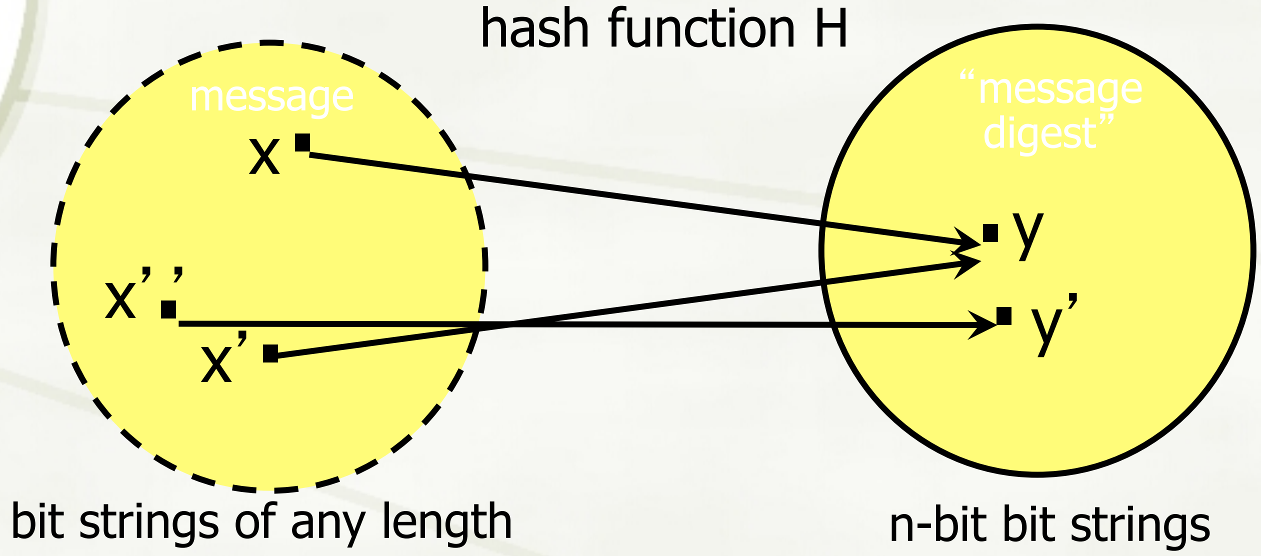
Public-Key Cryptography and Message Authentication

Ola Flygt
Linnaeus University, Sweden
http://w3.msi.vxu.se/users/ofl/
Ola.Flygt@lnu.se
+46 470 70 86 49

OUTLINE

- Secure Hash Functions and HMAC
- Public-Key Cryptography Principles
- Public-Key Cryptography Algorithms
- Approaches to Message Authentication
- Digital Signatures
- Key Management

Hash Functions: Main Idea



- Hash function H is a lossy compression function
  - Collision: H(x)=H(x') for some inputs x≠x'
- H(x) should look "random"
  - Every bit (almost) equally likely to be 0 or 1
- Cryptographic hash function must have certain properties

Simple Hash Function

	bit 1	bit 2	• • •	bit n
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	• • •	• • •	• • •	• • •
block m	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

- Hash code is bitwise XOR on the columns
- One-bit circular shift on the hash value after each block is processed would improve the code

Secure HASH Functions

- Purpose of the HASH function is to produce a ”fingerprint.
- Properties of a HASH function H :

1. H can be applied to a block of data at any size
2. H produces a fixed length output
3. H(x) is easy to compute for any given x.
4. For any given code h, it is computationally infeasible to find x such that H(x) = h
5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y)

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - nb. the algorithm is SHA, the standard is SHS
- based on design of MD4 with key differences
- produces 160-bit hash values
- In 2005 results on security of SHA-1 have raised concerns on its use in future applications

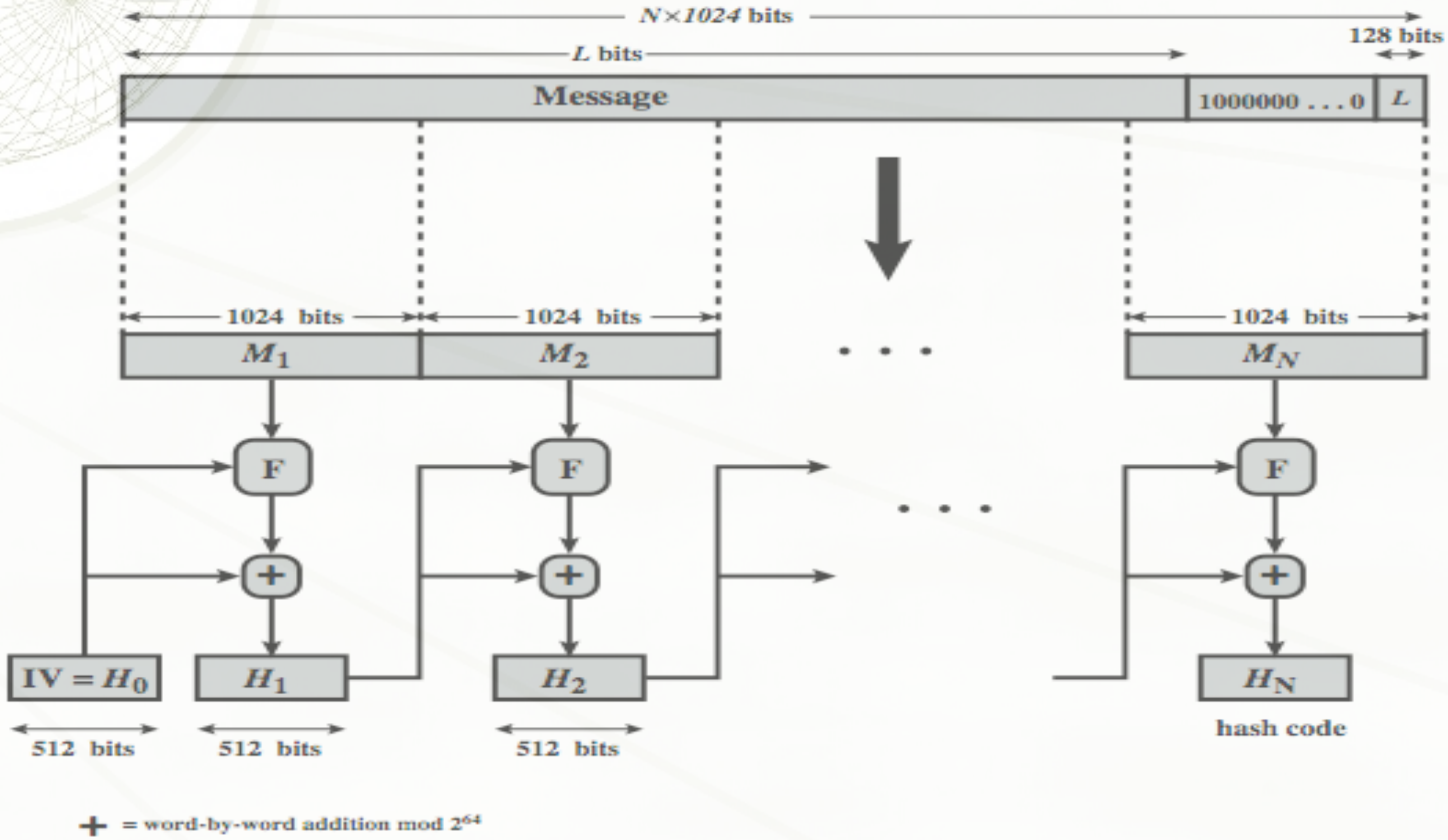
Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

