Information Systems Security

Lecture 5

Message Authentication and Digital Signature

Dr. En. Bader Ahmad

Outline

1. Message authentication
2. Authentication functions
   - 2.1. Message Encryption
   - 2.2. Hash functions
   - 2.3. MAC
3. Digital signature
   - 3.1. Arbitrated digital signature
   - 3.2. True digital signature
   - 3.3. RSA – based signature
   - 3.4. ElGamal – based signature (DSA)
4. Conclusion

1. Message Authentication

- ***Message authentication*** is a procedure to verify that received messages come from the pretended source and have not been altered.
  - Also called ***data origin authentication***
  - It provides integrity
  - Entity authentication (identification) is similar but entities are online.
- Message Authentication can thwart the following attacks:
  - Masquerade
  - Content modification
  - Sequence modification
  - Timing modification: Message delay or replay

2. Authentication functions

- Message authentication produces an ***authenticator***: a value to be used to authenticate a message.
- Three types of functions that may be used to produce an authenticator:
  1. **Message encryption**: The ciphertext of the entire message serves as its authenticator.
  2. ***Hash function***: a public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator .
  3. ***Message Authentication Code (MAC)***: a public function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

2.1. Message encryption

- Message encryption provides authentication:
  - **Symmetric encryption**: if the encryption/decryption key is not known to any other party (except the sender and receiver).
  - **Asymmetric encryption**
    - the sender should uses its private key to encrypt the message,
    - the sender's public key is then used to decrypt the message.
    - This helps providing only authentication
- If we need authentication and confidentiality, then the sender should encrypt the message twice:
  - 1st: with its private key (authentication)
  - 2nd: with the receiver's public key (confidentiality)

Message encryption

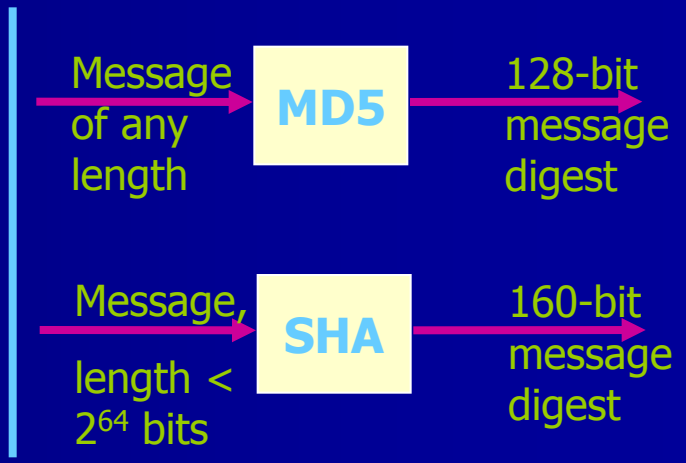
- Some careful considerations are needed here:
  - How does B recognize a meaningful message from an arbitrary sequence of bits?
    - He can apply the decryption key to **any** sequence of bits he receives
  - This is not necessarily easy task if the message is some sort of binary file
  - **Immediate idea of attack**: send arbitrary bit sequences to disrupt the receiver –he will try to figure out the meaning of that bit sequence
- Defense against this type of attack: add to the message a certain structure such as an error-correcting code (e.g., check-sum bits) and then encrypt the whole file
  - B will detect illegitimate messages because they will not have the required structure

2.2. Hash function

- **A hash function** produces a fixed-size output for a variable-size message *m* as an input.
  - It is denoted H(*m*) (the ***hash code***).
  - A hash code is also referred as ***message digest*** or ***hash value***.
- It takes as input only the message itself.
- It is a one-way function.
- H(*m*) provides error-detection capability.

