Chap. 3 Public-Key Cryptography and Message Authentication

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

Public-Key Crypto

- Problems of symmetric key algorithms
  - Key distribution is difficult
  - Large number of keys are required
- Public-Key algorithms use a public-key and private-key pair over a message
  - Only the public-key can decrypt a message encrypted with the private key
  - Similarly, only the private key can decrypt a message encrypted with the public key

Public-key Crypto

- First proposed by W. Diffie and M. Hellman, and independently by R. Merkle in the late 1970s
- A crucial feature is that the private key is difficult to determine from the public key
- RSA is the most widely used public-key algorithm
- Generally, these algorithms are much slower than symmetric key algorithms

Applications for PK Crypto

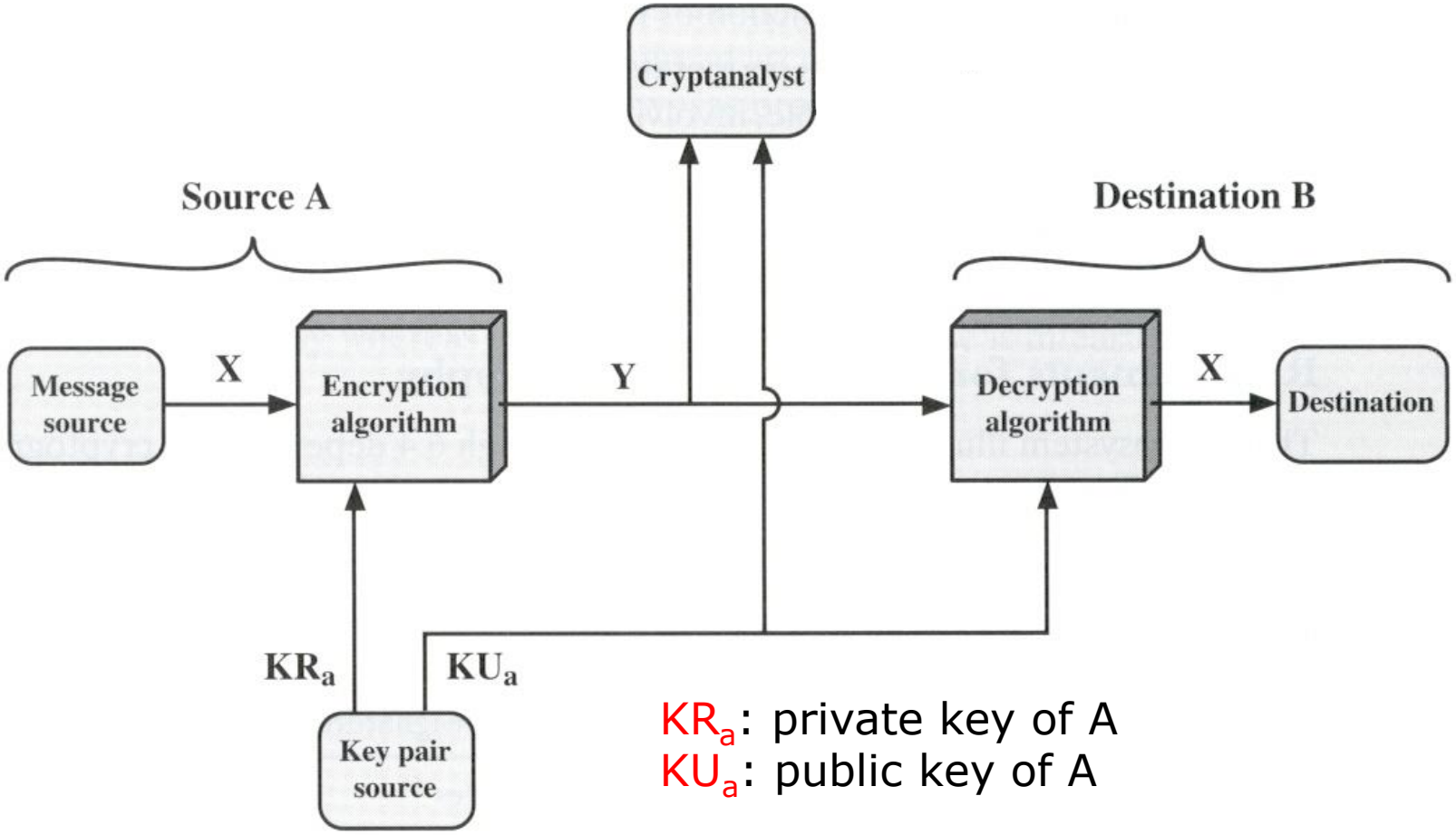
- Encryption/decryption: provides confidentiality
  - The sender encrypts a message with the receiver's public key
  - The receiver decrypts the ciphertext with his/her private key
- Digital signature: provides authentication
  - The sender signs a message with its private key
  - The receiver decrypts the ciphertext with the sender's public key

Applications for PK Crypto

- Key exchange
  - The sender generates a session key, and encrypts the session key with the receiver's public key and sends to the receiver
- Hybrid of symmetric and public-key approaches
  - Public key cryptosystem is used to distribute a session key for symmetric cryptosystem among peers
  - Symmetric cryptosystem is used to encrypt/decrypt messages

