Unit 4: Cryptographic Hash Functions and Digital Signatures

⇒ A **hash function H** accepts a variable-length block of data **M** as input and produces a fixed-size hash value **h = H(M).**

⇒ A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random.

⇒ In general terms, the principal object of a hash function is data integrity. A change to any bit or bits in **M** results, with high probability, in a change to the hash code.

⇒ The kind of hash function needed for security applications is referred to as a **cryptographic hash function**.

⇒ A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either

a) a data object that maps to a pre-specified hash result(the one-way property) or

b) two data objects that map to the same hash result (the collision-free property)

⇒ Because of these characteristics, hash functions are often used to determine whether or not data has changed.

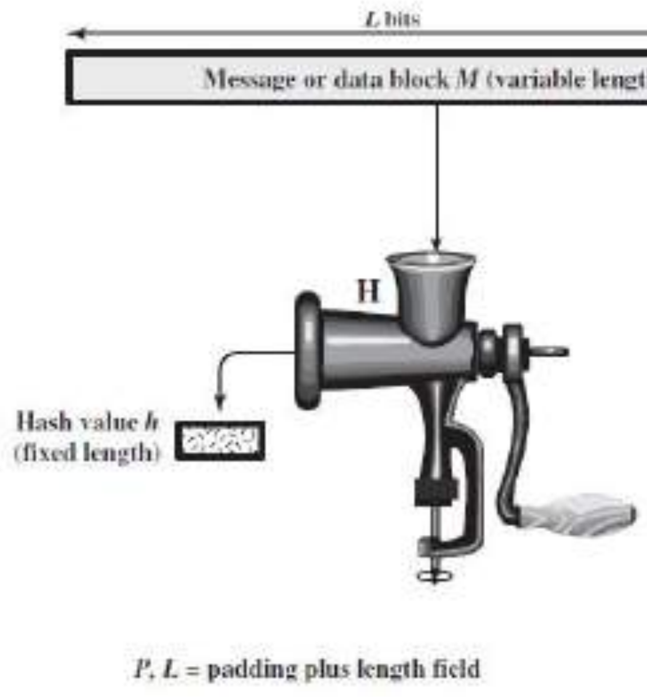

L bits
Message or data block M (variable length)
P, L
H
Hash value h
(fixed length)
P, L = padding plus length field

Figure: Cryptographic Hash Function; h = H(M)

Properties of Hash functions

- These are also known as Security Requirements for Cryptographic Hash Functions

1. **Variable input size**
   - **H** can be applied to a block of data of any size.
2. **Fixed output size**
   - **H** produces a fixed-length output.
3. **Efficiency**
   - **H(x)** is relatively easy to compute for any given **x**, making both hardware and software implementations practical.
4. **Preimage resistant (one-way property)**
   - For any given hash value **h**, it is computationally infeasible to find **y** such that **H(y) = h.**
   - ***It is easy to generate a code given a message, but virtually impossible to generate a message given a code***.
   - This property is important if the authentication technique involves the use of a secret value
5. **Second preimage resistant (weak collision resistant)**
   - For any given block **x**, it is computationally infeasible to find $\mathbf{y} \neq \mathbf{x}$ with **H(y) = H(x).**
   - This property guarantees that ***it is impossible to find an alternative message with the same hash value as a given message***.
   - This prevents forgery when an encrypted hash code is used.
   - If this property were not true, an attacker would be capable of the following sequence:
     - First, observe or intercept a message plus its encrypted hash code
     - Second, generate an unencrypted hash code from the message
     - Third, generate an alternate message with the same hash code
6. **Collision resistant (strong collision resistant)**
   - It is computationally infeasible to find any pair **(x, y)** such that **H(x) = H(y).**
   - A hash function that satisfies the first is referred to as a **weak hash function**.
   - If the sixth property, collision resistant, is also satisfied, then it is referred to as a **strong hash function**.

- A strong hash function protects against an attack in which one party generates a message for another party to sign.
    - For example, suppose Bob writes an IOU message, sends it to Alice, and she signs it.
    - Bob finds two messages with the same hash, one of which requires Alice to pay a small amount and one that requires a large payment.
    - Alice signs the first message, and Bob is then able to claim that the second message is authentic.

7. **Pseudorandomness**
    - Output of **H** meets standard tests for pseudorandomness.
    - pseudorandomness, has not traditionally been listed as a requirement of cryptographic hash functions but is more or less implied

Applications of Cryptographic Hash Functions

⇒ ***Message Authentication***

⇒ ***Digital signature***

⇒ ***Other Applications***

Message Authentication

— Message authentication is a mechanism or service used to verify the integrity of a message.

— Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay).

— In many cases, there is a requirement that the authentication mechanism assures that purported identity of the sender is valid.

— When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest

The essence of the use of a hash function for message authentication is as follows.

- The sender computes a hash value as a function of the bits in the message and transmits both the hash value and the message.
- The receiver performs the same hash calculation on the message bits and compares this value with the incoming hash value.
- If there is a mismatch, the receiver knows that the message (or possibly the hash value) has been altered

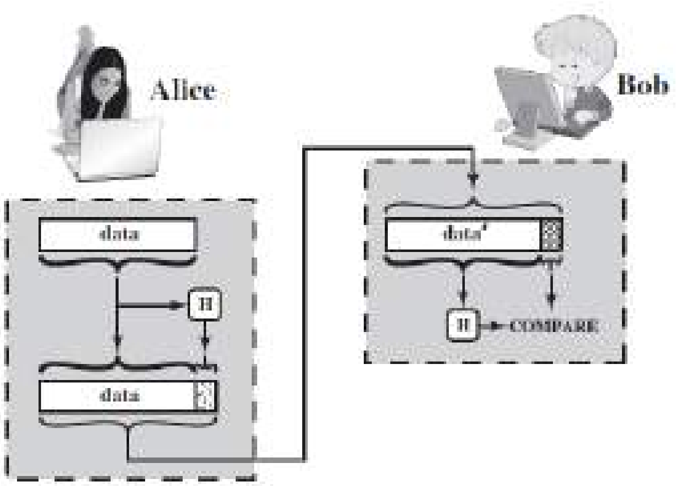


Figure: Use of hash function to check data integrity

The hash function must be transmitted in a secure fashion. That is, the hash function must be protected so that if an adversary alters or replaces the message, it is not feasible for adversary to also alter the hash value to fool the receiver. This type of attack is shown in following Figure.