Hash function

- Examples of hash algorithms
  - MD5: ([RFC 1321]*, 1992)
    - Input: any message of arbitrary length
    - Output: 128-bit message digest
  - SHA-1: ([FIPS 180]**, 1993, [RFC 3174])
    - Input: any message < 2^{64} bits
    - Output: 160-bit message digest
  - *SHA is more secure than MD5 but slower*



*: http://www.ietf.org **: US Federal Information Processing Standard

Hash function

- Stronger than MD5 because of longer message digest
- Slower than MD5 because of more rounds
- Hash functions are much faster than symmetric ciphers
- No known attacks
  - Secret design criteria
- Variants of SHA-1 with longer message digests have also been proposed: SHA-256, SHA-384, SHA-512 (n-bit hash for SHA-n)

Security of hash functions

- A hash is shorter than the message, collisions are inevitable,
  - We just want them to be hard to find.
- Suppose that we have a hash function that has a hash of just 2 bits:
  - there are only four possible hashes: 00, 01, 10 or 11
- What is the probability that:

  hash (Alice owes Bob 10 SP) = hash (Alice owes Bob 100 SP)?
- Suppose the hash is 10 bits long
  - Since there are only ~1000 different possible values of the hash, there is a **very good chance** that there will be at least one match
- In practice a common practical length for a hash is about 160 bits.

Security of hash functions

- Requirements for a hash function
  - H can be applied to a message of any size
  - H produces fixed-length output
  - Computationally easy to compute H(M)
  - Computationally infeasible to find M such that H(M)=h, for a given h
  - Computationally infeasible to find M' such that H(M')=H(M), for a given M
  - Computationally infeasible to find M,M' with H(M)=H(M')
- Note: the hash function is not considered secret –some other means are required to protect it
- Note 2: Hash function plus secrecy (key) gives a MAC

2.3. Message Authentication Code

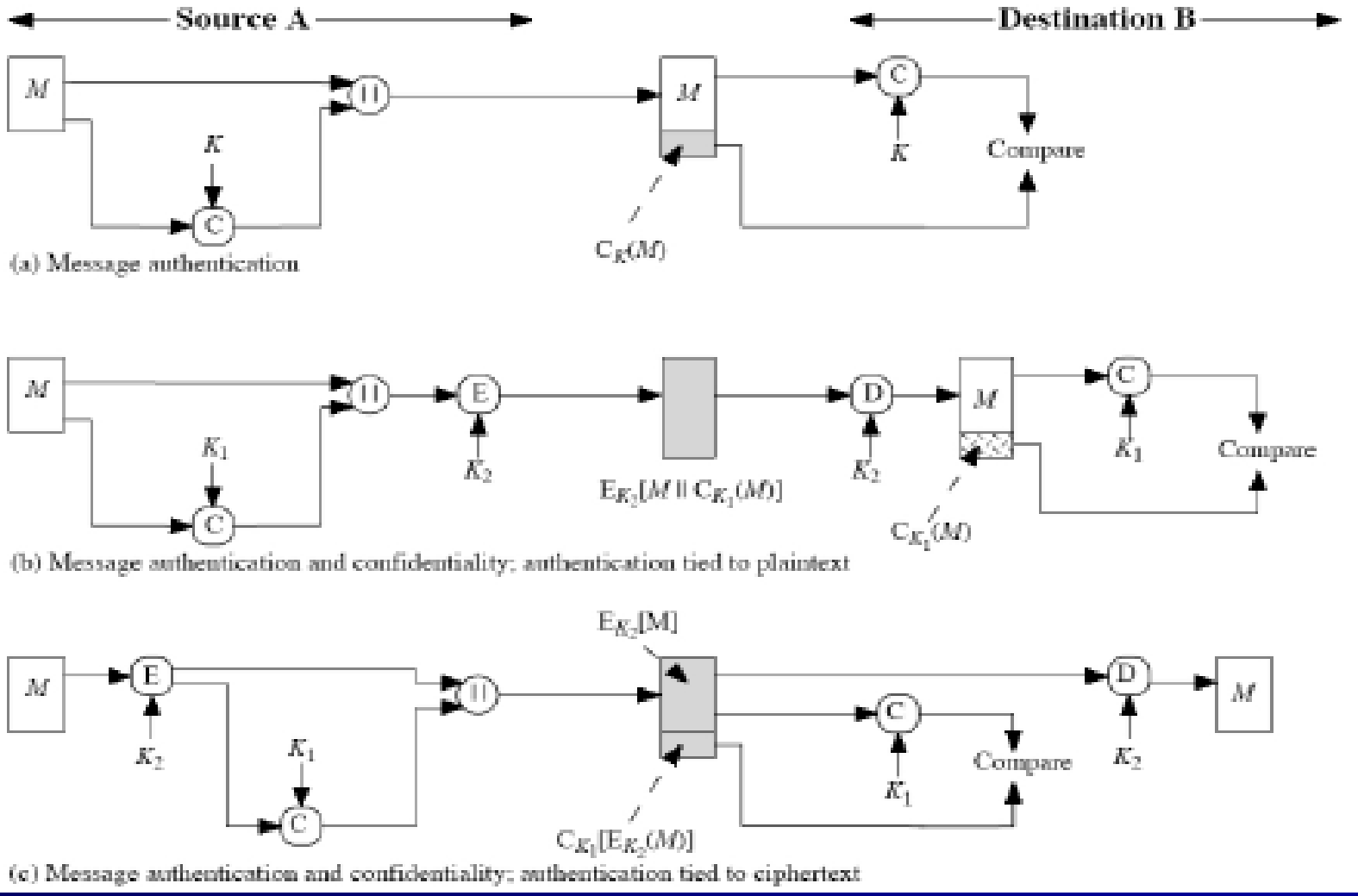
- **MAC** is a technique to provide authentication using a shared secret key to generate a small fixed-size block of data
  - The MAC is also known as a ***cryptographic checksum***
  - It is appended to the message.
- A MAC can be viewed as a hash function with a secret key.
- If A wants to send an authentic message *m* to B, using MAC:
  - A and B must share a secret key, *K*
  - A computes the MAC as a function of *m* and *K*,
    - i.e., MAC = $C_K(M)$
  - A sends B *m* plus the MAC

