notion of data integrity**.
  - If we had a list of MD5 hashes which contained information on all of our operating system files on our home computer you could verify the values of your files in the list and see which files have been changed or have been updated by say a virus.
- **BUT**
  - If a virus replaced the system file it could also replace the MD5 values in your list with new ones and you would not be aware this had happened…

Hash function applications

Digital signatures with appendix: hash-functions are used to bind data together and make the signature process more efficient.

Password storage: hash-functions are sometimes used to store highly confidential data such as passwords.

Cryptographic protocols: hash-functions are often used within cryptographic protocols (including entity authentication protocols) to bind different data items together.

Hash-functions can be used as components from which to construct other cryptographic primitives .

Cryptographic Hash Functions

- Public key algorithms are too slow to sign large documents. A better protocol is to use a one way hash function also known as a cryptographic hash function (CHF).
- CHFs are checksums or compression functions: they take an arbitrary block of data and generate a unique, short, fixed-size, bitstring.

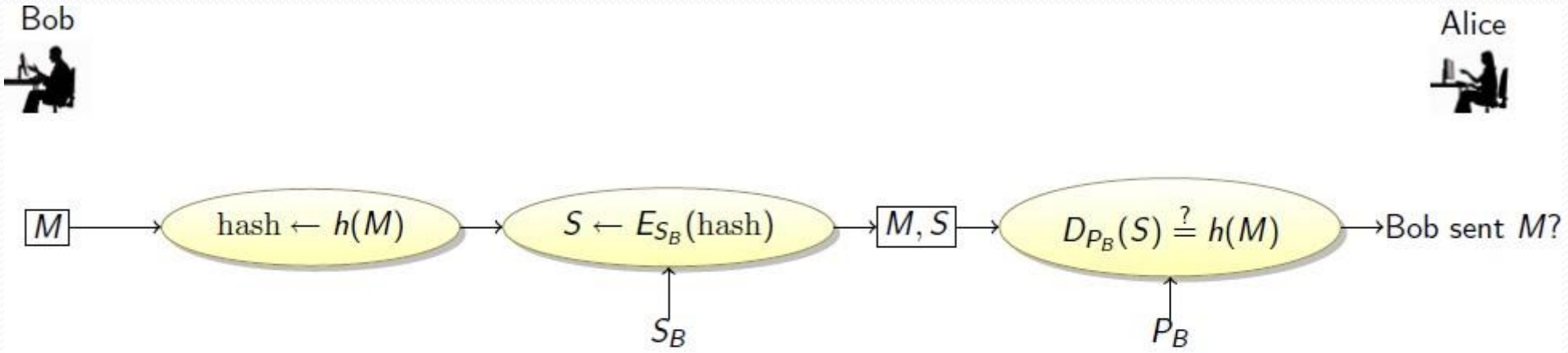
```
> echo "hello" | sha1sum
f572d396fae9206628714fb2ce00f72e94f2258f  -
> echo "hella" | sha1sum
1519ca327399f9d699afb0f8a3b7e1ea9d1edd0c  -
> echo "can't believe it's not butter!"|sha1sum
34e780e19b07b003b7cf1babba8ef7399b7f81dd  -
```

Signature Protocol

1. Bob computes a one-way hash of his document.
2. Bob encrypts the hash with his private key, thereby signing it.
3. Bob sends the encrypted hash and the document to Alice.
4. Alice decrypts the hash Bob sent him, and compares it against a hash she computes herself of the document. If they are the same, the signature is valid.

$$\begin{aligned} \text{hash} &\leftarrow h(M) \\ \text{sig} &\leftarrow E_{S_B}(\text{hash}) \\ D_{P_B}(\text{sig}) &\stackrel{?}{=} h(M) \end{aligned}$$

Signature Protocol. . .



- Advantage: the signature is short; defends against MITM attack.