Applications for PK Crypto

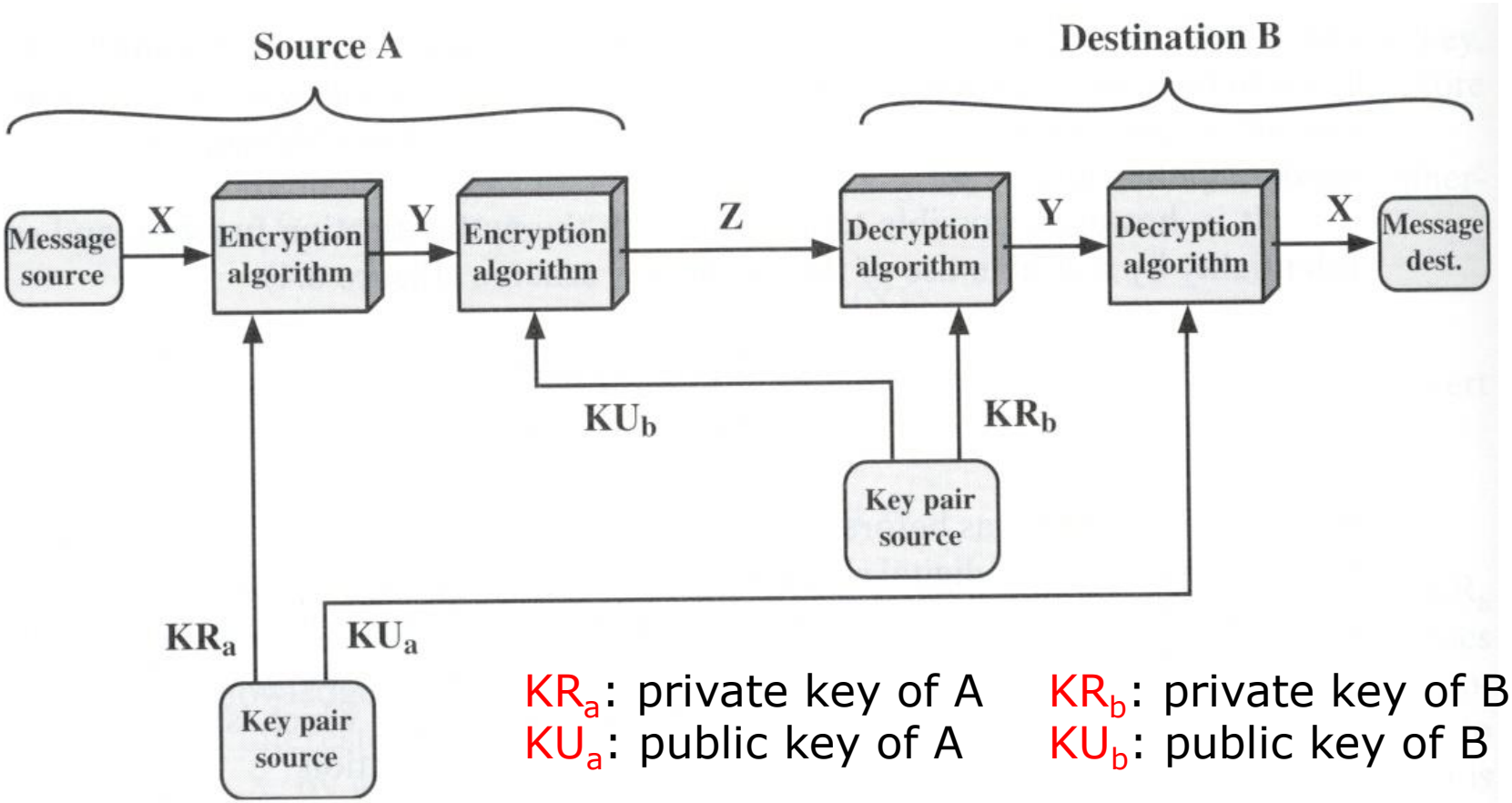
- Authentication using public-key cryptosystem



KR$_a$: private key of A
KU$_a$: public key of A

Applications for PK Crypto

- Confidentiality and authentication using public-key cryptosystem



KR_a: private key of A
KU_a: public key of A
KR_b: private key of B
KU_b: public key of B

Requirements for PK Crypto: Diffie & Hellman

- For a secure public-key crypto system b/w Alice and Bob:

1. It's *computationally easy* for Bob to generate a key pair
2. It's *computationally easy* for Alice to encrypt a message M generating ciphertext C
3. It's *computationally easy* for Bob to decrypt C
4. It's *computionally infeasible* for an Attacker knowing Bob's public key to recover the private key
5. It's *computionally infeasible* for an Attacker knowing Bob's public key to recover the original message

Rivest, Shamir, and Adleman (RSA)

- Relies on the difficulty of factoring large numbers
- Choose two large prime numbers, p and q
- Compute $n = pq$ and $z = (p-1)(q-1)$
- Choose a number, e, less than n, which has no common factors (other than 1) with z
  - e and z are relatively prime; normally e=3, 17, 65537
- Find a number, d, such that $ed \equiv 1 \pmod{z}$
  - $d = e^{-1} \bmod z$; Euclid algorithm
- public key : (n, e), private key : (n, d)

RSA

- Encryption of a plaintext M_i

$$C_i = M_i^e \pmod{n}$$

- Decryption of a ciphertext C_i

$$M_i = C_i^d \pmod{n}$$

Theorem: $C_i^d \pmod n = M_i$

- Proof:

$C_i^d \pmod n$

$= (M_i^e)^d \pmod n$; $C_i = M_i^e \pmod n$

$= M_i^{ed} \pmod n$; $(a^b)^c = a^{bc}$

$= M_i^{kz+1} \pmod n$; $ed \equiv 1 \pmod z$ and $ed = kz+1$

$= M_i M_i^{kz} \pmod n$; $a^{b+1} = aa^b$

$= M_i M_i^{k(p-1)(q-1)} \pmod n$; $z = (p-1)(q-1)$

$= M_i$

Euler theorem: when p and q are prime,

$M_i^{k(p-1)(q-1)} \pmod n = 1$

Example

- Choose two primes *p=47 and q=71*
- Then, *n = pq = 3337, z = 46 x 70 = 3220*
- Choose *e,* relatively prime to z : (e.g.) e = 79
- We need a *d* such that *ed ≡ 1 (mod z),* in other words,

  $d = e^{-1} \bmod z$: (e.g.) $d = 79^{-1} \pmod{3220} = 1019$
- Public key: (3337,79), Private key: (3337,1019)
- Encrypt: 688 -> 688^{79} mod 3337 = 1570
- Decrypt: 1570 -> 1570^{1019} mod 3337 = 688

Some Aspects of RSA

- Computational complexity of RSA comes from exponentiation
  - Exponentiation needs handling very large numbers
  - Uses the following property of modular arithmetic:

$$(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)]$$

  - Efficiency of exponentiation

    (e.g.) x^{16}*:* needs 4 multiplication only if we successively form x^2, x^4, x^8, and x^{16}