Message Authentication Code

- MACs assure:
  1. Message hasn't been altered during transmission.
  2. Message coming from the pretended sender.
  3. Sequence number hasn't been altered during transmission if the message includes a sequence number.

- Examples of MACs:
  - HMAC ([RFC 2104]):
    - Used in IPsec and SSL

Basic uses of MAC



Basic uses of MAC

$A \rightarrow B: M \| C_K(M)$

- Provides authentication
  - Only A and B share K

(a) Message authentication

$A \rightarrow B: E_{K_2}[M \| C_{K_1}(M)]$

- Provides authentication
  - Only A and B share K_1
- Provides confidentiality
  - Only A and B share K_2

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

$A \rightarrow B: E_{K_2}[M] \| C_{K_1}(E_{K_2}[M])$

- Provides authentication
  - Using K_1
- Provides confidentiality
  - Using K_2

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

3. Digital signature

A ***digital*** signature is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, such that the sender cannot deny having sent the message (authentication and non-repudiation) and that the message was not altered in transit (integrity). Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery or tampering.

We use the term:

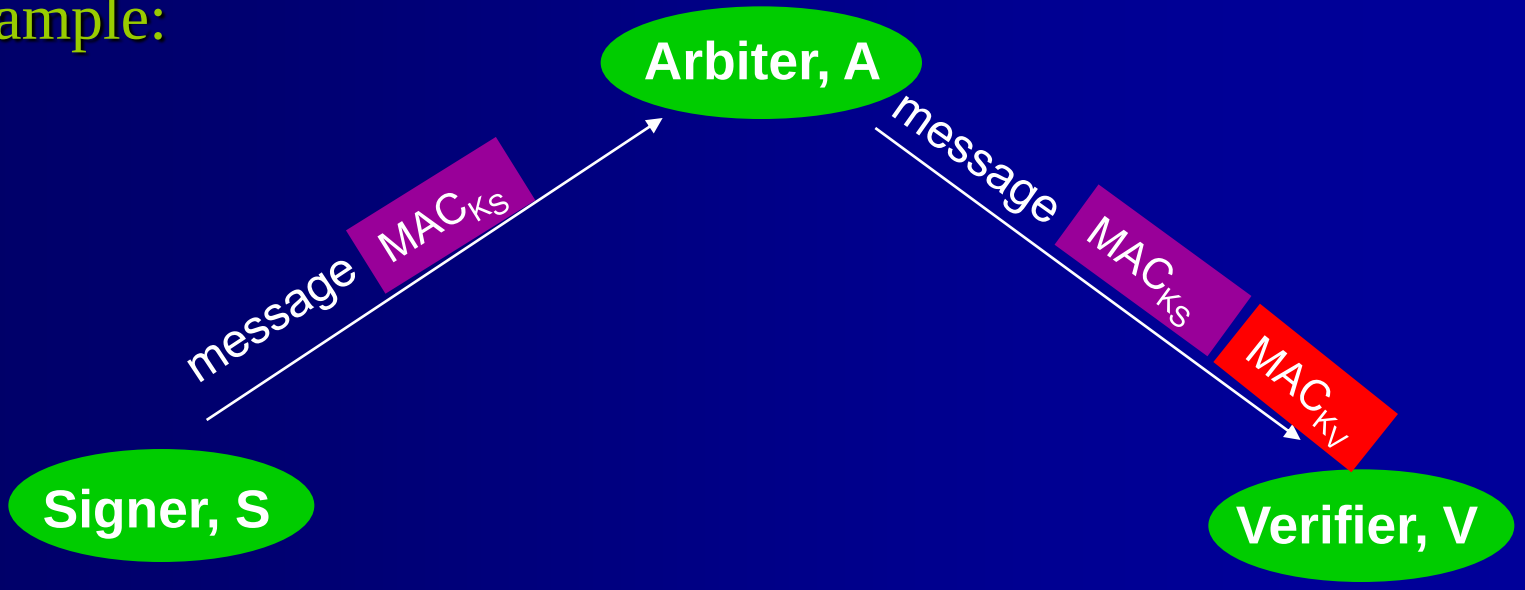
- ***signer*** for an entity who creates a digital signature,
- ***verifier*** for an entity who receives a signed message and attempts to check whether the digital signature is "correct" or not.

Digital signature

- A digital signature on a message provides:
  - Message authentication: message's origin is known + integrity
  - Non-repudiation
- the digital signature takes as input parameters the message itself and a secret value, **known only to the signer**.
- A digital signature must be:
  - Easy to compute by the signer.
  - Easy to verify by anyone.
  - Hard to compute by anyone except the signer.
- There are 2 types of digital signatures:
  - ***arbitrated digital signature,*** and
  - ***true digital signature***

3.1. Arbitrated digital signature

- Arbitrated digital signature is based on a trusted third party (arbiter). There are 2 types:
  - Based on symmetric key
  - Based on public-key
  - Example:



  - K_s shared between A and S, k_V shared between A and V

3.2. True digital signature

- The vast majority of digital signature techniques do not involve communication 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".