In this example, Alice transmits a data block and attaches a hash value. Darth intercepts the message, alters or replaces the data block, and calculates and attaches a new hash value. Bob receives the altered data with the new hash value and does not detect the change.

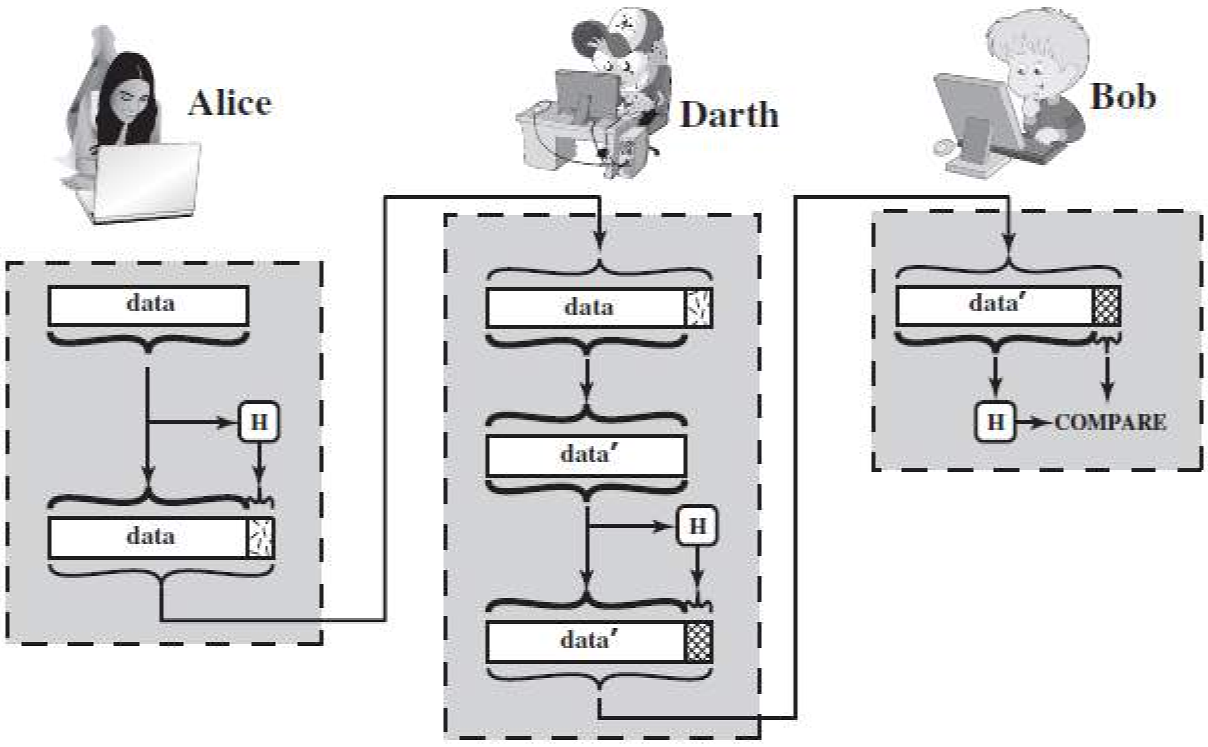


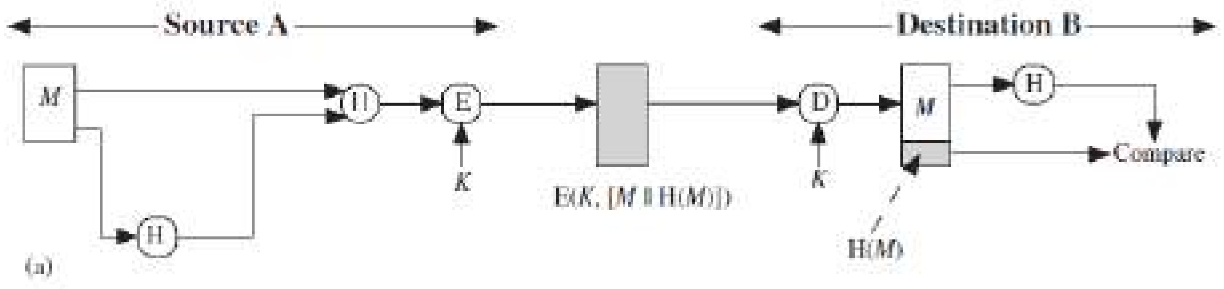
Figure: Attack Against Hash Function

To prevent this attack, the hash value generated by Alice must be protected.

Variety of ways in which a hash code can be used to provide message authentication:

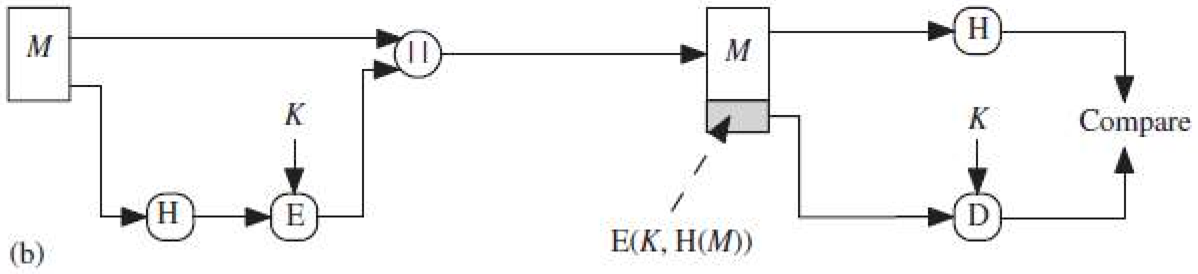
a) The message plus concatenated hash code is encrypted using symmetric encryption.

- Because only A and B share the secret key, the message must have come from **A** and has not been altered.
- The hash code provides the structure or redundancy required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, confidentiality is also provided.



b) Only the hash code is encrypted, using symmetric encryption.

