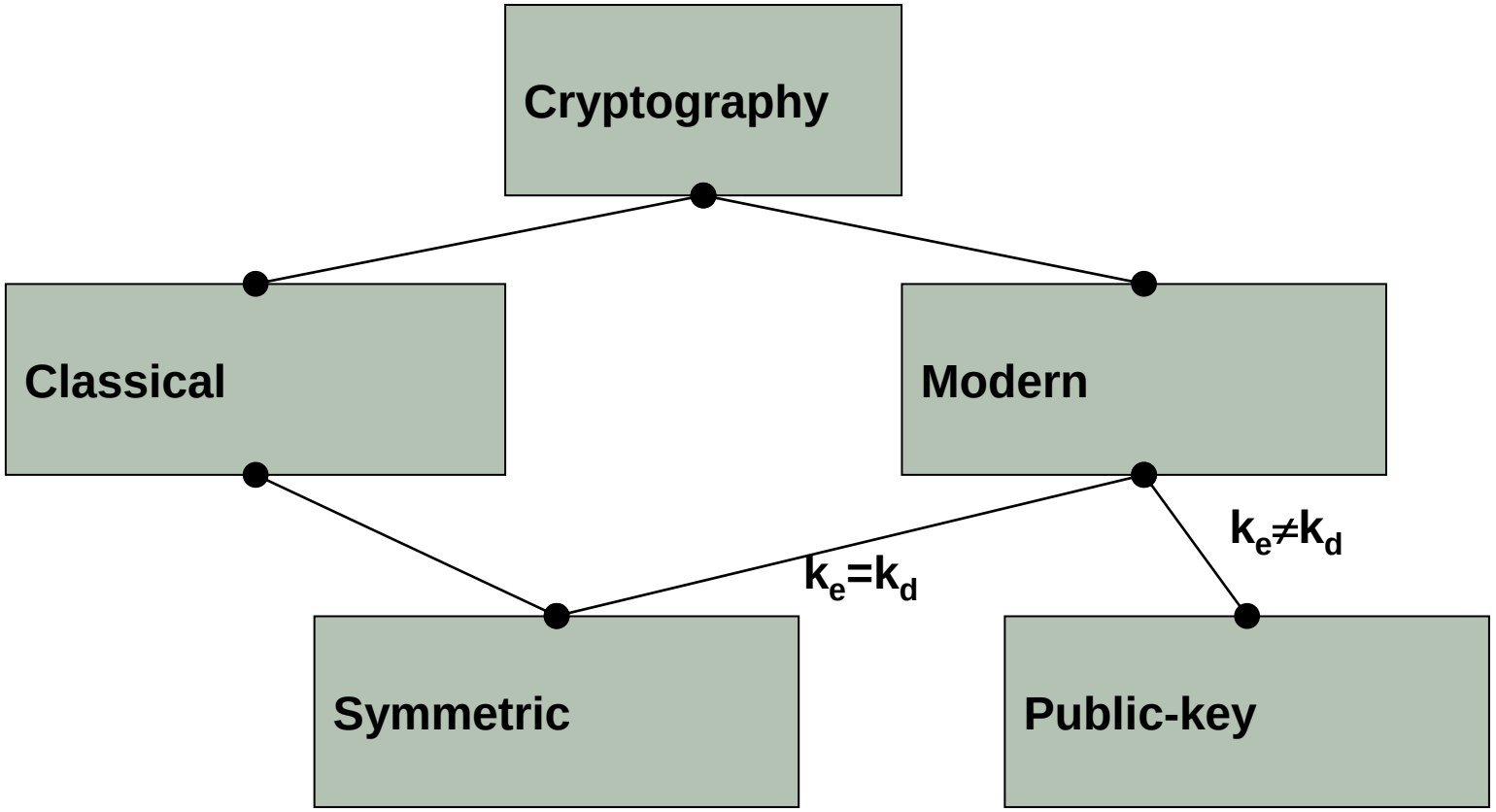
Cryptography Basics

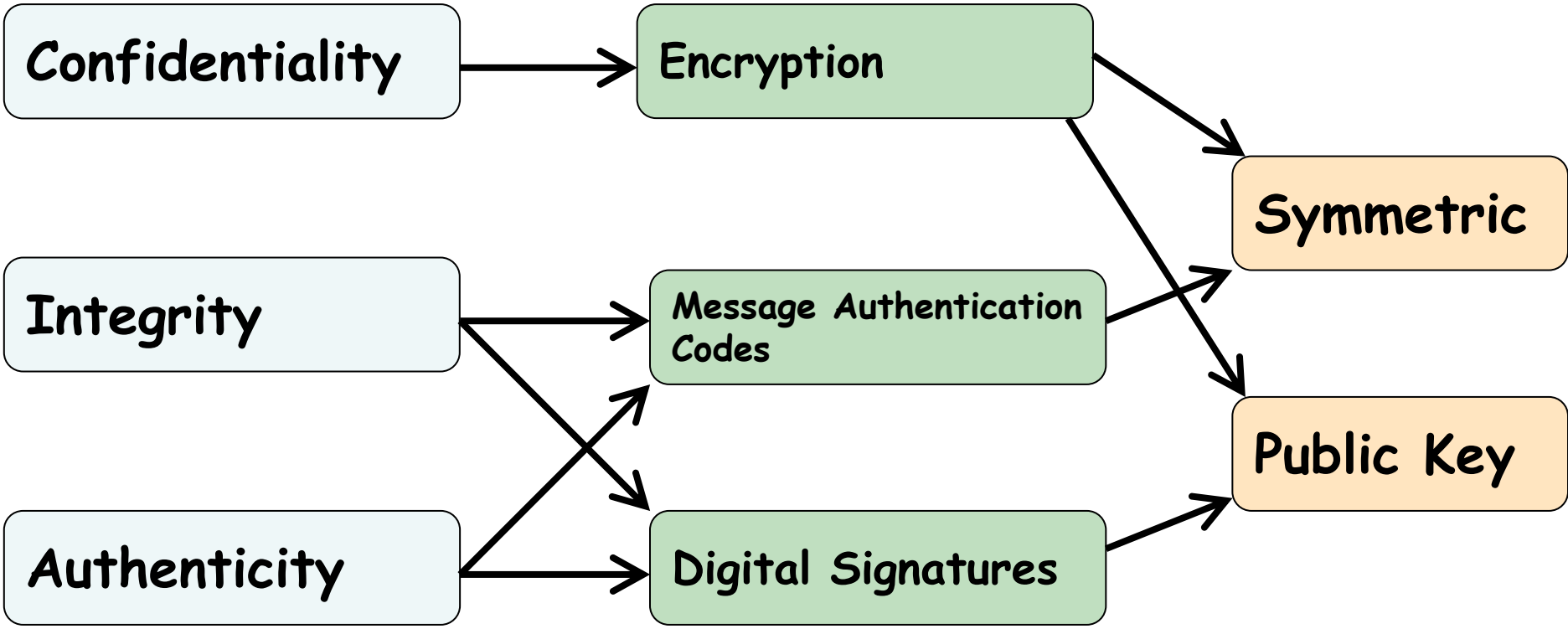
Outline

- Tools for confidentiality
  - Symmetric-key cryptosystems
  - Public-key cryptosystems
- Tools for integrity and authenticity
  - Symmetric-key cryptosystems
  - Public-key cryptosystems
- Hybrid systems