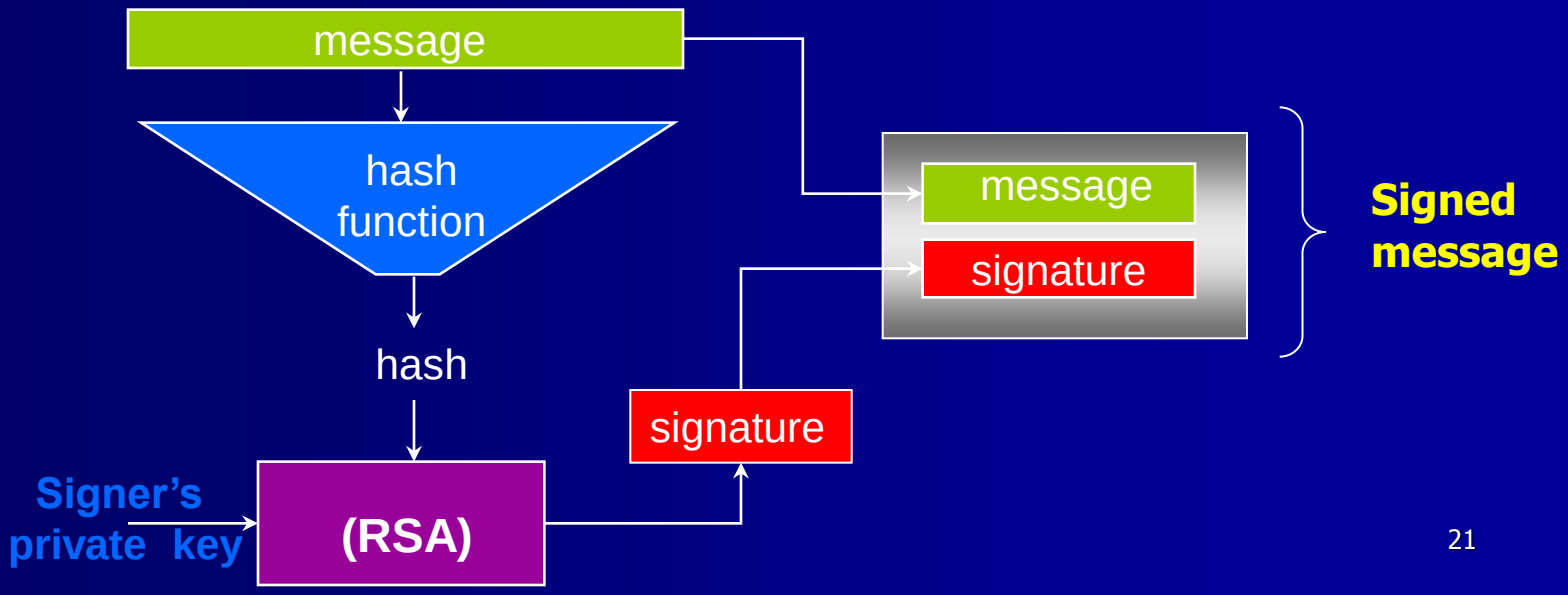
3.2. True digital signature

- A digital signature may be formed by:
  - encrypting the entire message with the signer's private key,
  - encrypting a hash code of the message with sender's private key.
  - Signer's public key should be available to the verifier.

- We'll see two schemes
  - RSA-based scheme
  - ElGamal- based scheme (or DSA)