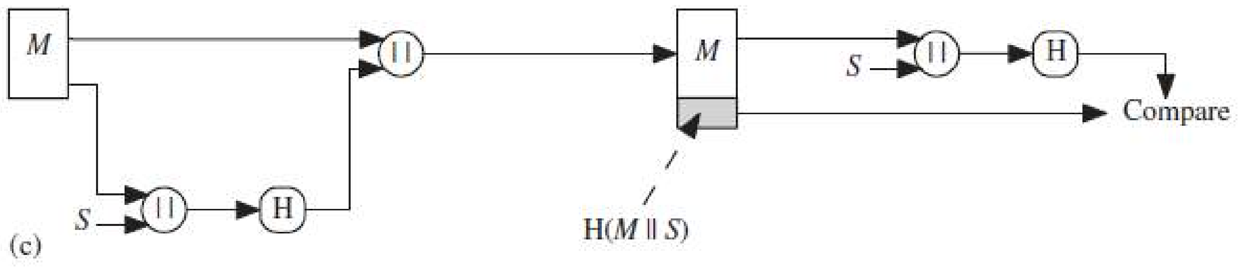
- This reduces the processing burden for those applications that do not require confidentiality.



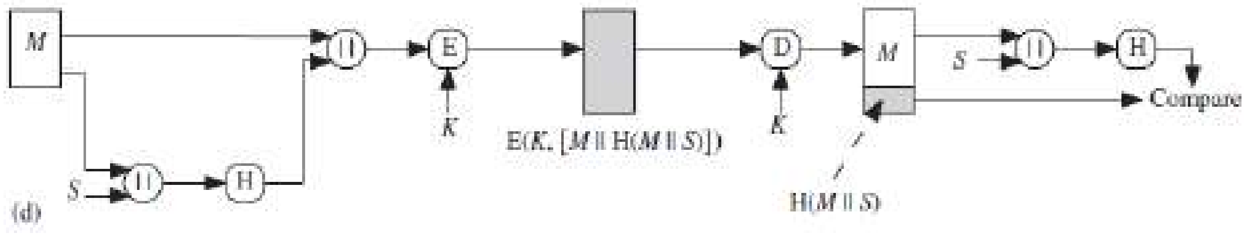
c) It is possible to use a hash function but no encryption for message authentication.

- The technique assumes that the two communicating parties share a common secret value **S**.
- A computes the hash value over the concatenation of **M** and **S** and appends the resulting hash value to **M**.
- Because **B** possesses **S**, it can recompute the hash value to verify.

— Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.



d) Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.



Note:

— When confidentiality is not required, method (b) has an advantage over methods (a) and (d), which encrypts the entire message, in that less computation is required. Nevertheless, there has been growing interest in techniques that avoid encryption

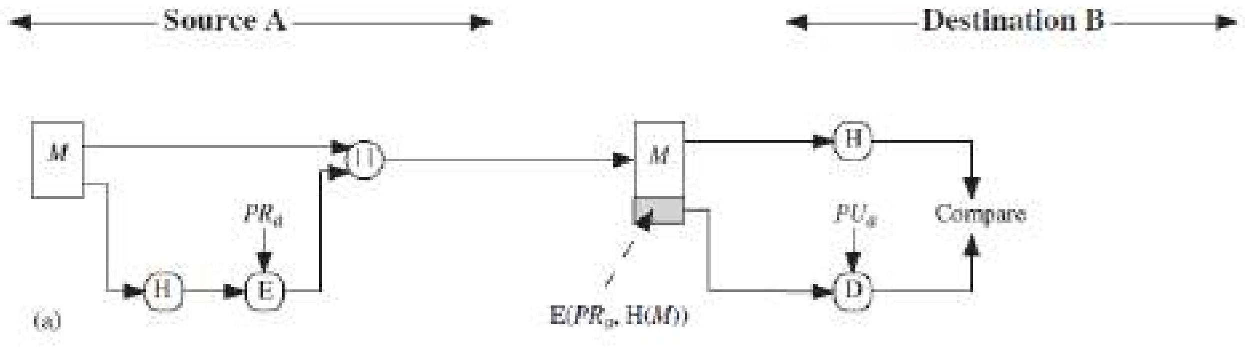
— More commonly, message authentication is achieved using a **message authentication code** (MAC), also known as a **keyed hash function**.

Digital Signatures

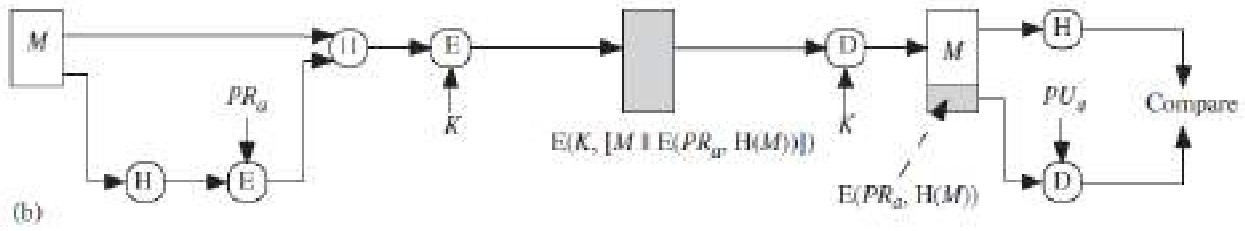
- Another important application, which is similar to the message authentication application, is the digital signature.
- The operation of the digital signature is similar to that of the MAC.
  - In the case of the digital signature, the *hash value of a message is encrypted with a user's private key.*
  - Anyone *who knows the user's public key can verify the integrity of the message that is associated with the digital signature.*
- In this case, an attacker who wishes to alter the message would need to know the user's private key.

Simplified Examples of Digital Signatures

a) **The hash code is encrypted, using public-key encryption with the sender's private key**. A
   - Provides authentication.
   - It also provides a digital signature, because only the sender could have produced the encrypted hash code.
   - In fact, this is the essence of the digital signature technique.



b) **If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key**.
   - This is a common technique.



Other Applications

- ❖ Hash functions are commonly used to create a **one-way password file**.
  - — When a user enters a password, the hash of that password is compared to the stored hash value for verification.
- ❖ Hash functions can be used for **intrusion detection** and **virus detection**
  - — Store H(F) for each file on a system and secure the hash values (e.g., on a CD-R that is kept secure). One can later determine if a file has been modified by recomputing H(F). An intruder would need to change F without changing H(F).
- ❖ A cryptographic hash function can be used to construct a **pseudorandom function (PRF)** or a **pseudorandom number generator (PRNG).**
  - — A common application for a hash-based PRF is for the generation of symmetric keys.