Hybrids

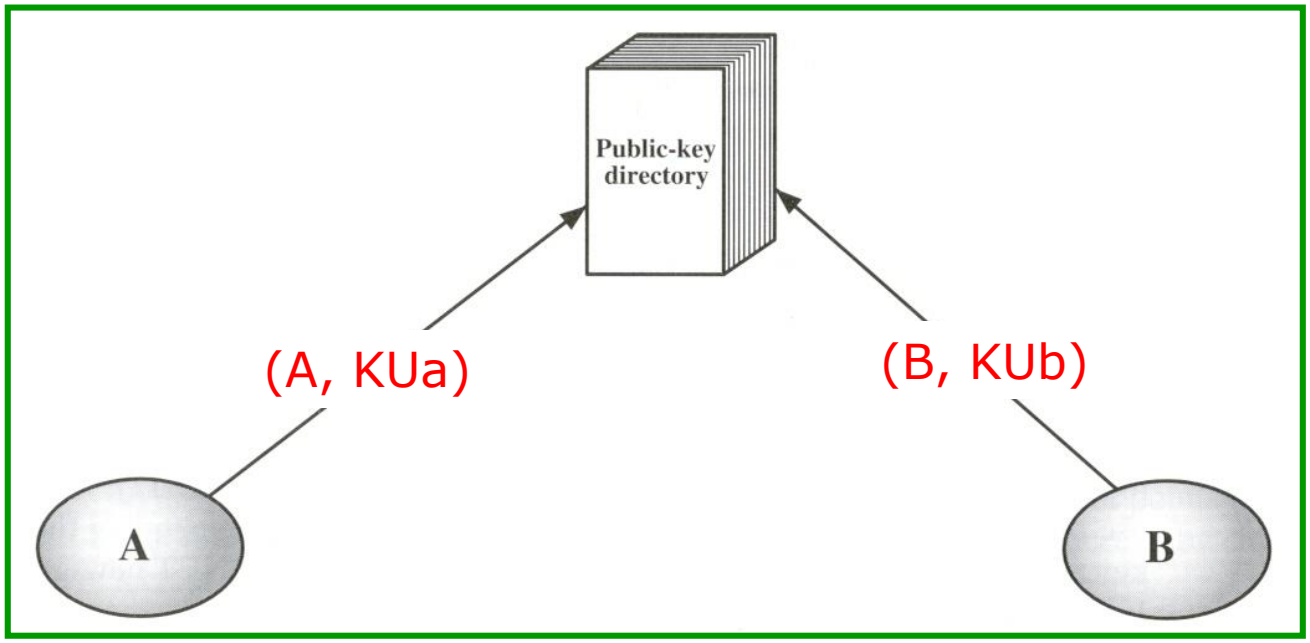
- Public keys are nice for communication, but slow
- Symmetric key algorithms are fast, but require many keys and difficult to share a session key
- A symmetric session key is generated by the sender, encrypted with the receiver's public key, and sent to the receiver
- Further communication occurs with the session key

Public Key Distribution

- Public Key distribution methods
  - Public announcement
  - Publicly available directory
  - Public-key authority
  - Public-key certificates
- Public announcement
  - Announce using public news group or mailing lists
  - used in PGP (Pretty Good Privacy)

Public Key Distribution

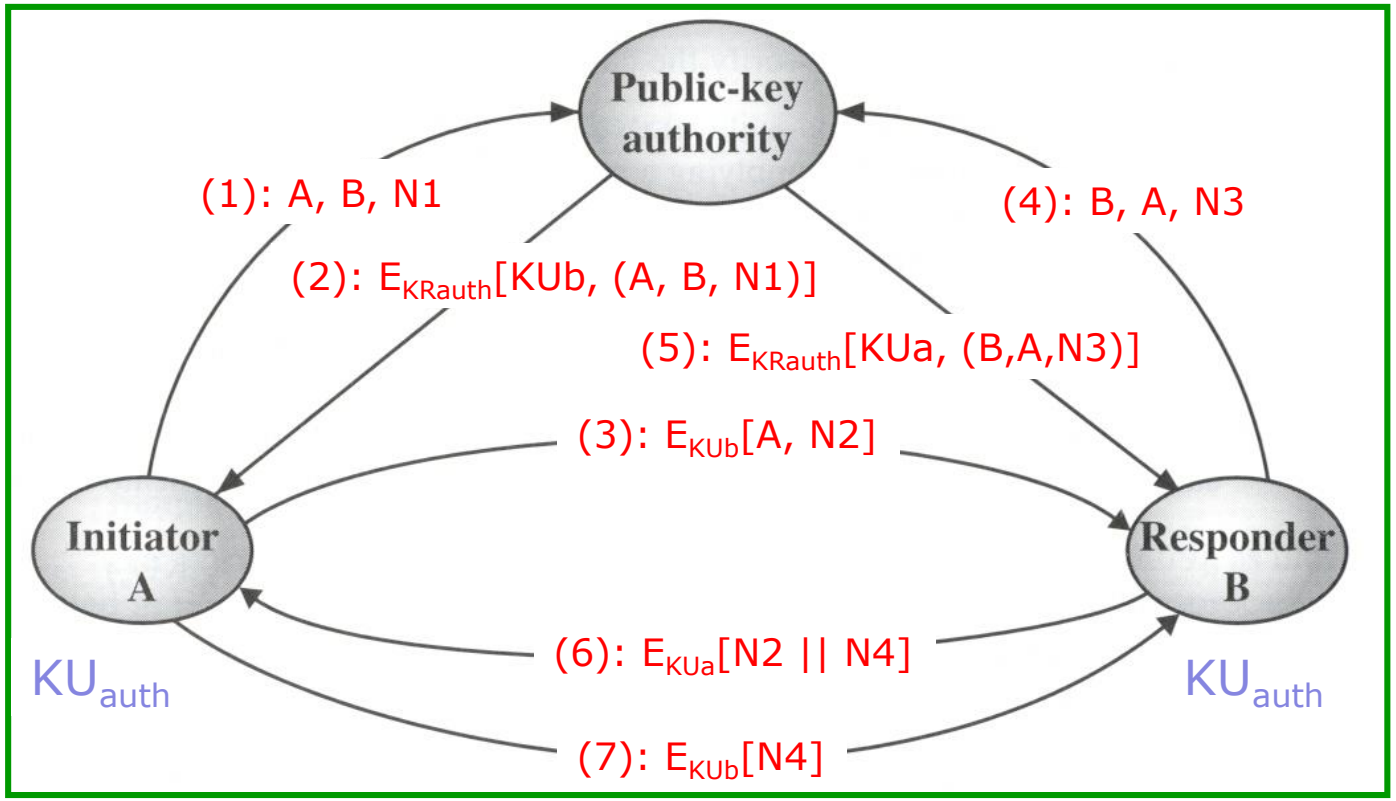
- Publicly available directory
  - Maintains (ID, KU_{ID}) pairs in public key directory
  - Each principal creates its public key and registers to the public directory
  - Each principal accesses to obtain other's public key