SHA Versions

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message digest size	160	224	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024
Word size	32	32	32	64	64
Number of steps	80	64	64	80	80

SHA-512 Overview



SHA-512 Compression Function

- heart of the algorithm (the Merkle-Damgård construction)
- processing message in 1024-bit blocks
- consists of 80 rounds
  - updating a 512-bit buffer
  - using a 64-bit value derived from the current message block and a round constant based on cube root of first 80 prime numbers

One-Way

- Intuition: hash should be hard to invert
  - "Preimage resistance"
  - Let $h(x')=y\in\{0,1\}^n$ for a random x'
  - Given random y, it should be hard to find any x such that h(x)=y
- How hard?
  - Brute-force: try every possible x, see if h(x)=y
  - SHA-1 (common hash function) has 160-bit output
    - Suppose have hardware that'll do 2^{30} trials a pop
    - Assuming 2^{34} trials per second, can do 2^{89} trials per year
    - Will take 2^{71} years to invert SHA-1 on a random image

"Birthday Paradox"

- T people
- Suppose each birthday is a random number taken from K days (K=365) – how many possibilities?
  - K^T - samples with replacement
- How many possibilities that are all different?
  - $(K)_T = K(K-1)\dots(K-T+1)$ - samples without replacement
- Probability of no repetition?
  - $(K)_T/K^T \approx 1 - T(T-1)/2K$
- Probability of repetition?
  - $O(T^2)$

Collision Resistance

- Should be hard to find x≠x' such that h(x)=h(x')
- Brute-force collision search is $O(2^{n/2})$, <u>not</u> $O(2^n)$
  - n = number of bits in the output of hash function
  - For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$
- Example:
  - In a group of people, how many are required to have a 50% chance of two people having the same birthday?
  - Perhaps surprisingly the answer is only 23 even if there are 365 days on a year.
  - If there are more then 60 people the probability for a collision is almost 100%

SHA-1 weakness

- In 2005 Chinese cryptographer Xiaoyun Wang found collision-finding attacks that require only 2^{63} operations, rather than the 2^{80} operations stated by the birthday attack. Such an attack is feasible for a very well-funded adversary.
- Consequently, for many applications one needs to look for stronger hash functions. The SHA-2 family, including SHA-224, SHA-256 and SHA-512 already exists, but they are based on similar ideas as SHA-1.
- It seems that new ideas are needed in the design of hash functions, as well as careful analysis of what properties are needed for various applications.

SHA-3, the new kid in the block

- On November 2, 2007, NIST announced a public competition to develop a new cryptographic hash algorithm.
- Deadline for proposals was October 31, 2008.14 candidates selected for 2nd round in 2009, 6 for 3rd round in 2010.
- The winner, Keccak, announced on October 2, 2012.
- Keccak is now officially SHA-3; see www.nist.gov/hash-competition.

Keyed Hash Functions as MACs

- want a MAC based on a hash function
  - because hash functions are generally faster
  - crypto hash function code is widely available
- hash includes a key along with message
- original proposal:

```
KeyedHash = Hash(Key|Message)
```

  - some weaknesses were found with this
- eventually led to development of HMAC

HMAC Design Objectives

- use, without modifications, hash functions
- allow for easy replaceability of embedded hash function
- preserve original performance of hash function without significant degradation
- use and handle keys in a simple way.
- have well understood cryptographic analysis of authentication mechanism strength