Message Authentication, Message Authentication Functions, and Message Authentication Codes

Message Authentication

- Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered.

Message Authentication Requirements

In the context of communications across a network, the following attacks can be identified.

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis**: Discovery of the pattern of traffic between parties. In a connection- oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.
4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing modification**: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
7. **Source repudiation:** Denial of transmission of message by source.
8. **Destination repudiation:** Denial of receipt of message by destination

How to deal with such attacks?

- Measures to deal with the first two attacks are in the realm of message confidentiality and are dealt with in previous chapters (Symmetric and Asymmetric Ciphers).
- Measures to deal with items (3) through (6) in the foregoing list are generally regarded as message authentication.
- Mechanisms for dealing specifically with item (7) come under the heading of digital signatures.
- Generally, a digital signature technique will also counter some or all of the attacks listed under items (3) through (6).
- Dealing with item (8) may require a combination of the use of digital signatures and a protocol designed to counter this attack.

In summary, message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

Message Authentication Functions

- ➢ Any message authentication or digital signature mechanism has two levels of functionality.
    - ▪ At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.
    - ▪ This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

This section is concerned with the types of functions that may be used to produce an authenticator. These may be grouped into three classes.

Types of functions which are used to produce authenticator

- **Hash function:** A function that maps a message of any length into a fixed length hash value, which serves as the authenticator. (As discussed above)
- **Message encryption:** The ciphertext of the entire message serves as its authenticator
- **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

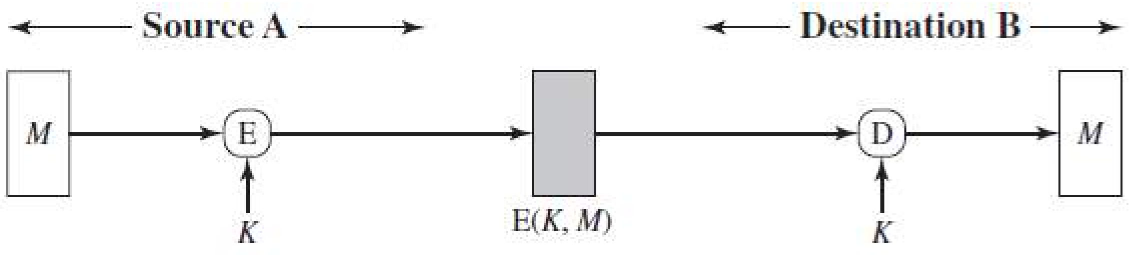
We have already discussed about Hash function.

<u>Message Encryption</u>

- Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

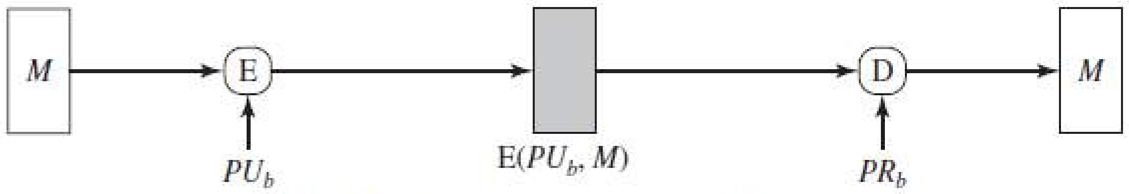
<u>Basic Uses of Message Encryption</u>

a) Symmetric encryption: confidentiality and authentication



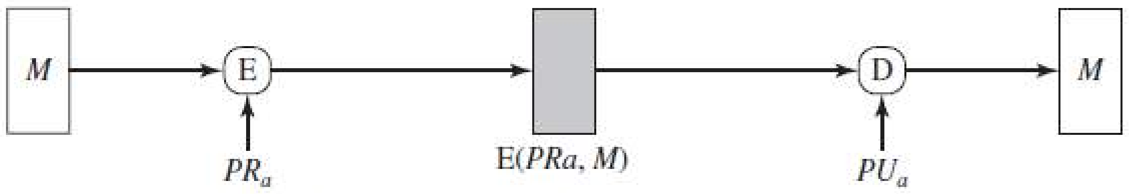
(a) Symmetric encryption: confidentiality and authentication

b) Public-key encryption: confidentiality



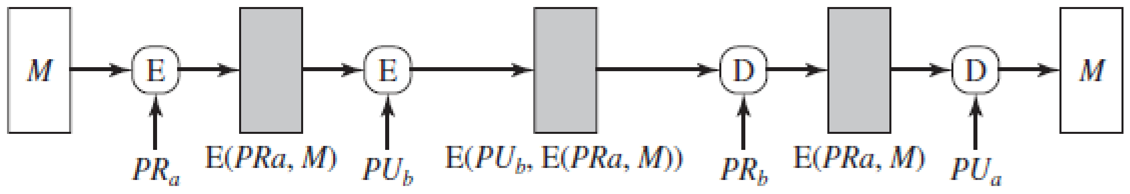
(b) Public-key encryption: confidentiality

c) Public-key encryption: authentication and signature



(c) Public-key encryption: authentication and signature

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



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

Message Authentication Code (MAC)

- An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum** or **MAC**, that is appended to the message.
- This technique assumes that two communicating parties, say **A** and **B**, share a common secret key **K**. When **A** has a message to send to **B**, it calculates the **MAC** as a ***function of the message and the key:***

  MAC = C(K, M)

  where

  M = input message; **C =** MAC function; **K =** shared secret key;

  MAC = message authentication code;

  - The message plus MAC are transmitted to the intended recipient.
  - The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC.