Types of Cryptography



Cryptography and Security Assurance



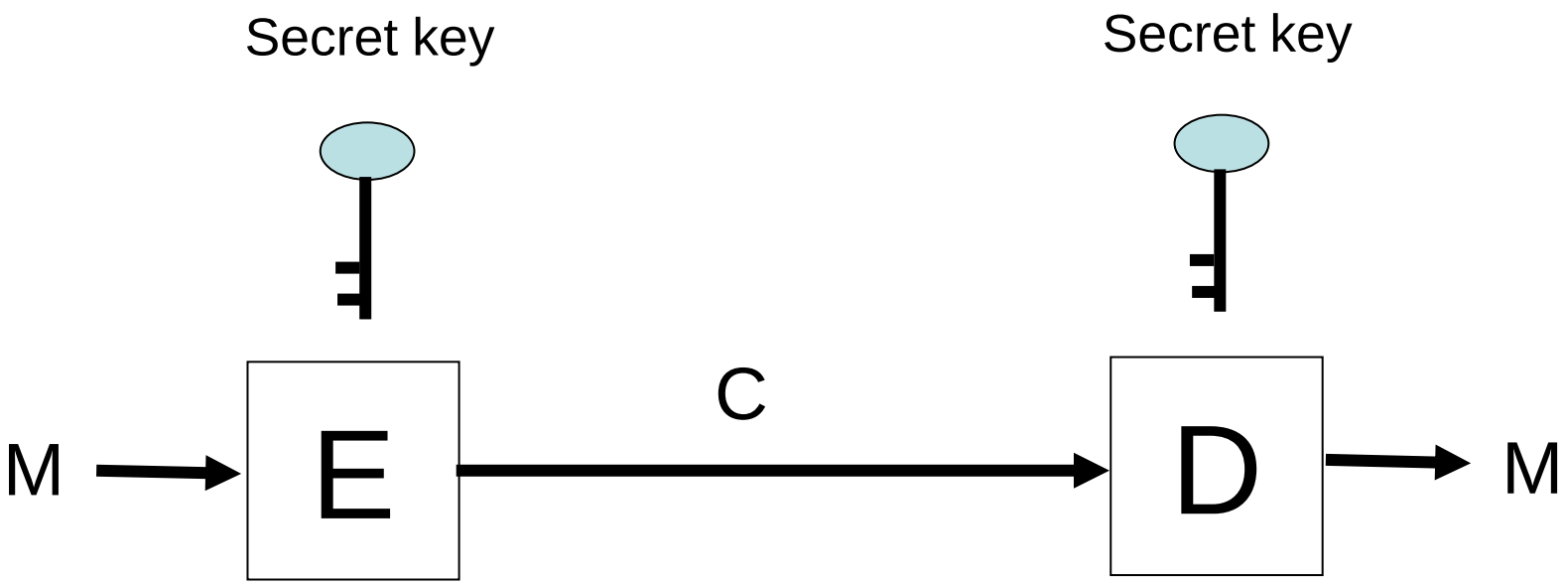
TOOLS FOR CONFIDENTIALITY

Models of Encryption and Decryption

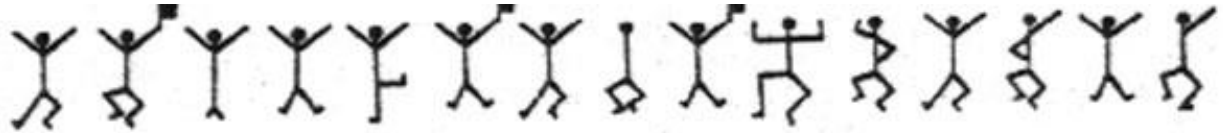
- Symmetric-key encryption: Encryption key and decryption key are the same.

- Asymmetric-key encryption: Encryption key and decryption key are different.