HMAC

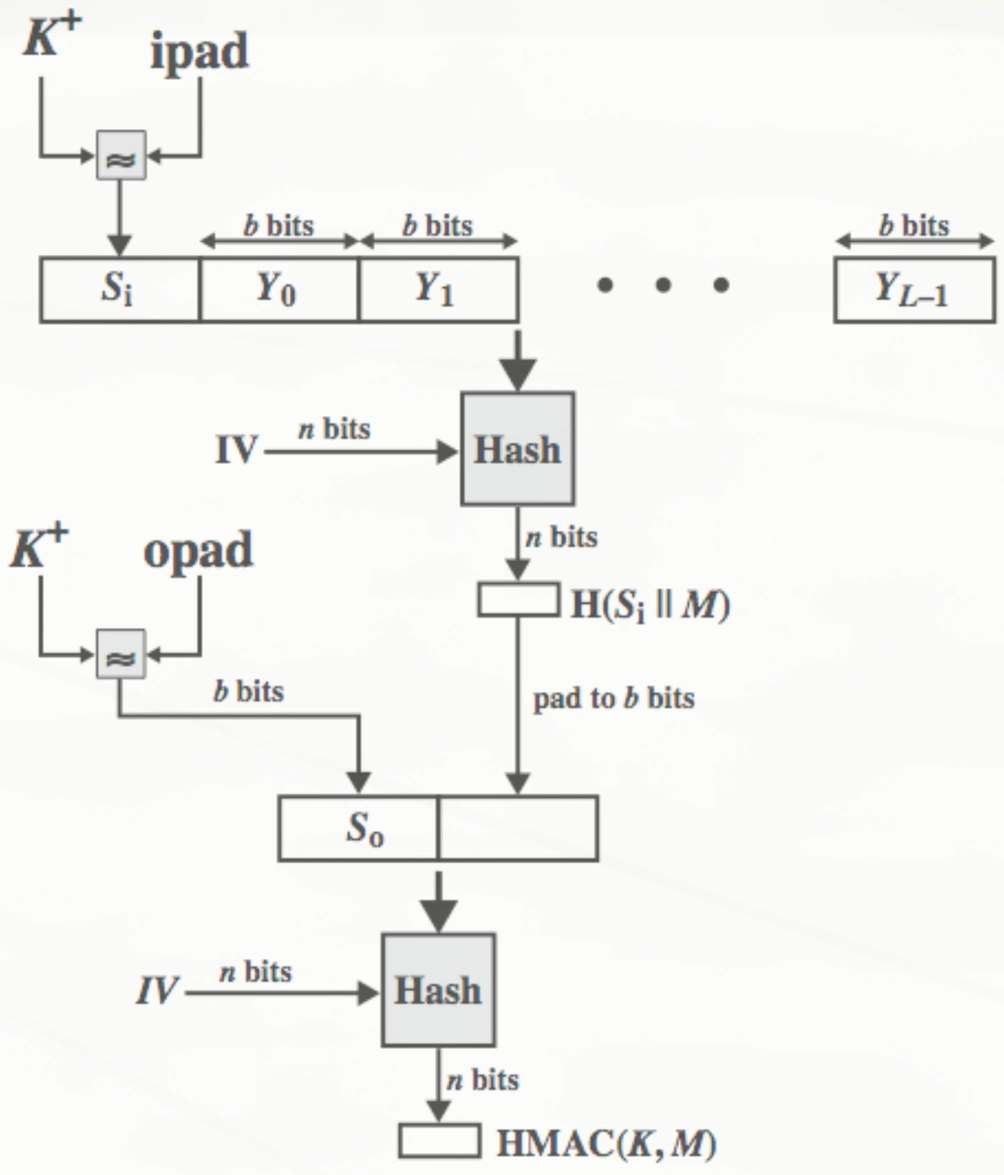
- specified as Internet standard RFC2104
- uses hash function on the message:

$\texttt{HMAC}_K\texttt{(M)= Hash[(K}^{+}\texttt{ XOR opad) ||}$

$\texttt{Hash[(K}^{+}\texttt{ XOR ipad) || M)]]}$

  - where K^{+} is the key padded out to size
  - `opad`, `ipad` are specified padding constants
- overhead is just 3 more hash calculations than the message needs alone
- any hash function can be used
  - eg. MD5, SHA-1, RIPEMD-160, Whirlpool

HMAC Overview



HMAC Security

- proved security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed, would need to observe a very large number of messages)
- choose hash function used based on speed verses security constraints

Private-Key Cryptography (revisited)

- traditional **private/secret/single key** cryptography uses **one** key
- shared by both sender and receiver
- if this key is disclosed communications are compromised
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender

Public-Key Cryptography the new idea

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
- **asymmetric** since parties are **not** equal
- uses clever application of number theoretic concepts to function
- complements **rather than** replaces private key crypto

Why Public-Key Cryptography?

- developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
  - **digital signatures** – how to verify a message comes intact from the claimed sender
- public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976
  - known earlier in classified community