Public Key Distribution

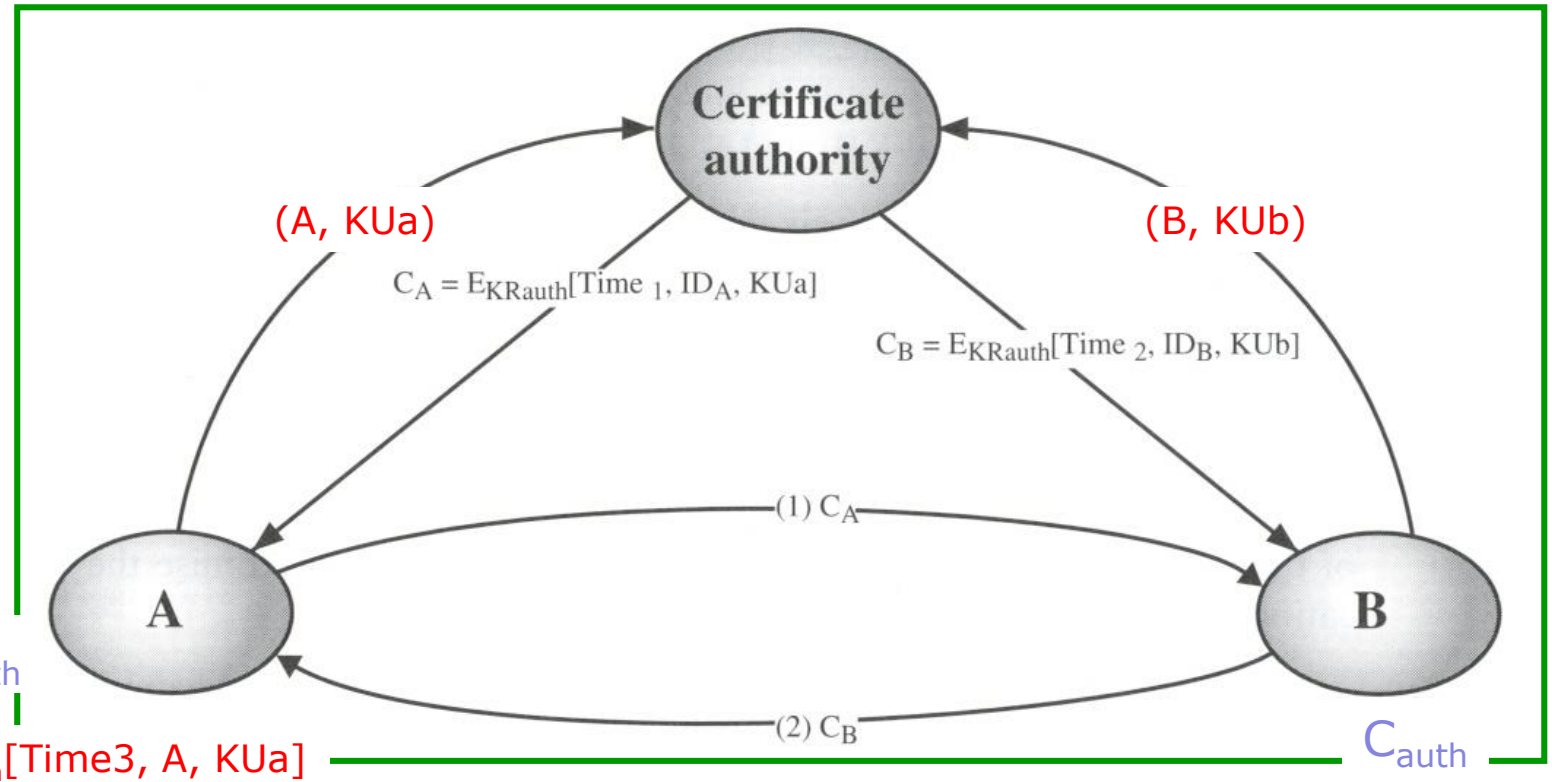
- Public-key authority
  - Controls the secure distribution of public keys in the public-key directory

Each peer knows public key of public-key authority



Key Distribution

- Public-key certificates
  - Certificate: (ID, KU_{ID}, $Sign_{CA}$) ; signed by CA
  - PKI (Public Key Infrastructure)



Diffie-Hellman Key Distribution

- Global public elements
  - Depends on the difficulty of computing discrete logarithms
  - q: a large prime number
  - α: α < q and a primitive root of q

-> { α mod q, α^2 mod q, ..., $\alpha^{(q-1)}$ mod q } generates 1 ~ (q-1)

When $Y_A = \alpha^{X_A} \bmod q$, its difficult to find X_A even if we know q, α, and Y_A

Diffie-Hellman Key Distribution

- Example: $q = 7$
  - $\alpha = 3$: $3^1 \bmod 7 = 3$, $3^2 \bmod 7 = 2$, $3^3 \bmod 7 = 6$, $3^4 \bmod 7 = 4$, $3^5 \bmod 7 = 5$, $3^6 \bmod 7 = 1$
  - $\alpha = 5$: $5^1 \bmod 7 = 5$, $5^2 \bmod 7 = 4$, $5^3 \bmod 7 = 6$, $5^4 \bmod 7 = 2$, $5^5 \bmod 7 = 3$, $5^6 \bmod 7 = 1$