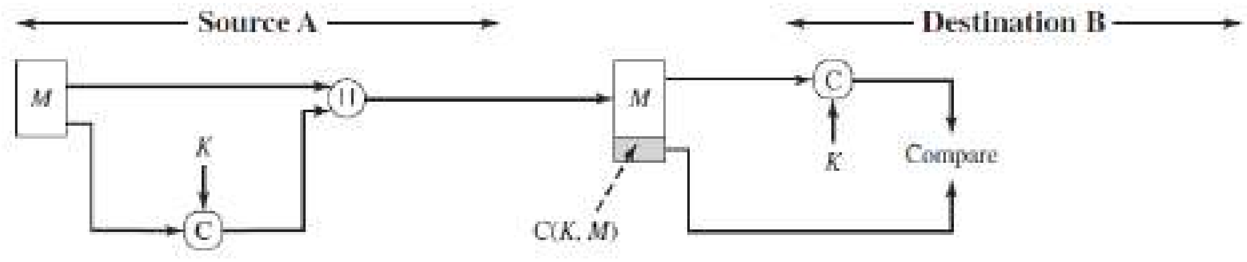


Figure: Message Authentication using MAC

❖ If we assume that **only the receiver and the sender know the identity of the secret key**, and **if the received MAC matches the calculated MAC**, then

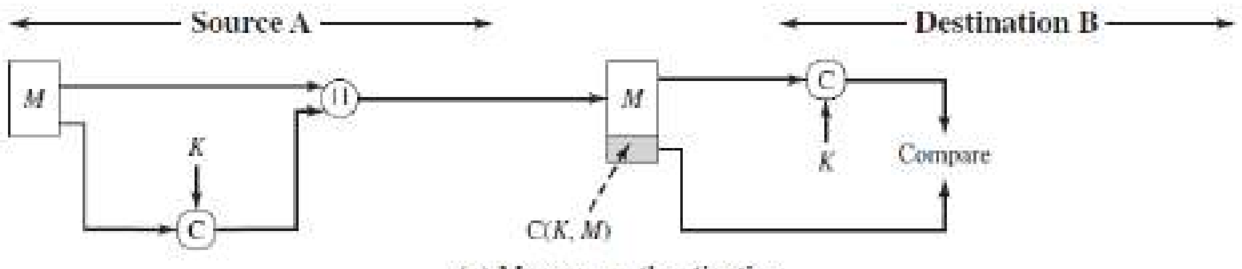
1. ***The receiver is assured that the message has not been altered***.
   - If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC.
   - Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
2. ***The receiver is assured that the message is from the alleged sender***.
   - Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
3. ***If the message includes a sequence number*** (such as is used with HDLC, X.25, and TCP), ***then the receiver can be assured of the proper sequence***
   - Because an attacker cannot successfully alter the sequence number.

Note:

- A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must be for decryption.
- In general, the MAC function is a many-to-one function.
- The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys.
- If an n-bit MAC is used, then there are 2^n possible MACs, whereas there are N possible messages with $N > 2^n$.
- Furthermore, with a k-bit key, there are 2^k possible keys.

<u>Basic Uses of Message Authentication code (MAC)</u>

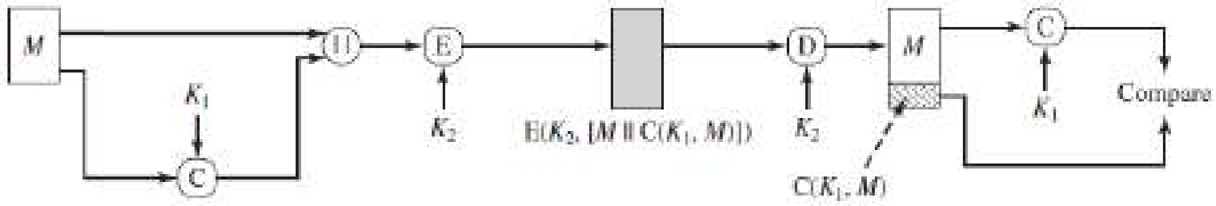
a) Message authentication



(a) Message authentication

- ✓ Provides authentication but not confidentiality, because the message as a whole is transmitted in the clear.

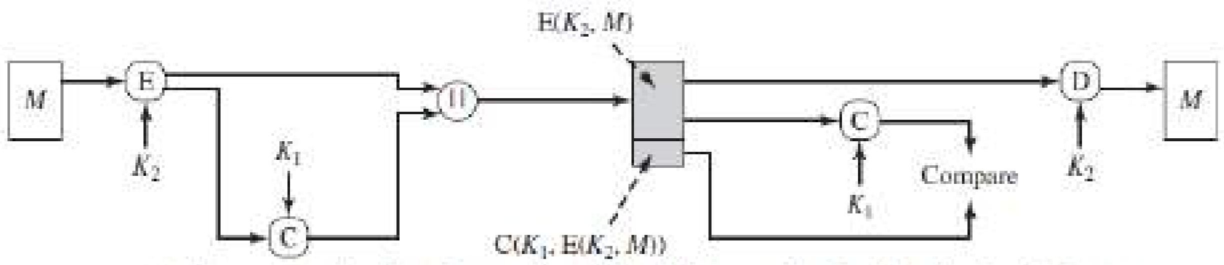
b) Message authentication and confidentiality; authentication tied to plaintext



(b) Message authentication and confidentiality; authentication tied to plaintext

- ✓ Confidentiality can be provided by performing message encryption after the MAC algorithm

c) Message authentication and confidentiality; authentication tied to ciphertext



(c) Message authentication and confidentiality; authentication tied to ciphertext

- ✓ Confidentiality can be provided by performing message encryption before the MAC algorithm

Note: Typically, it is preferable to tie the authentication directly to the plaintext (as in (b)).

Because symmetric encryption will provide authentication and because it is widely used with readily available products, why not simply use this instead of a separate message authentication code?

⇒ The situations in which a message authentication code is used:

1. **There are a number of applications in which the same message is broadcast to a number of destinations**.
   - Examples are notification to users that the network is now unavailable or an alarm signal in a military control center.
   - It is cheaper and more reliable to have only one destination responsible for monitoring authenticity.
   - Thus, the message must be broadcast in plaintext with an associated message authentication code.
   - The responsible system has the secret key and performs authentication.
   - If a violation occurs, the other destination systems are alerted by a general alarm.
2. **Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages**.
   - Authentication is carried out on a selective basis, messages being chosen at random for checking.
3. **Authentication of a computer program in plaintext is an attractive service**.
   - The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources.
   - However, if a message authentication code were attached to the program, it could be checked whenever assurance was required of the integrity of the program.
4. **For some applications, it may not be of concern to keep messages secret, but it is important to authenticate messages**.
   - An example is the Simple Network Management Protocol Version 3 (SNMPv3), which separates the functions of confidentiality and authentication.
   - For this application, it is usually important for a managed system to authenticate incoming SNMP messages, particularly if the message contains a command to change parameters at the managed system.

— On the other hand, it may not be necessary to conceal the SNMP traffic.