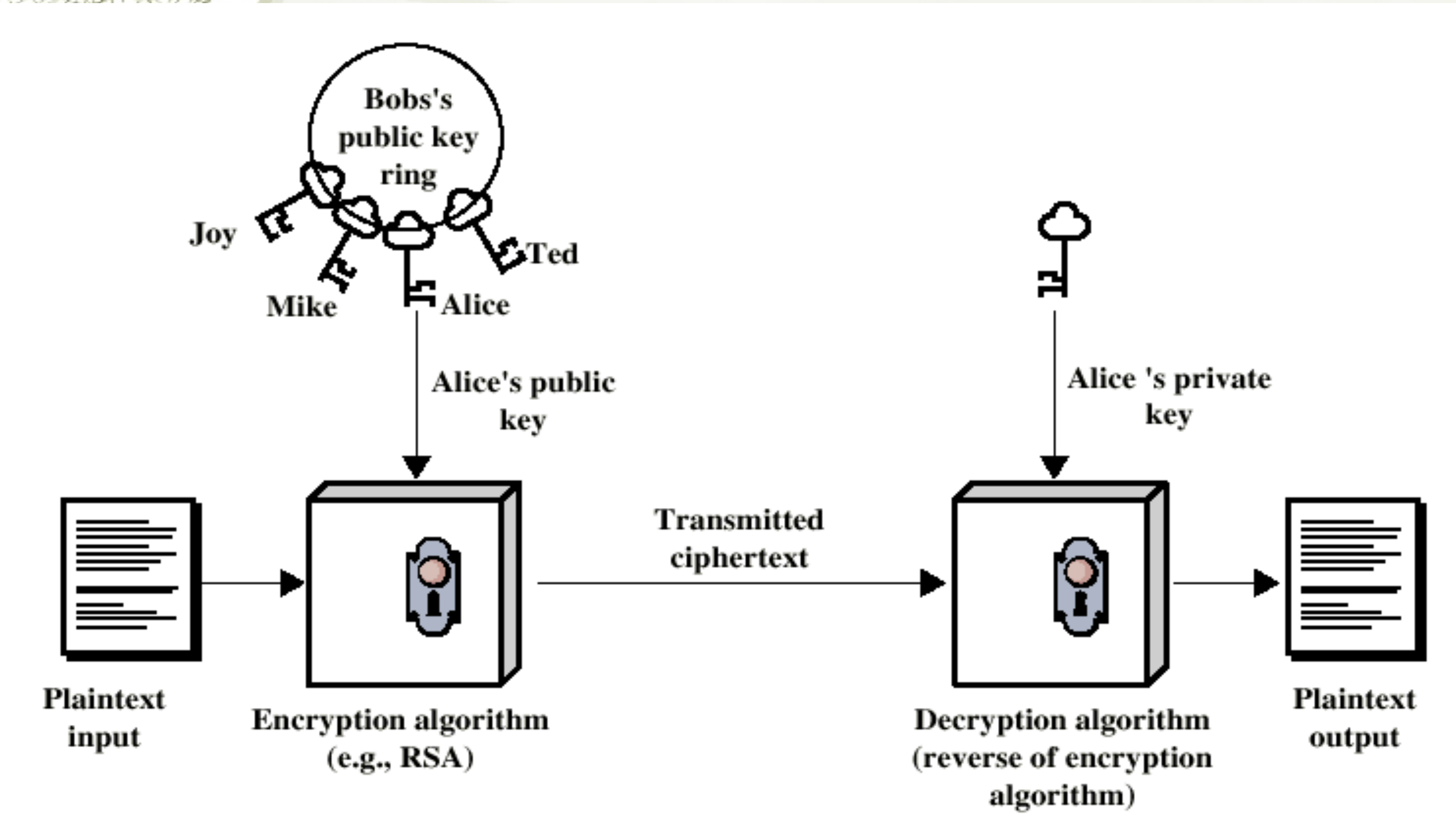
Public-Key Cryptography

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a related **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**
- **infeasible to determine private key from public**
- is **asymmetric** because
  - those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