Diffie-Hellman Key Distribution

- Global public: user A and B knows q and α
- User A
  - Select a private X_A such that $X_A < q$
  - Calculate public Y_A such that $Y_A = \alpha^{X_A} \bmod q$
  - Send Y_A to User B
- User B
  - Select a private X_B such that $X_B < q$
  - Calculate public Y_B such that $Y_B = \alpha^{X_B} \bmod q$
  - Send Y_B to User B

Diffie-Hellman Key Distribution

- Secret key generation by User A
  - $K = y_B^{X_A} \mod q$
- Secret key generation by User B
  - $K = y_A^{X_B} \mod q$
- Diffie-Hellman public-key algorithm
  - limited to the exchange of session keys
  - not resistant to the man-in-the middle attack

Other Public-Key Cryptogrpahy

- DSS (Digital Signature Standard)
  - Digital signature technique using hash algorithm, published by NIST
  - Uses SHA-1 hash algorithm
- ECC (Elliptic Curve Cryptography)
  - Public-key cryptosystem based on elliptic curve algorithm
  - ECC provides equal security for a smaller key size compared to RSA

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

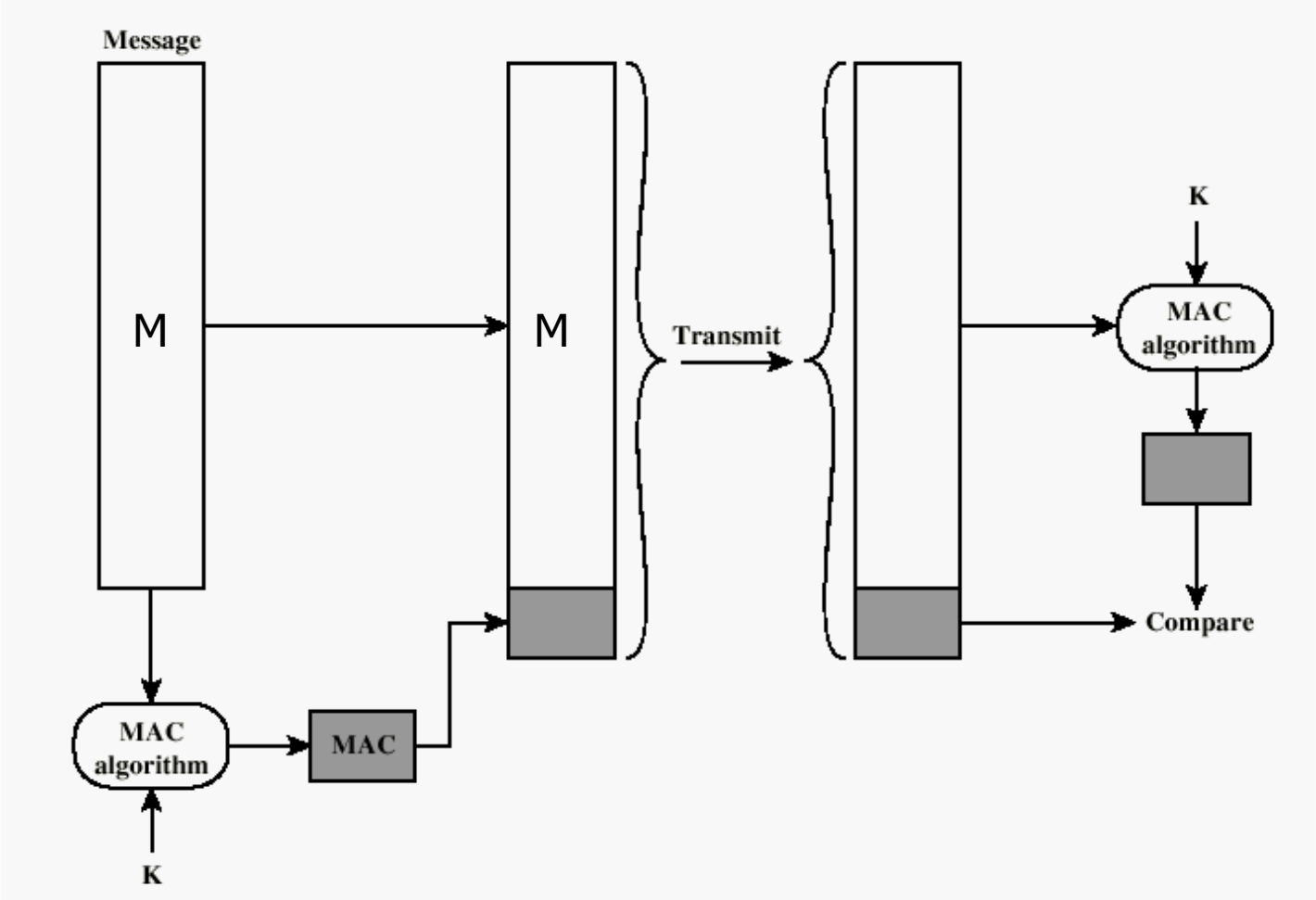
- Message Authentication using conventional encryption
  - Only the sender and receiver should share a key

Message Authentication

- Message Authentication without message encryption
  - An authentication tag is generated and appended to each message
- Message Authentication Code (MAC)
  - Calculate the MAC as a function of the message and the key: mac(M) = F(K, M)
  - Send [M, mac(M)]