Symmetric-Key Encryption

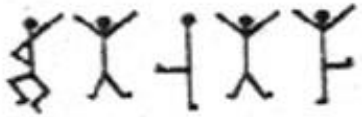


Little Dancing Men from Sherlock Holmes

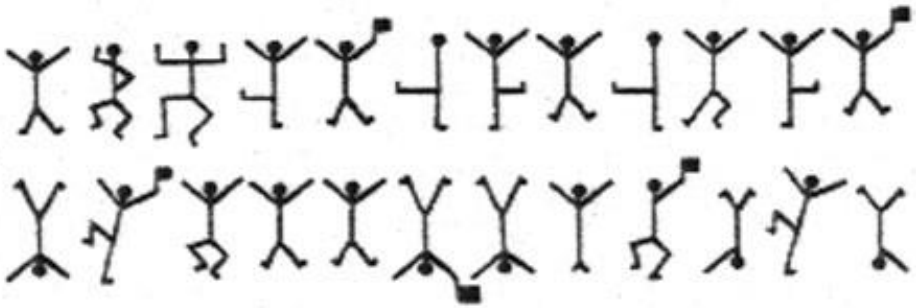


criminal's message (1)

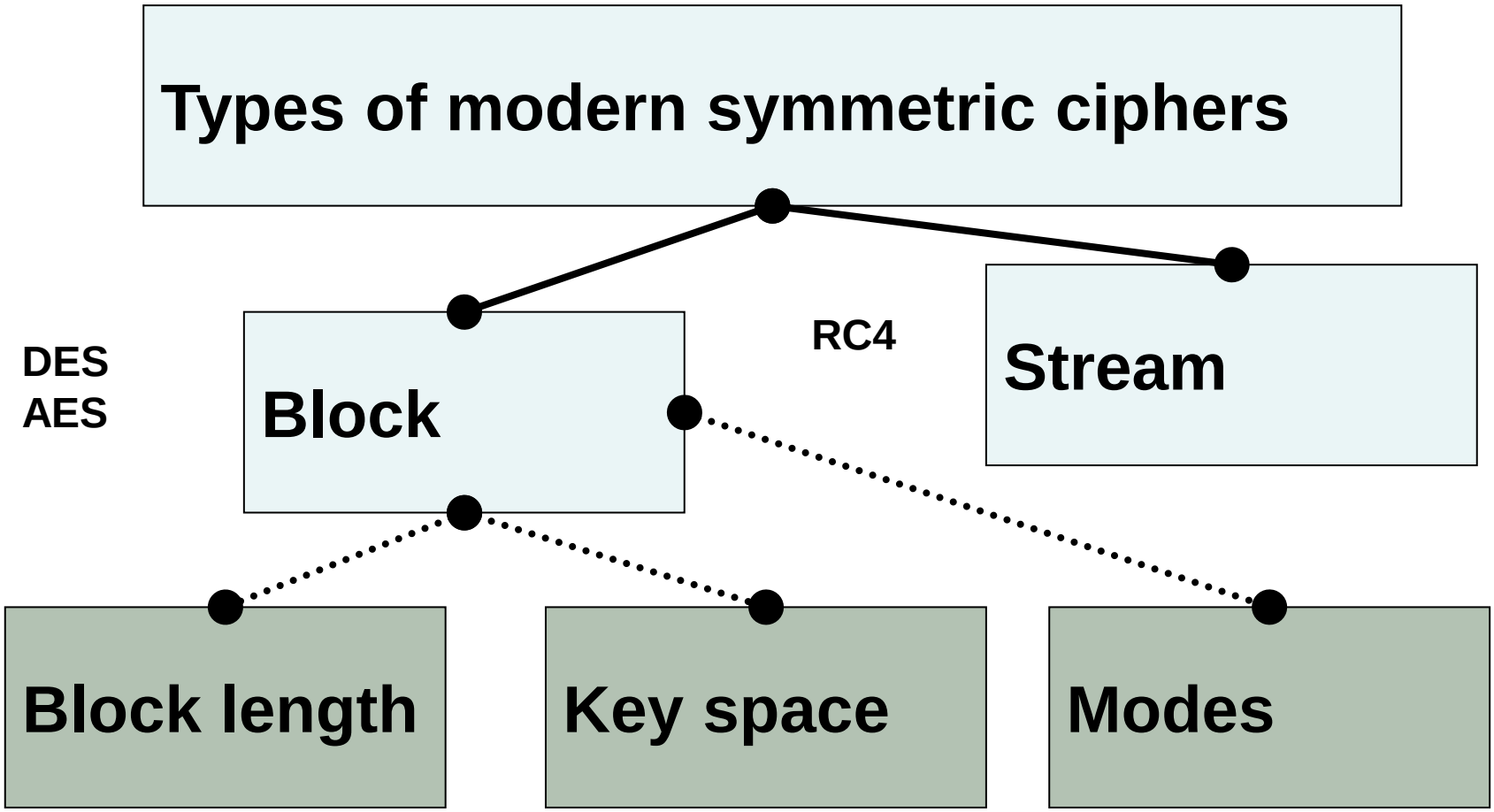
criminal's message (2)