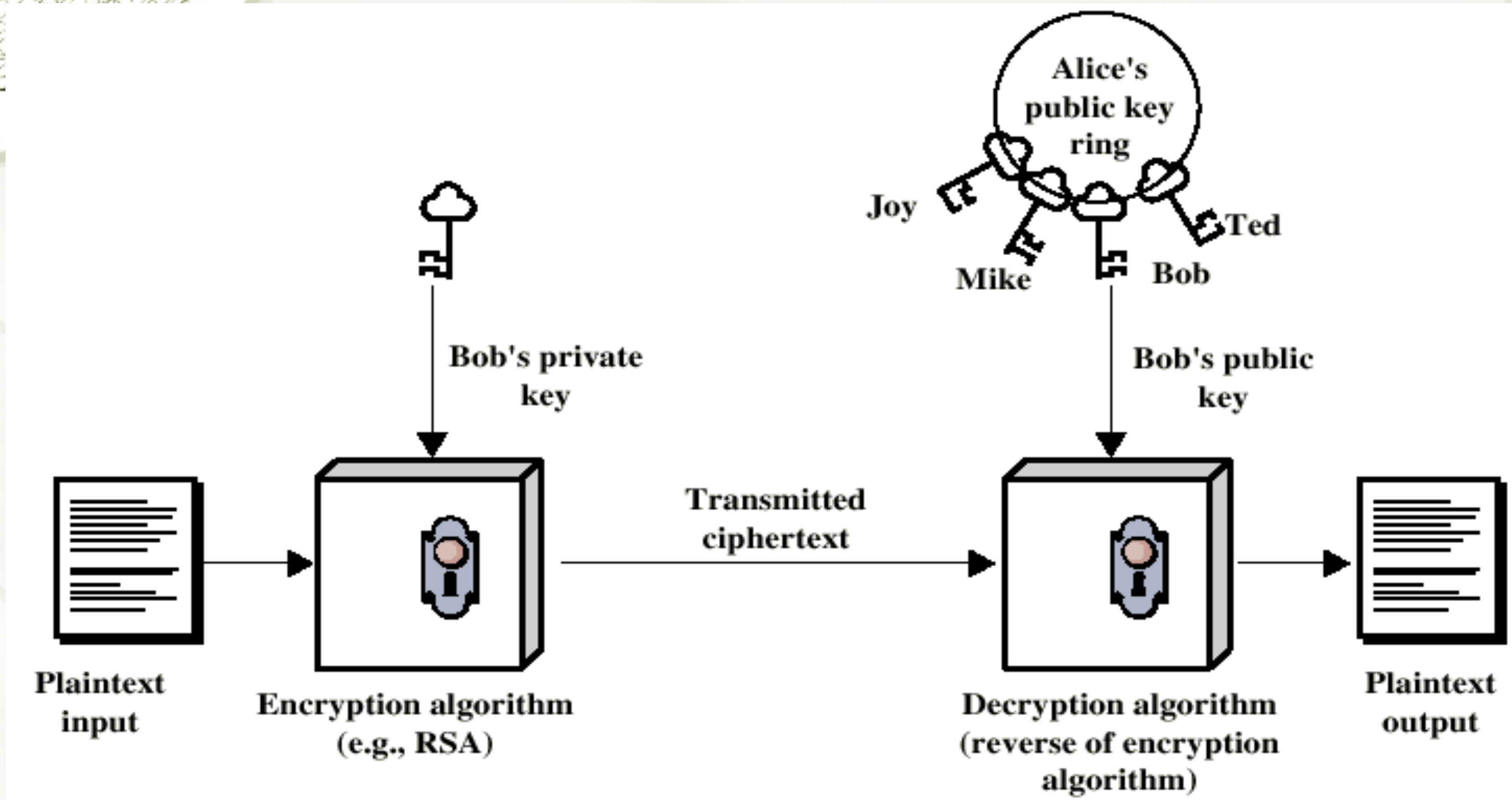
Public-Key Cryptography Principles

- The use of two keys has consequences in: key distribution, confidentiality and authentication.
- The scheme has six ingredients
  - Plaintext
  - Encryption algorithm
  - Public and private key
  - Cipher text
  - Decryption algorithm

Encryption using Public-Key System



Authentication using Public-Key System



Applications for Public-Key Cryptosystems

- Three categories:
  - **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
  - **Digital signature:** The sender ”signs” a message with its private key.
  - **Key exchange:** Two sides cooperate two exchange a session key.

Requirements for Public-Key Cryptography

1. Computationally easy for a party B to generate a pair (public key KU_b, private key KR_b)
2. Easy for sender to generate cipher text:
$$C = E_{KUb}(M)$$
3. Easy for the receiver to decrypt cipher text using private key:
$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

Requirements for Public-Key Cryptography

4. Computationally infeasible to determine private key (KR_b) knowing public key (KU_b)
5. Computationally infeasible to recover message M, knowing KU_b and cipher text C
6. Either of the two keys can be used for encryption, with the other used for decryption:

$$M = D_{KRb}[E_{KUb}(M)] = D_{KUb}[E_{KRb}(M)]$$

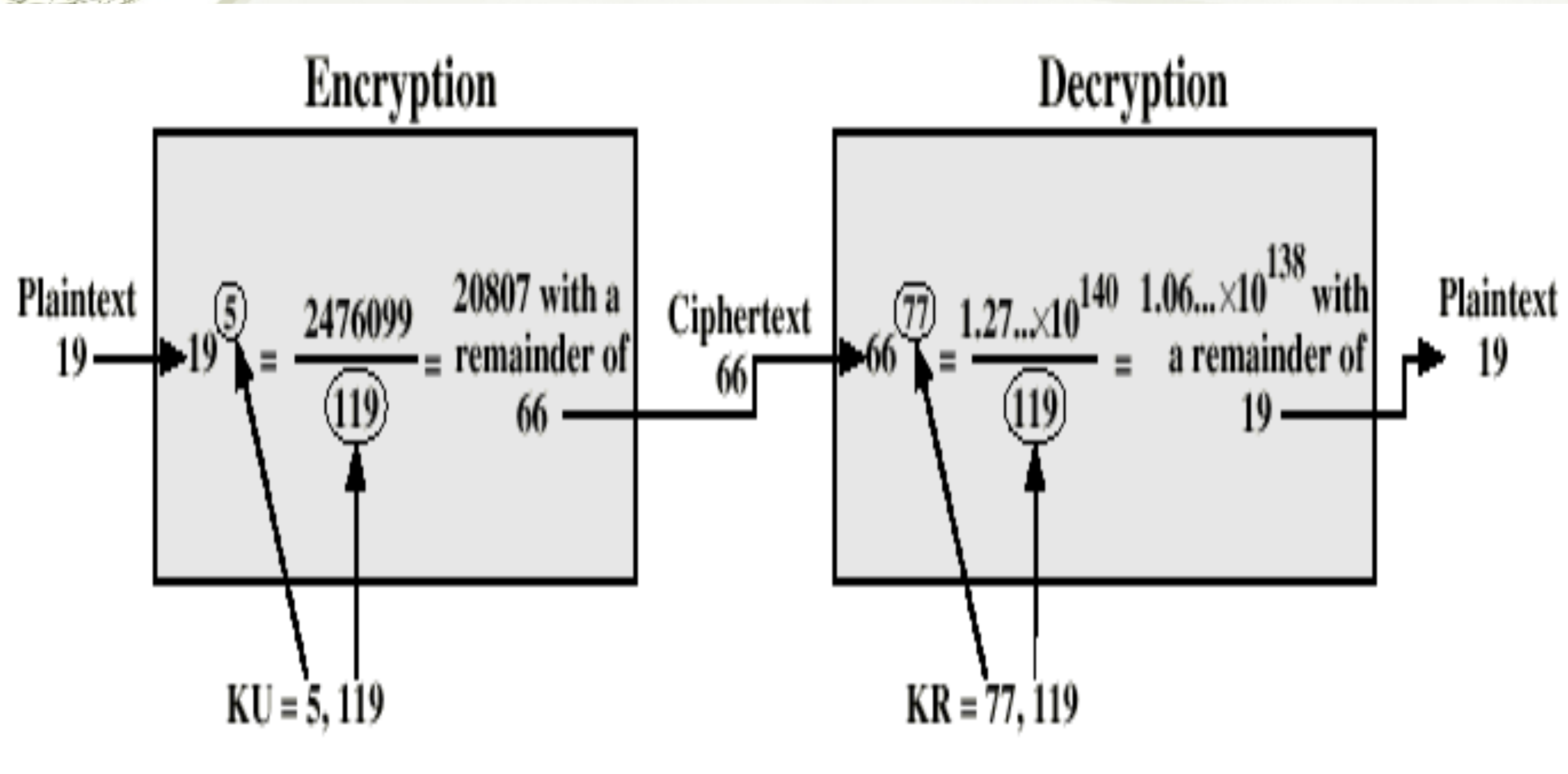
Public-Key Cryptographic Algorithms

- RSA and Diffie-Hellman
  - **RSA** - Ron Rivest, Adi Shamir and Len Adleman at MIT, in 1977.
    - RSA is a block cipher
    - The most widely implemented
  - **Diffie-Hellman**
    - Exchange a secret key securely
    - Compute discrete logarithms