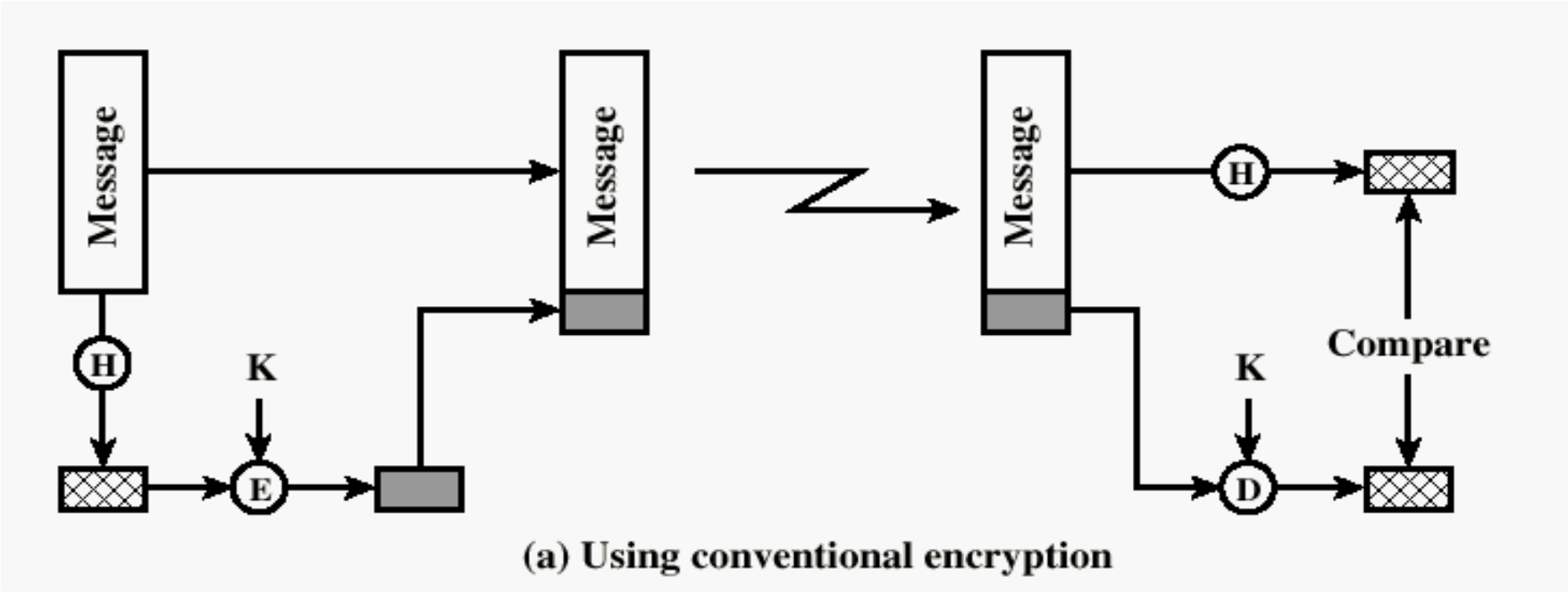
Message Authentication

- Message Authentication using MAC



Message Authentication

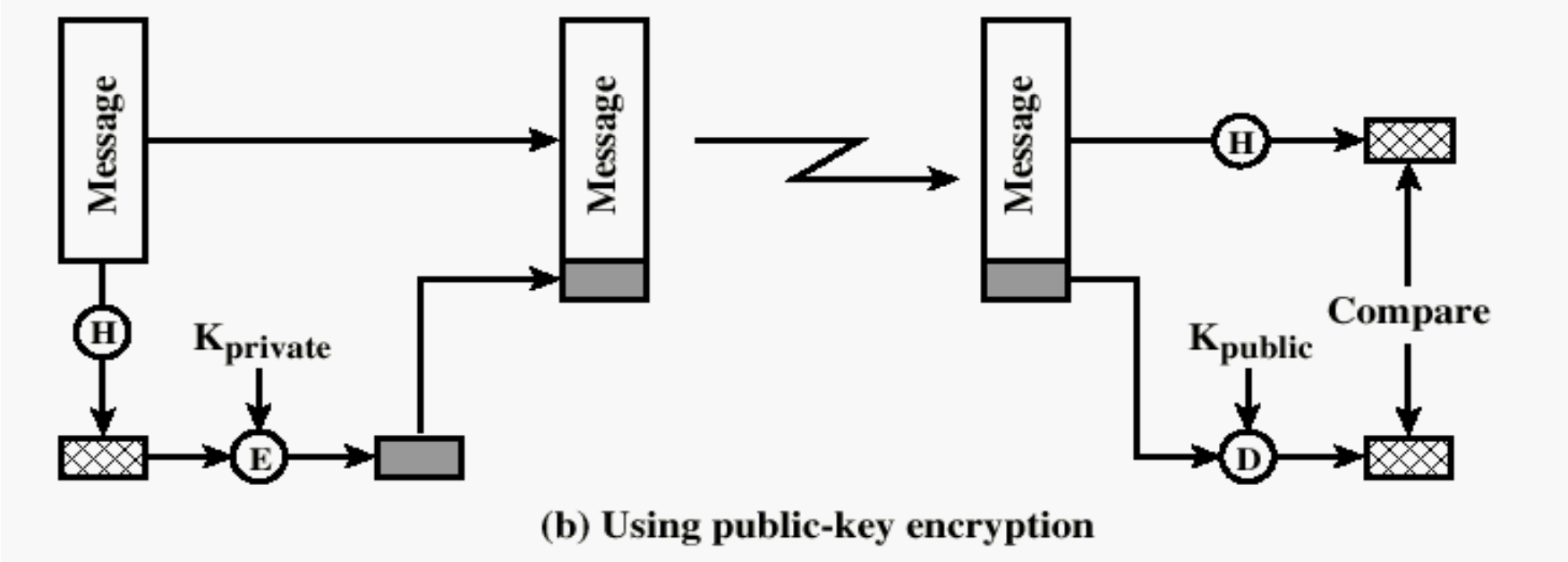
- Message Authentication using hashed MAC