3. Digital signature algorithms

Elgamal Scheme

- This scheme is variant of digital signature algorithm.
- This scheme is based on computing assumption of ***large prime number.***
- It is computationally very complex to compute S_1 and S_2.
- This scheme assure that authenticity of message m sent by sender/signer to verifier.
- As with Elgamal encryption, the global elements of **Elgamal digital signature** is based on prime number **q** and $\boldsymbol{\alpha}$, which is a primitive root of **q**.

Elgamal Scheme

❑**Generating private key & public key pair:**

1. Generate a random integer X_A, such that $1 < X_A < q-1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. ***A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.***

❑**For example, q = 19, α = 10**

1. Sender chose ***$X_A = 16$.***
2. Compute $Y_A = \alpha^{X_A} \bmod q = 10^{16} \bmod 19 = 4$. ***$Y_A = 4$.***
3. Sender's private key is 16; Sender's public key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.

✓***Sender wants to sign a message with hash value m = 14.***

Elgamal Scheme

- **Create Digital Signature:**

1. Choose a random integer K such that $1 \leq K \leq q-1$ and gcd (K, q-1) = 1. K is relatively prime to q-1.
2. Compute $S_1 = \alpha^K \bmod q$.
3. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
4. The signature consists of the pair (S_1, S_2).

- **For example,**

1. Sender chooses K = 5, which is relatively prime to q – 1 = 18.
2. $S_1 = \alpha^K \bmod q = 10^5 \bmod 9 = 3$. $S_1 = 3$
3. $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1) = 11\ (14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

$S_2 = 4$

Elgamal Scheme

❑Signature Verification

1. Calculate $V_1 = \alpha^m \bmod q$
2. Calculate $V_2 = (Y_A)^{S1} (S_1)^{S2} \bmod q$

❑For example,

1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16.$ **$V_1 = 16$**

2. $V_2 = (Y_A)^{S1} (S_1)^{S2} \bmod q = (4)^3 (3)^4 \bmod 19 = 5184 \bmod 19 = 16.$

$V_2 = 16$

Schnorr Scheme

- The Schnorr signature scheme is also based on discrete logarithms.
- The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature.
- The main work for signature generation does not depend on the message.
- The scheme is based on using a prime modulus p, with having a (p-1) prime factor of q appropriate size; that is, $p = 1 \pmod q$. Typically, we use $p = 2^{1024}$ and $q = 2^{160}$

Schnorr Scheme

❑Generating private key & public key pair:

1. Choose primes p and q, such that q is a prime factor of p-1.
2. Choose an integer α, such that $\alpha^q = 1 \bmod p$. The values α, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's **private key.**
4. Calculate $v = \alpha^{-s} \bmod p$. This is the user's **public key.**

❑ Create Digital Signature:

1. Choose a random integer r with $0 < r < q$ and compute $x = \alpha^r \bmod p$. This computation is a pre-processing stage independent of the message M to be signed.
2. Concatenate the message with and hash the result to compute the value :

$$e = H(M \| x)$$

3. Compute $y = (r + se) \bmod q$. The signature consists of the pair (e, y).

Schnorr Scheme

❑ **Signature Verification**

1. Compute $x' = \alpha^{y} v^{e} \bmod p$

$= \alpha^{y} \alpha^{-se} \bmod p \qquad (\because v = \alpha^{-s} \bmod p)$

$= \alpha^{(y-se)} \bmod p$

$= \alpha^{r} \bmod p \qquad (\because y = r + se)$

$= x$

So, Here $x' = x$.

2. Verify that $e = H(M \| x)$.

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

Comparison of Elgamal and Schnorr

- Elgamal Signature scheme is ***more time consuming*** in compare to Schnorr Scheme.