The RSA Algorithm - Key Generation

1.	Select *p*,*q*	*p* and *q* both prime
2.	Calculate	$n = p \times q$
3.	Calculate	$\Phi(n) = (p-1)(q-1)$
4.	Select integer *e*	$\gcd(\Phi(n), e) = 1; 1 < e < \Phi(n)$
5.	Calculate *d*	$d = e^{-1} \bmod \Phi(n)$
6.	Public Key	KU = {e,n}
7.	Private key	KR = {d,n}

Example of RSA Algorithm



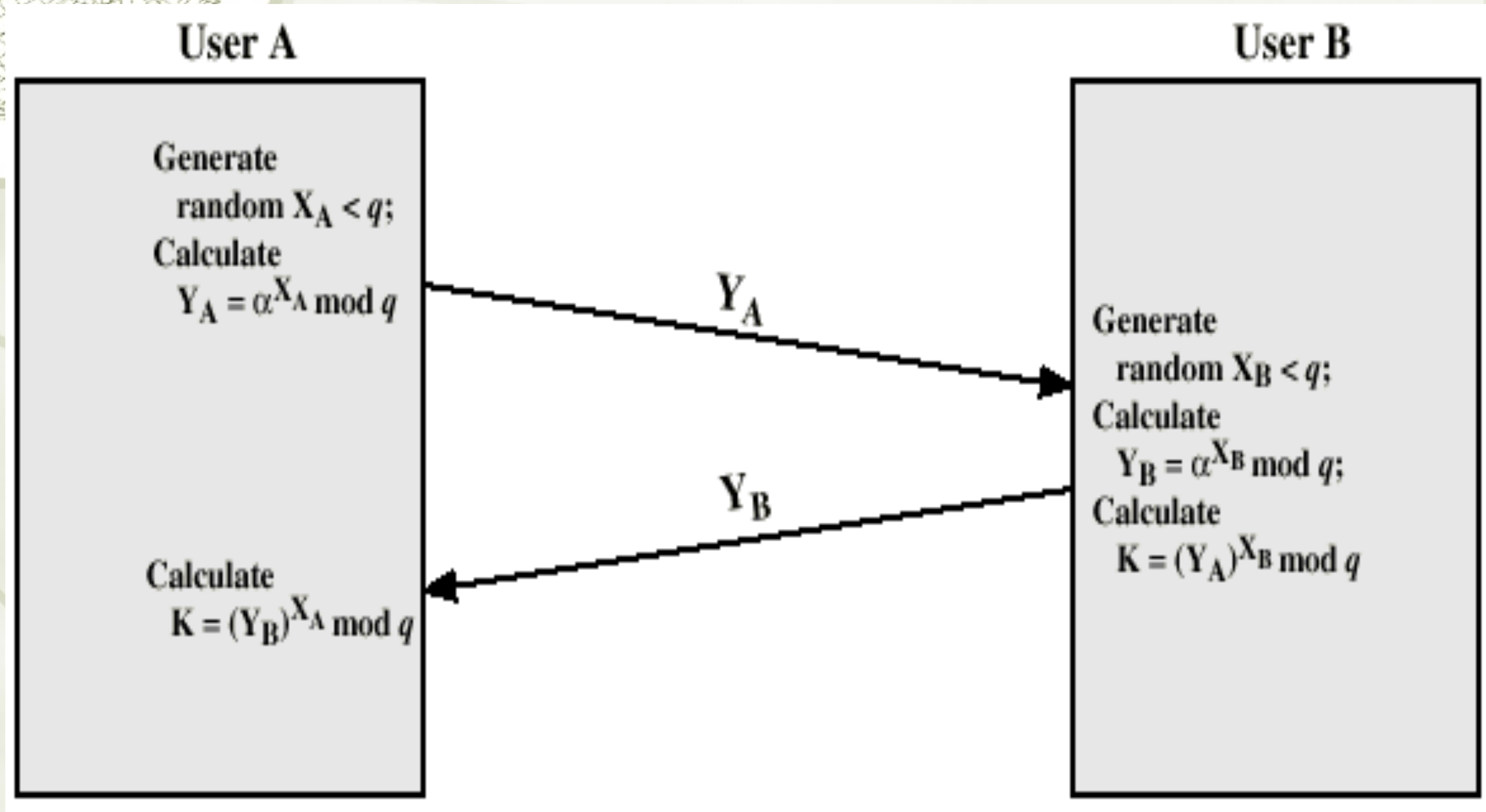
The RSA Algorithm - Encryption

- Plain text: $M<n$
- Cipher text: $C = M^e \pmod{n}$

The RSA Algorithm - Decryption

- Cipher text: C
- Plain text: $M = C^d \pmod{n}$

Diffie-Hellman Key Exchange



Other Public-Key Cryptographic Algorithms

- Digital Signature Standard (DSS)
  - Makes use of the SHA-1
  - Not for encryption or key exchange
- Elliptic-Curve Cryptography (ECC)
  - Good for smaller bit size
  - Low confidence level, compared with RSA
  - Very complex
- ElGamal
  - an asymmetric key encryption algorithm based on the Diffie–Hellman key exchange