5. **Separation of authentication and confidentiality functions affords architectural flexibility**.
   - For example, it may be desired to perform authentication at the application level but to provide confidentiality at a lower level, such as the transport layer.
6. **A user may wish to prolong the period of protection beyond the time of reception and yet allow processing of message contents.**
   - With message encryption, the protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.

❖ Finally, note that the **MAC does not provide a digital signature**, because both sender and receiver share the same key.

Requirements for Message Authentication Codes

❖ Assume that an opponent knows the MAC function but does not know K. Then the MAC function should satisfy the following requirements.

1. If an opponent observes **M** and **MAC(K, M),** it should be computationally infeasible for the opponent to construct a message **M'** such that

$$\mathbf{MAC(K, M') = MAC(K, M)}$$

2. **MAC(K, M)** should be uniformly distributed in the sense that for randomly chosen messages, **M** and **M'**, the probability that **MAC(K, M) = MAC(K, M')** is $\mathbf{2^{-n}}$, where **n** is the number of bits in the tag.

3. Let **M'** be equal to some known transformation on **M**. That is, **M' = f(M).**

   For example, **f** may involve inverting one or more specific bits. In that case,

$$\mathbf{Pr\,[MAC(K, M) = MAC(K, M')] = 2^{-n}}$$

Message Digest

Secure Hash Algorithms

Digital Signatures

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

❖ The digital signature provides a set of security capabilities that would be difficult to implement in any other way.

❖ Following figure is a generic model of the process of making and using digital signatures.

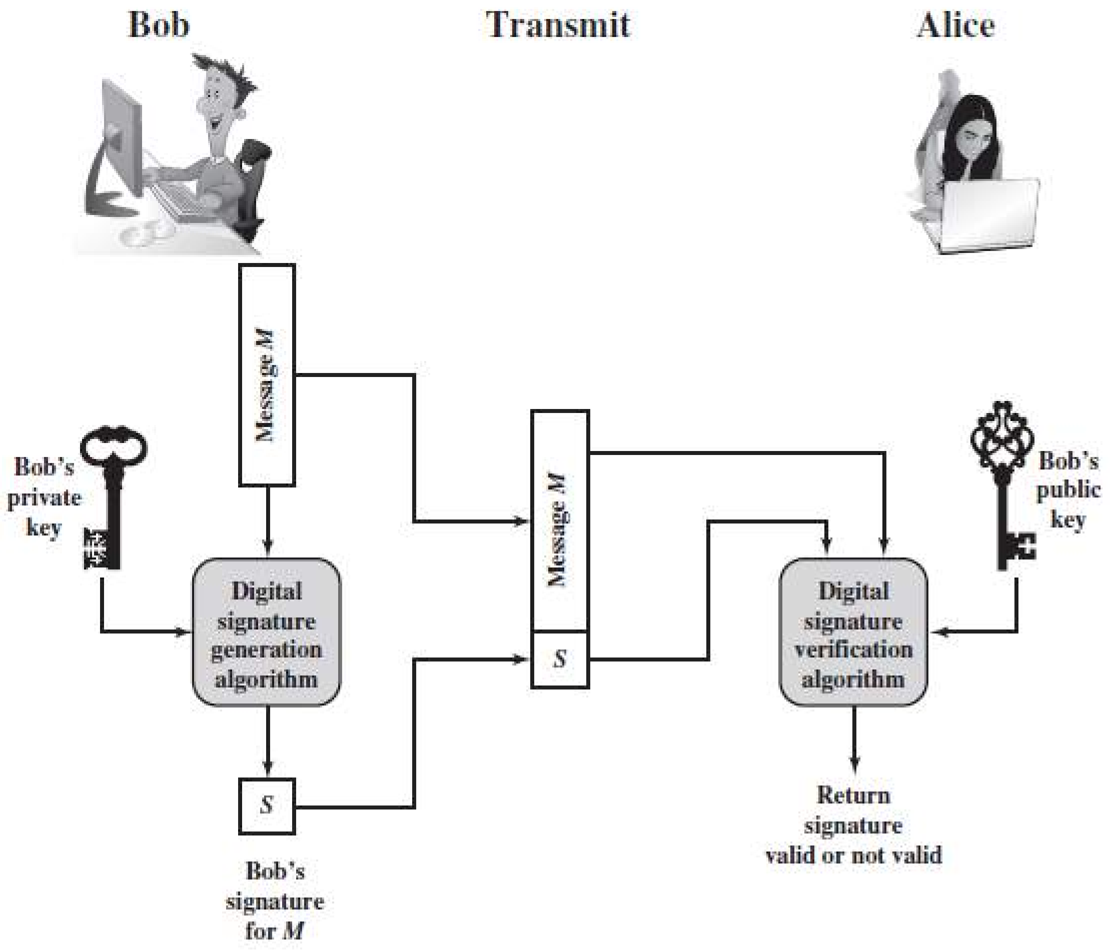
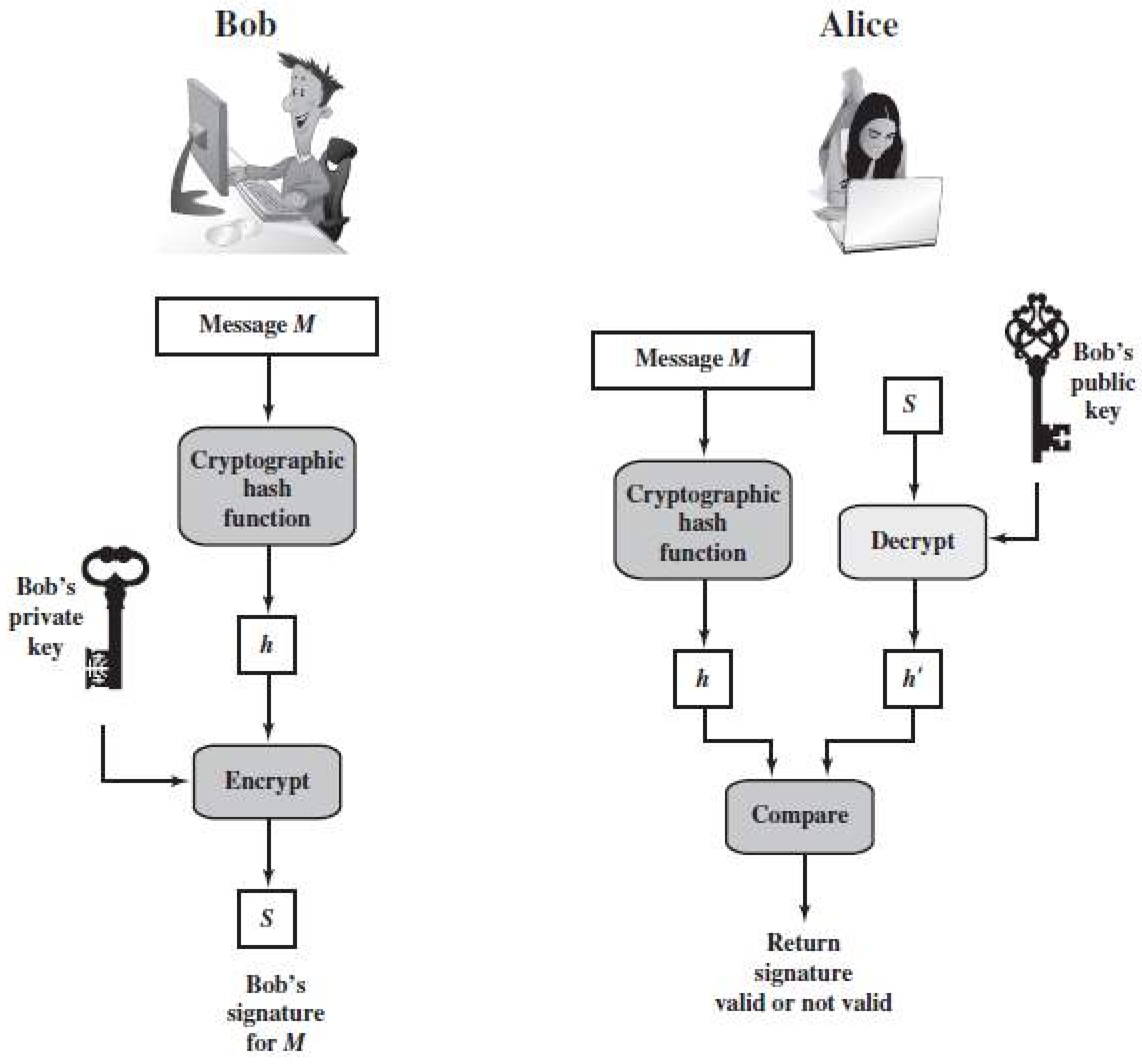