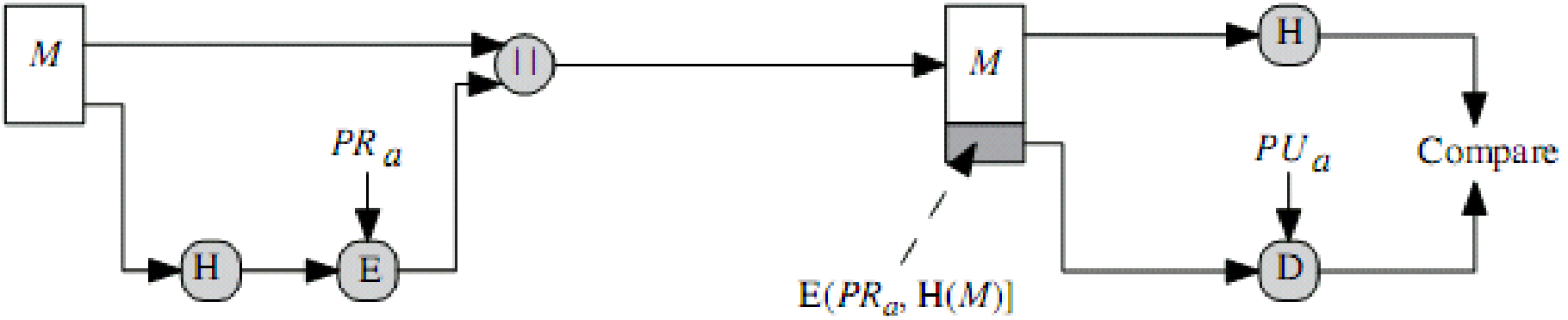
- Schnorr scheme is ***6 times faster*** than Elgamal and produce signature which is ***6 times smaller***.

DIGITAL SIGNATURE STANDARD

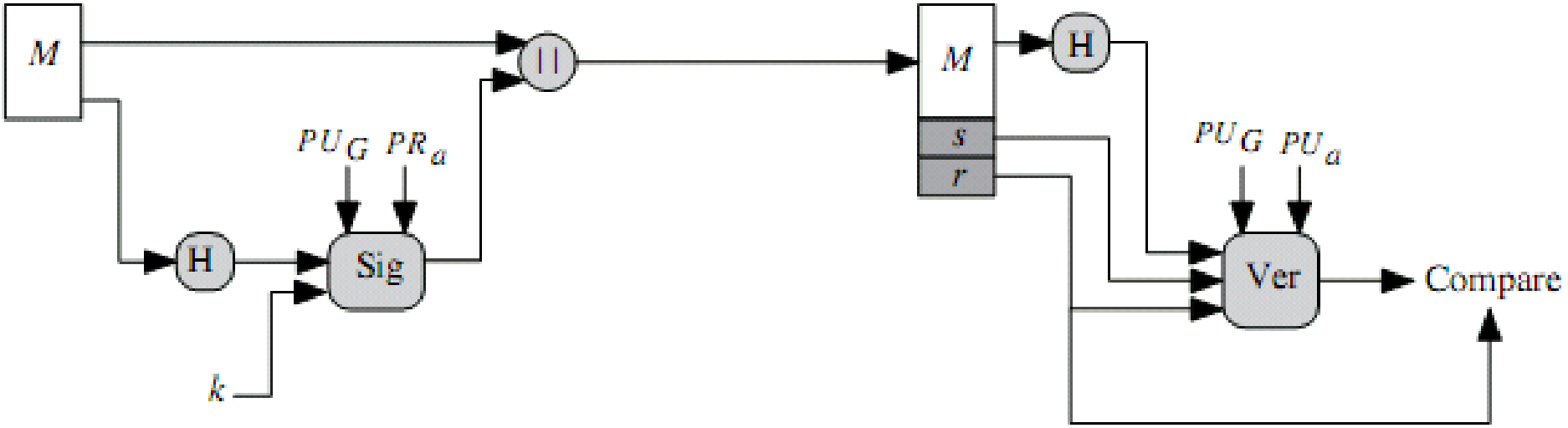
The Digital Signature Standard (DSS) makes use of the Secure Hash Algorithm (SHA) described and presents a new digital signature technique, the Digital Signature Algorithm (DSA).

This latest version incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.

In this section, we discuss the original DSS algorithm. The DSS 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.



(a) RSA Approach



(b) DSS Approach

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.

The DSS 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).

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.

Digital Signature Algorithm

The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by ElGamal and Schnorr.