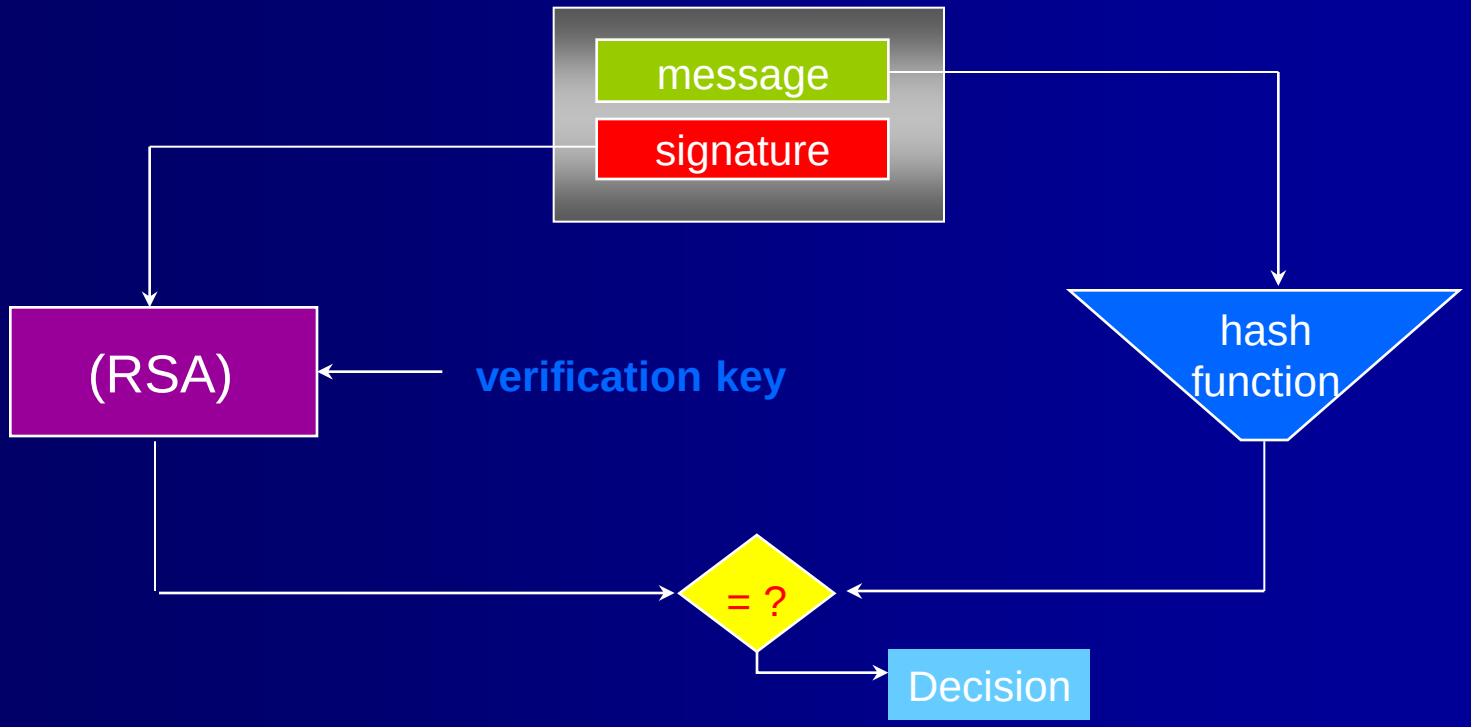
3.3. RSA Signature

- RSA signature is similar to RSA encryption with inverse roles of keys, i.e., for signing, the sender's private key is used and for verification, the sender's public key is used.
- Example: RSA is used to encrypt the hashed code with the sender's private key.



Verification of a RSA Signature

- The verifier decrypts the signature with the sender's public key and then compares the result with the message's hash code.



RSA is special

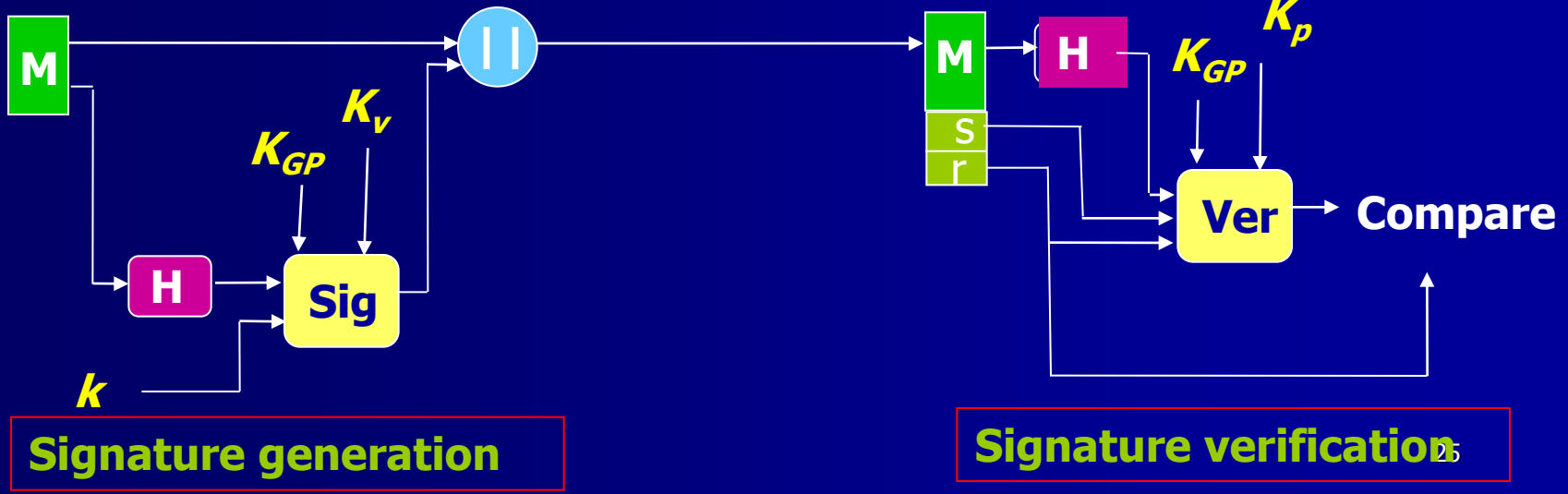
- You cannot obtain a digital signature scheme by swapping the roles of the private and public keys of any public key cipher system
- You cannot obtain a public key cipher scheme by swapping the roles of the signature and verification keys of any digital signature scheme
- Key separation: In real applications you should avoid using the same RSA key pair for both encryption and for digital signatures.
  - The reason is that good key management follows a principle known as **key separation**, where any cryptographic key has a specific role and is not used for different purposes.