(a) Using conventional encryption

Message Authentication

- Message Authentication using hashed MAC

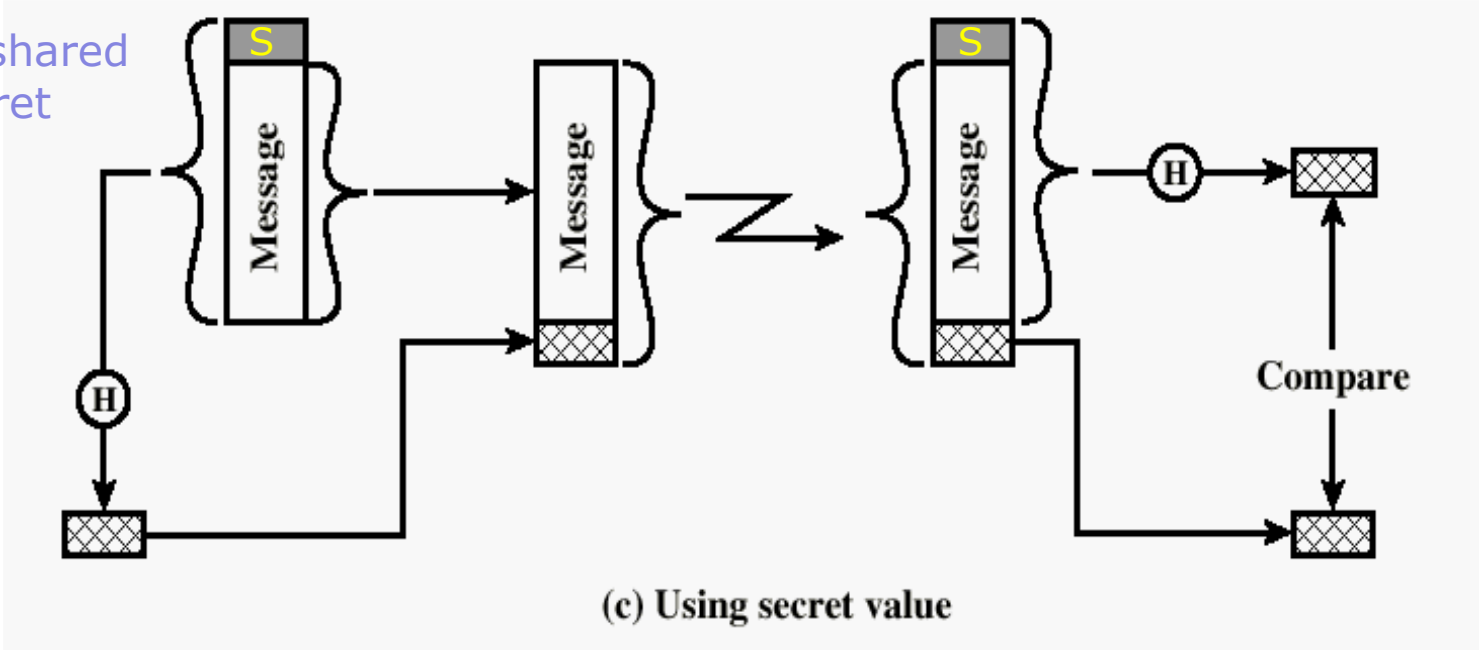


(b) Using public-key encryption

$K_{private}$: private key of sender
K_{public} : public key of sender

Message Authentication

- Message Authentication using hashed MAC and secret value
  - Secret value is added before the hash and removed before transmission.



HASH Functions

- HASH function is to produce a "fingerprint"
- Properties of a HASH function *H*
  - *H* can be applied to a block of data at any size
  - *H* produces a fixed length output
  - *H(x)* is easy to compute for any given x
  - For any given block (x, h=H(x)), it is computationally infeasible to find y such that H(y) = h : weak collision resistance
  - It is computationally infeasible to find any pair (x, y) such that H(x) = H(y) : strong collision resistance