The DSA signature scheme has advantages, being both smaller (320 vs 1024bit) and faster over RSA. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique

DSA typically uses a common set of global parameters (p,q,g) for a community of clients, as shown. A 160-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 the form $h^{(p-1)/q} \ mod \ p$ where h is an integer between 1 and (p – 1) with the restriction that g must be greater than 1. Thus, the global public key components of DSA have the same for as in the Schnorr signature scheme.

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

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

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

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

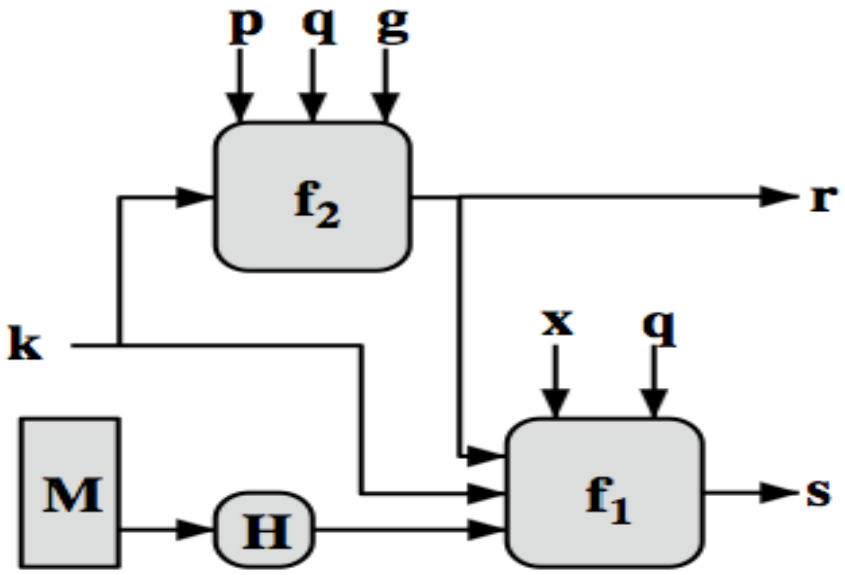
$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

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 13.3 The Digital Signature Algorithm (DSA)

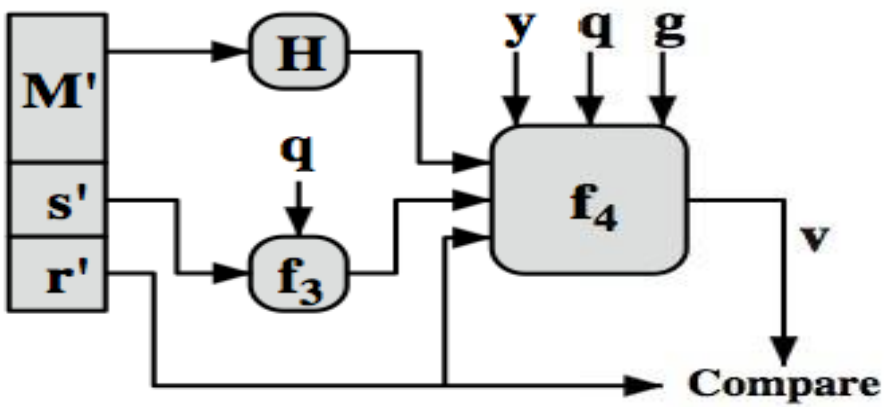
Then each DSA, choose a random private key x, and compute their public key as shown. The calculation of the public key y given x is relatively straightforward. However, given the public key y, it is computationally infeasible to determine x, which is the discrete logarithm of y to base g, mod p.



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

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

(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

At the receiving end, receiver generates a quantity verification is performed using the formulas shown. The v that is a function of the public key components, the sender's public key, and the hash of the incoming message. If this quantity matches the r component of the signature, then the signature is validated.

Note that the difficulty of computing discrete logs is why it is infeasible for an opponent to recover k from r, or x from s. Note also that nearly all the calculations are mod q, and hence are much faster save for the last step.

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.

Signing and Verifying

The structure of the algorithm, as revealed here 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 (mod q) is passed to a function that also has 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.