Authentication

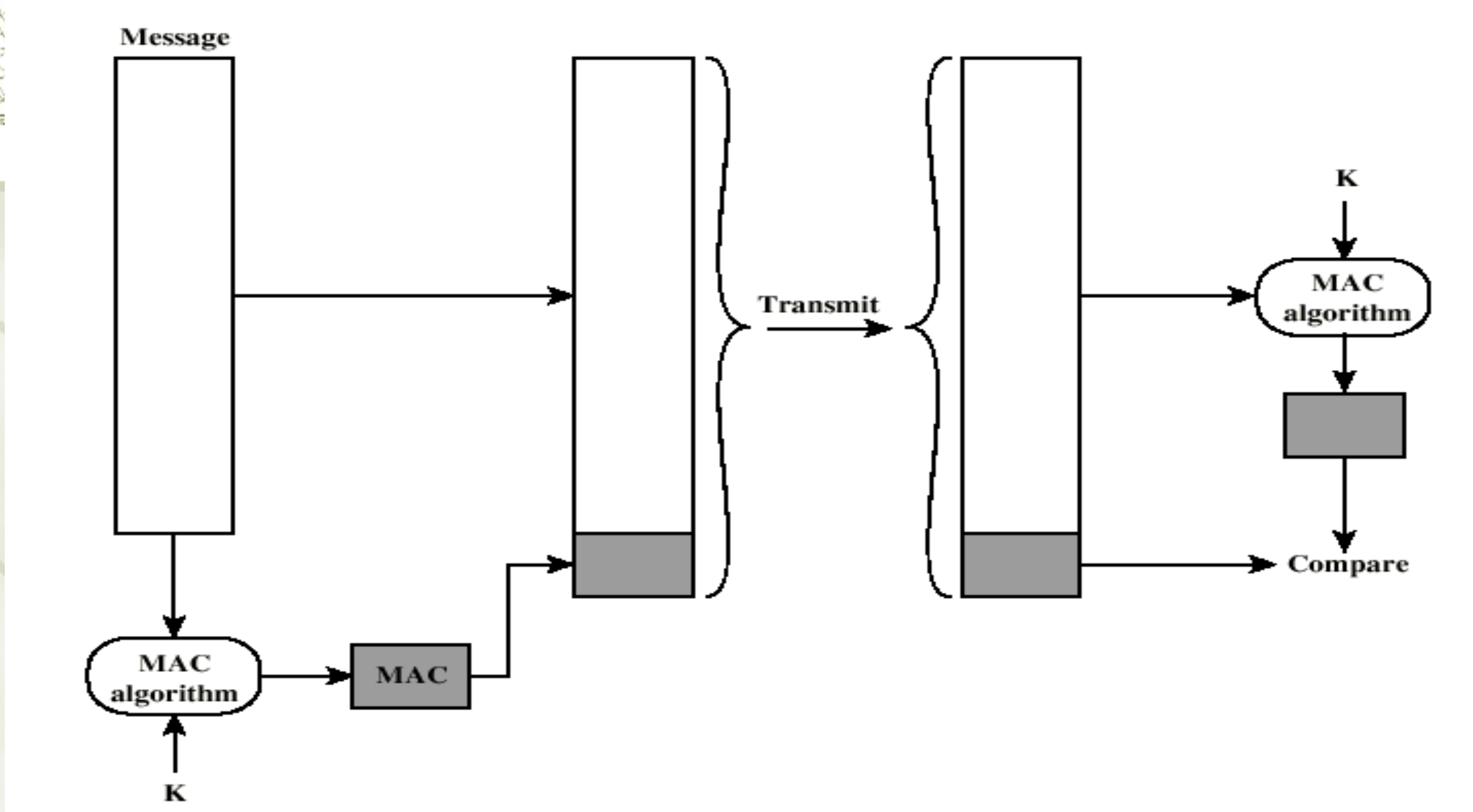
- Requirements - must be able to verify that:
  1. Message came from apparent source or author,
  2. Contents have not been altered,
  3. Sometimes, it was sent at a certain time or sequence.

- Protection against active attack (falsification of data and transactions)