Bob
Transmit
Alice
Message M
Bob's private key
Digital signature generation algorithm
S
Bob's signature for M
Message M
S
Digital signature verification algorithm
Bob's public key
Return signature valid or not valid

Figure: Generic Model of Digital Signature Process

Bob can sign a message using a digital signature generation algorithm. The inputs to the algorithm are the message and Bob's private key. Any other user, say Alice, can verify the signature using a verification algorithm, whose inputs are the message, the signature, and Bob's public key.

Simplified Depiction of Essential Elements of Digital Signature Process



⇒ Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two are possible.

⇒ In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the **digital signature**.

Properties

The digital signature must have the following 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.

Thus, the digital signature function includes the authentication function.

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 and Arbitrated Digital Signature

⇒ The **Direct Digital Signature** includes only two parties one to send message and other one to receive it.

- According to direct digital signature both parties trust each other and knows there public key.

⇒ **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.

⇒ Need time-stamps and timely key revocation.

❖ The **Arbitrated Digital Signature** includes three parties in which one is sender, second is receiver and the third is arbiter who will become the medium for sending and receiving message between them.

❖ The problems associated with direct digital signatures can be addressed by using an arbiter, in a variety of possible arrangements.

❖ The arbiter plays a sensitive and crucial role in this sort of scheme, and all parties must have a great deal of trust that the arbitration mechanism is working properly.

❖ These schemes can be implemented with either private or public-key algorithms, and the arbiter may or may not see the actual message contents.

- ➢ involves use of arbiter
  - o validates any signed message
  - o 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

Digital Signature Standard (DSS)

➢ DSA is the US Govt approved signature scheme, which is designed to provide strong signatures without allowing easy use for encryption.

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

➢ The DSS was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme.

- There was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2, which incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.
    - 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

Digital Signature Algorithm (DSA)

- The DSA signature scheme has advantages, being both smaller (320 vs 1024bit) and faster (much of the computation is done modulo a 160 bit number), over RSA. Unlike RSA, it cannot be used for encryption or key exchange..
- The DSA is based on the difficulty of computing discrete logarithms, and is based on schemes originally presented by ElGamal and Schnorr
    - 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

The DSA Approach

⇒ The DSA 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.

⇒ The DSA 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.