Elsie's reply



criminal's message (3)



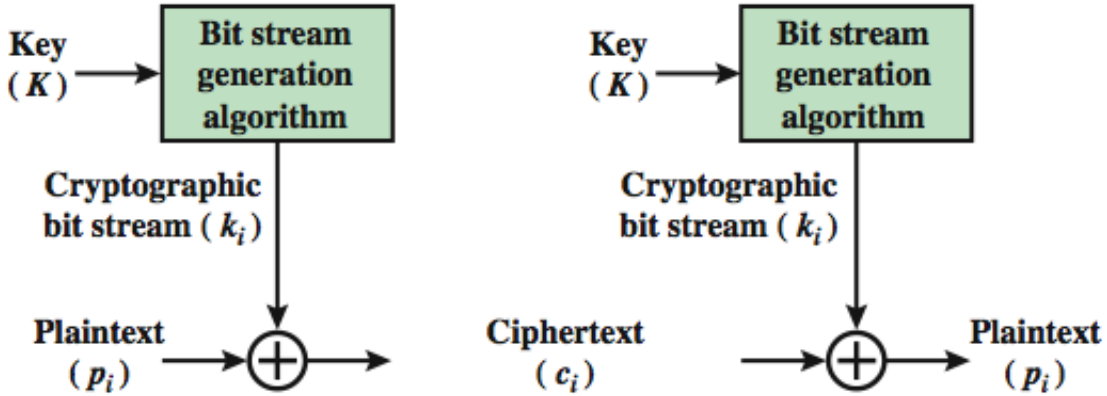
Stream ciphers

Operate on the plaintext a single bit (or sometimes byte) at a time

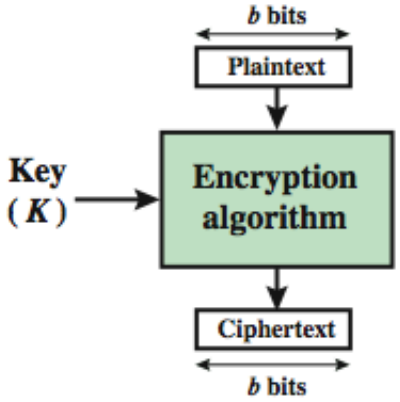
Block ciphers

Operate on the plaintext in groups of bits. The groups of bits are called blocks. Typical block size is 64 bits or multiple of it (e.g., 128 bits, 256 bits).

Block vs Stream Ciphers



(a) Stream Cipher Using Algorithmic Bit Stream Generator



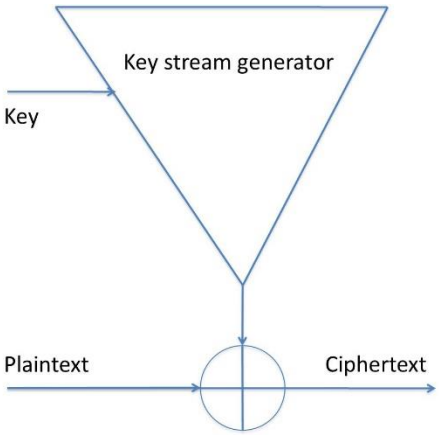
(b) Block Cipher

Stream Ciphers

- Stream Ciphers convert plaintext to ciphertext by a key stream.