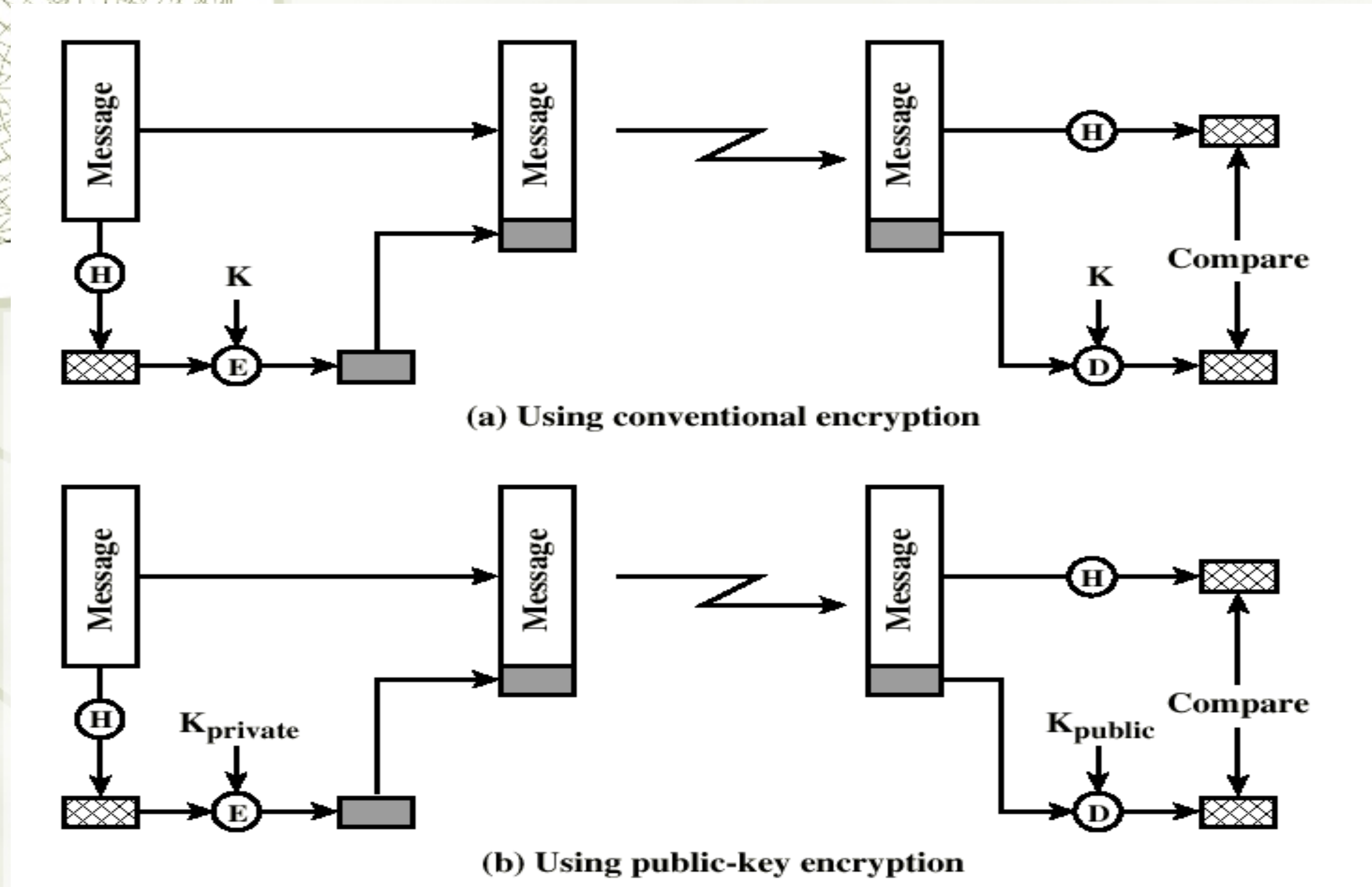
Approaches to Message Authentication

- Authentication Using Conventional Encryption
  - Only the sender and receiver should share a key
- Message Authentication without Message Encryption
  - An authentication tag is generated and appended to each message
- Message Authentication Code
  - Calculate the MAC as a function of the message and the key. MAC = F(K, M)

Authentication using a Message Authentication Code

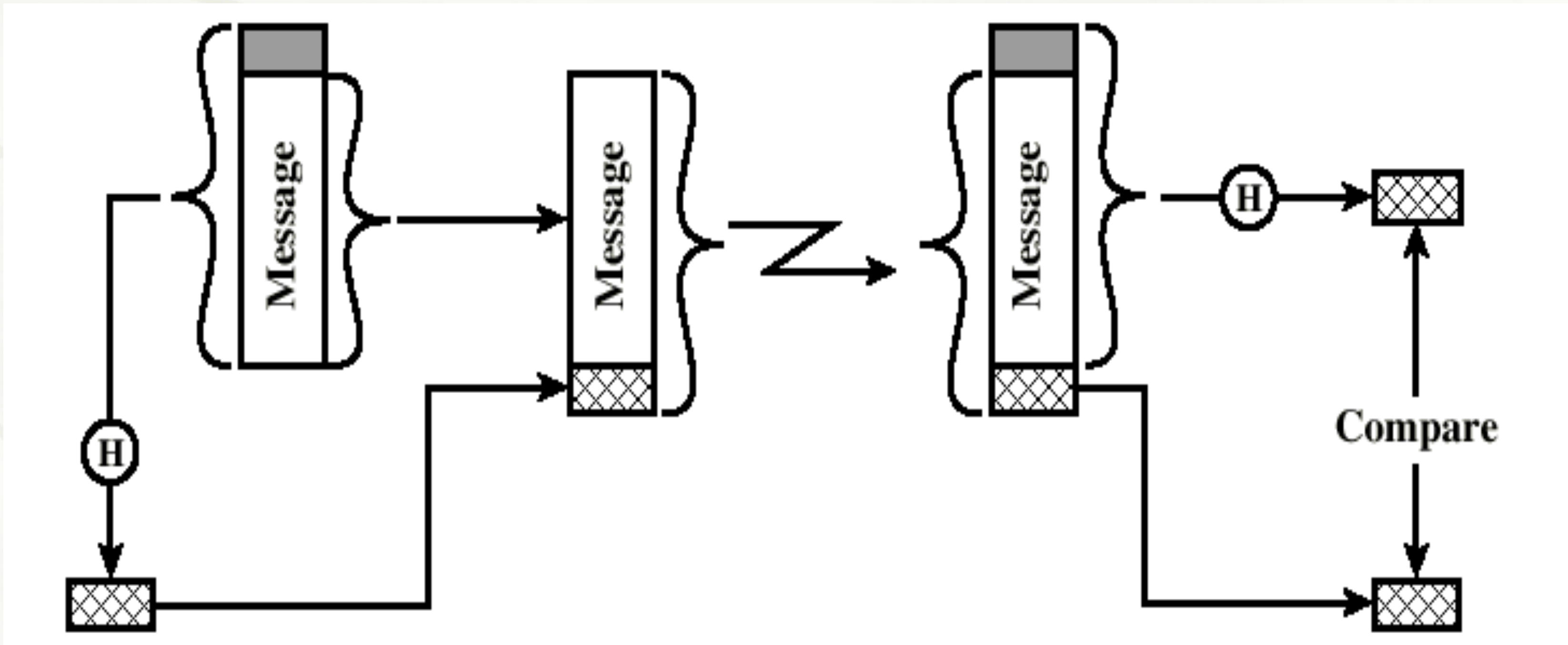


One-way HASH function



One-way HASH function

- Secret value is added before the hash and removed before transmission.



Key Management
Public-Key Certificate Use