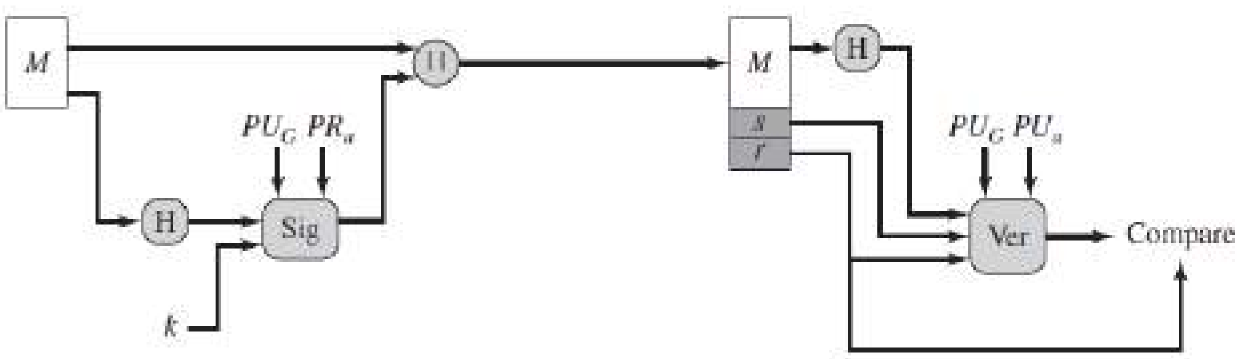
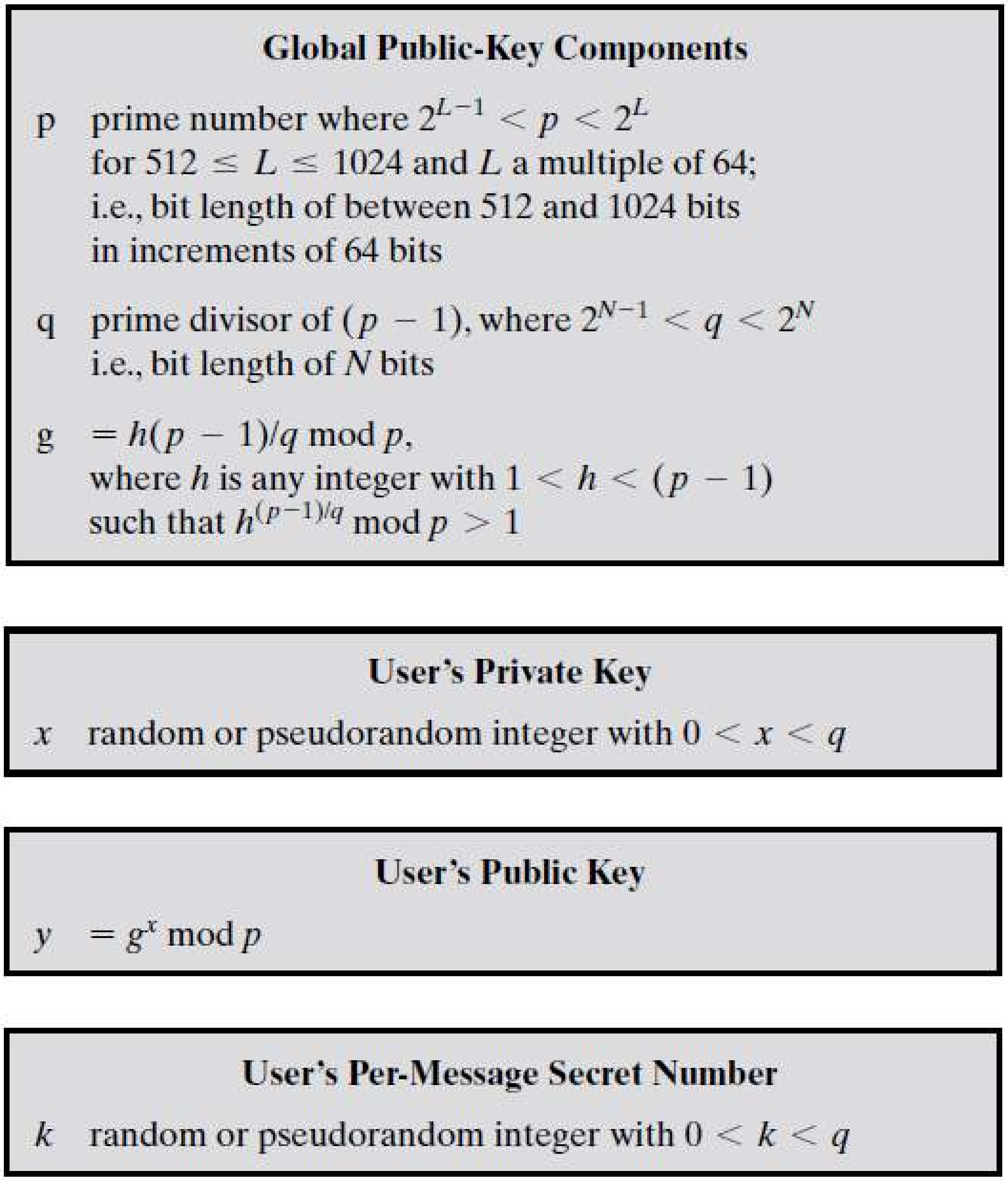


Figure: DSA Approach to Digital Signature

The Digital Signature Algorithm



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

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

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

$$\text{Signature} = (r, s)$$

Verifying

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

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

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

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

$$\text{TEST: } v = r'$$

Here,

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

- ❖ There are three parameters that are public and can be common to a group of users.
  - An N-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 $h^{(p-1)/q} \bmod p$, where **h** is an integer between 1 and **(p - 1)** with the restriction that **g** must be greater than **1**

- With these numbers in hand, each user selects a private key and generates a public key. The private key **x** must be a number from **1** to **(q - 1)** and should be chosen randomly or pseudorandomly.
- The public key is calculated from the private key as $\mathbf{y = g^x \bmod p}$. The calculation of **y** given **x** is relatively straightforward.
- However, given the public key **y**, it is believed to be computationally infeasible to determine **x**, which is the discrete logarithm of **y** to the base **g**, mod **p**.
- To create a signature, a user calculates two quantities, **r** and s, that are functions of the public key components **(p, q, g),** the user's private **key (x),** the hash code of the message **H(M),** and an additional integer **k** that should be generated randomly or pseudorandomly and be unique for each signing.
- Let $\mathbf{M'}$**,** $\mathbf{r'}$, and $\mathbf{s'}$ be the received versions of **M, r**, and **s**, respectively.
- Verification is performed using the formulas shown above. The receiver generates a quantity **v** that is a function of the public key components, the sender's public key, and the hash code of the incoming message. If this quantity matches the **r** component of the signature, then the signature is validated.

The RSA 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.

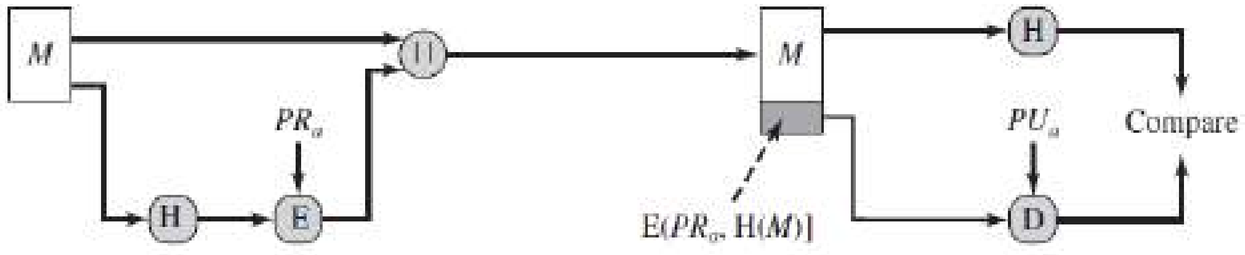


Figure: RSA approach to Digital Signature