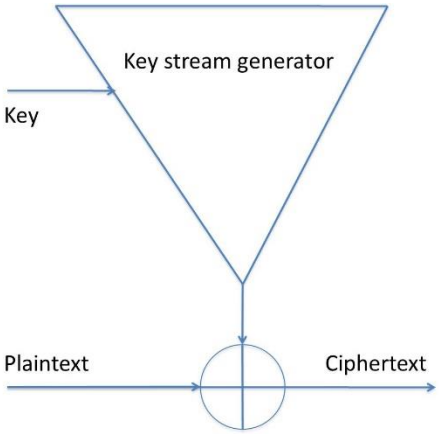
$$C = c_1\ c_2\ ... = E_{k1}(p_1)\ E_{k2}(p_2)\ ...$$

- The simplest stream cipher
  - Keystream generator: $\{k_i\},\ i=1,2,\ ...n$
  - A stream of plaintext bits: $\{p_i\},\ i=1,2,...,n$
  - Stream of ciphertext bits: $\{c_i\},\ i=1,2,...,n$
  - Encryption: $c_i = p_i \oplus k_i$
  - Decryption: $p_i = c_i \oplus k_i$



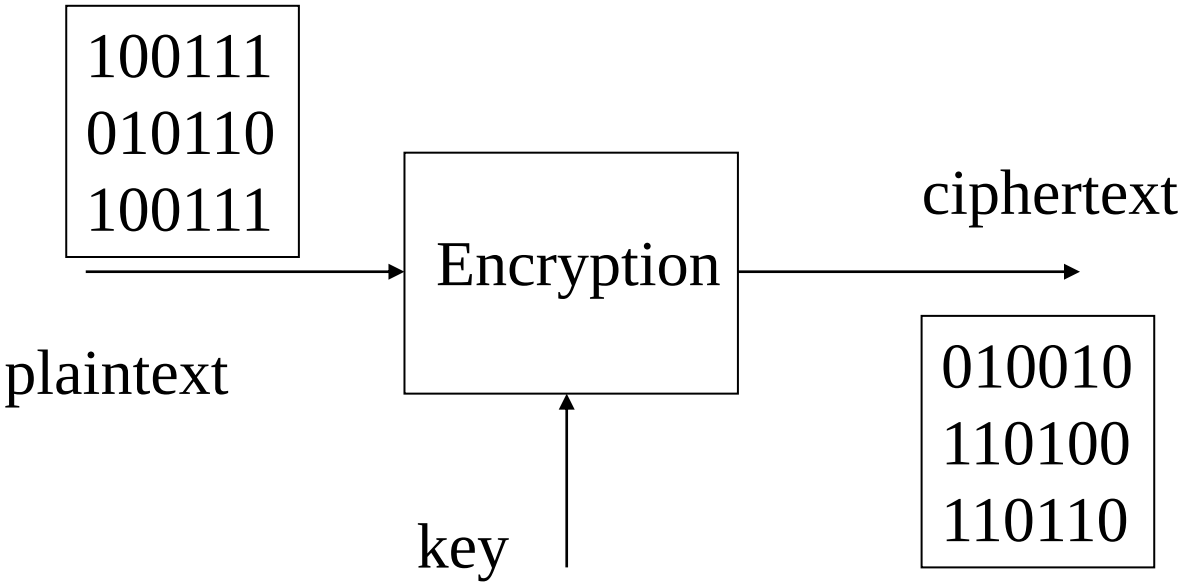
Stream Ciphers

- Security issues of stream ciphers
  - The security depends entirely on the insides of the keystream generator
    - If the keystream is an endless stream of zeros, ...
    - If the keystream is an endless random bits, we have a one-time pad.