Digital Signature Algorithm (DSA)

- In 1991, the DSA has been published by the US NIST (National Institute of Standardizations and Technology) and become a US FIPS-186 under the name DSS (***Digital Signature Standard***).
- DSS is the 1st digital signature scheme to be recognized by any government.
- DSS is a **variant of ElGamal** signature scheme
- DSS makes use of SHA

3.4. DSS Approach

- DSS depends on:
  - A hash function H
  - a random number k, (used once).
  - The sender's key pair (K_v: **private,** K_p: **public**)
  - Global Public parameters, K_{GP}



DSS parameter generation

- Global Public (GP) parameters (q,p,g):
  - q: a 160-bit prime number
  - p: a prime number; such that $2^{512} \leq p \leq 2^{1024}$ and $q|(p-1)$
  - g: $= h^{(p-1)/q} \bmod p$; and $g > 1$
    - h is an integer: $1 < h < p-1$
  - (q,p,g) can be common to a group of users
- User's private key $\boldsymbol{K_v}$
  - x: A random integer, $1 < x < q$
- User's public key $\boldsymbol{K_p}$
  - $\boldsymbol{y = g^x \bmod p}$

DSS signature generation

- Signing: if an entity A wants to send a signed message m to another entity B.
  - Assume that (p,q,g): the global public parameters, x: A's private key, and y: A's public key.
  - 1st A randomly picks an integer k: $1 < k < q$
  - 2nd A computes r and s
    - $r = (g^k \bmod p) \bmod q$
    - $s = k^{-1}(H(m) + xr) \bmod q$
  - The signature is (r,s)
  - A sends to B $[m \,||\, (r,s)]$

DSS signature verification

- Verification: assume that B receives [m'+(r',s')], *i.e.*, m', r' ,s' are the received versions of m, r, s.
  - Assume that B has an authentic copy of A's public key, y, and GP parameters (p, q, g).
  - 1st, B computes w, u_1 , u_2 such that :
    - $w = (s')^{-1} \bmod q$,
    - $u_1 = w.\mathrm{H}(m') \bmod q$,
    - $u_2 = (r')\, w \bmod q$
  - 2nd B computes $v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$
  - 3rd B checks if $v = r'$ then signature is authentic

4. Conclusion

- Message encryption, MAC, and digital signature ensure message authentication
- Digital signature: RSA and El Gamal –based (DSS)
- For RSA, the encryption and decryption are identical, so the signature and verification processes are also identical.
- For DSA, signature and verification processes are different.
  - requires a random number generator (extra processing), whereas RSA does not.
  - always produces a fixed length 320-bit signature.
- For RSA the signature block and the modulus have the same size, which increases as the security level increases.


Questions?