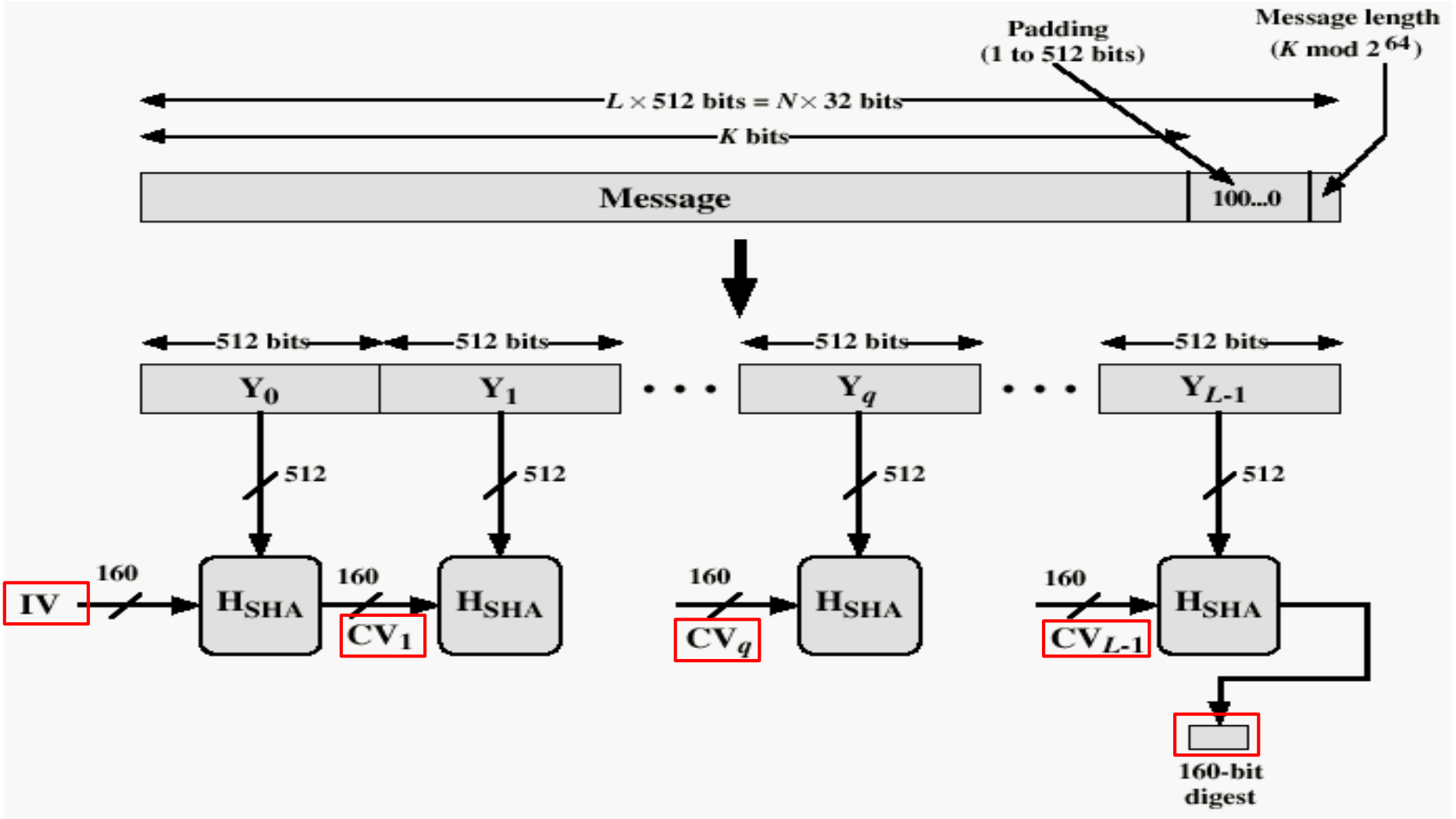
HASH Functions

- Simple hash function using XOR

	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

Does not provide collision resistance

HASH Functions : SHA-1



Other HASH Functions

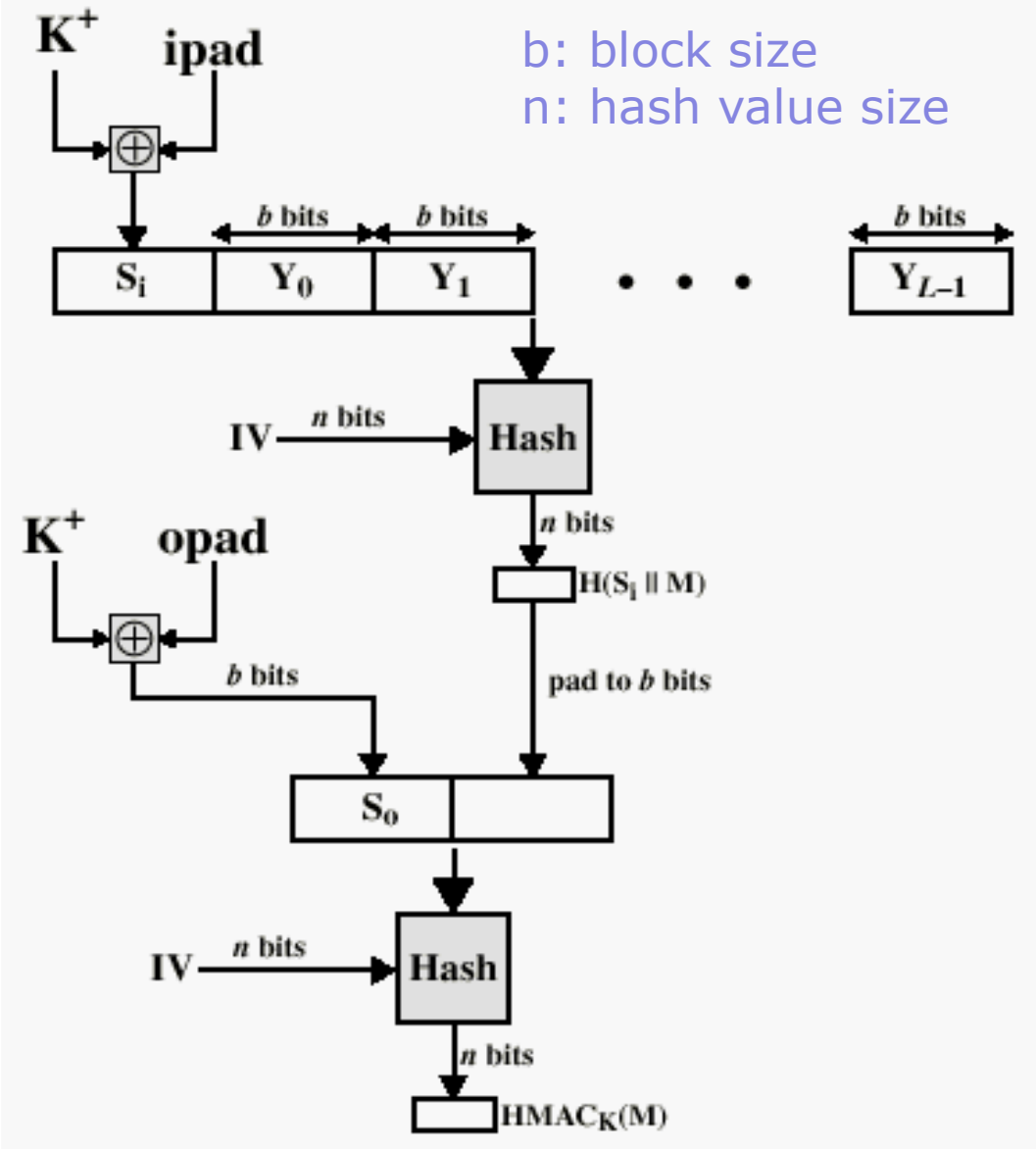
	SHA-1, SHA-256	MD5	RIPEMD-160
Digest length	160, 256 bits	128 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	80 (4 rounds of 20)	64 (4 rounds of 16)	160 (5 paired rounds of 16)
Maximum message size	2^{64}-1 bits	∞	∞

HMAC

- HMAC: hashed MAC with secret value derived from a cryptographic hash code, such as SHA-1

- Motivations:
  - Cryptographic hash functions executes faster than encryption algorithms such as DES and AES
  - Library code for cryptographic hash functions is widely available

HMAC Structure



K^+: K padded with 0s on the left (b-bit)
ipad: 00110110 repeated b/8 times
opad: 01011100 repeated b/8 times

Hash: hash function like SHA-1 or MD5