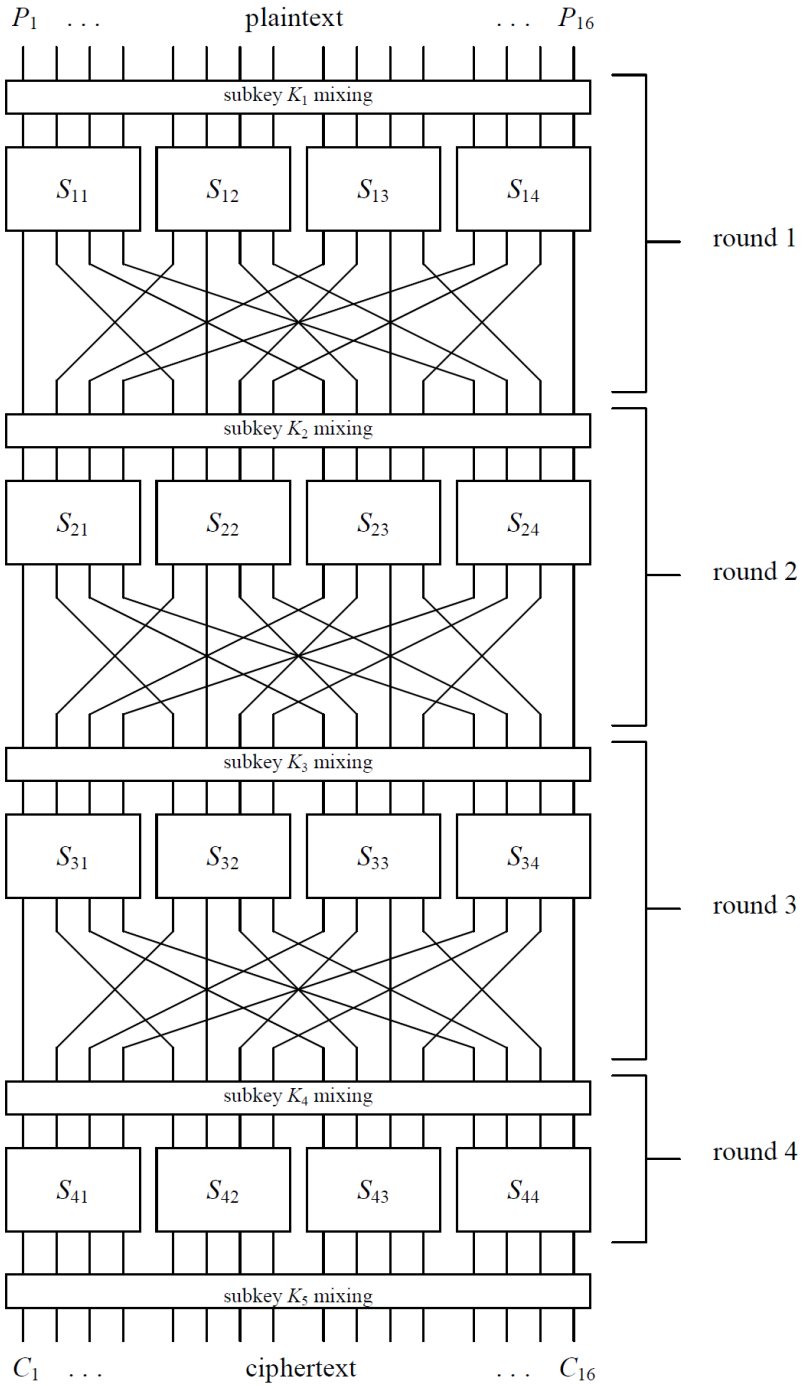
Block Ciphers

- A block of plaintext is encrypted as a whole to produce a ciphertext block of equal length



$$C = c_1\ c_2\ \ldots = E_k(p_1)\ E_k(p_2)\ \ldots$$

Block Ciphers

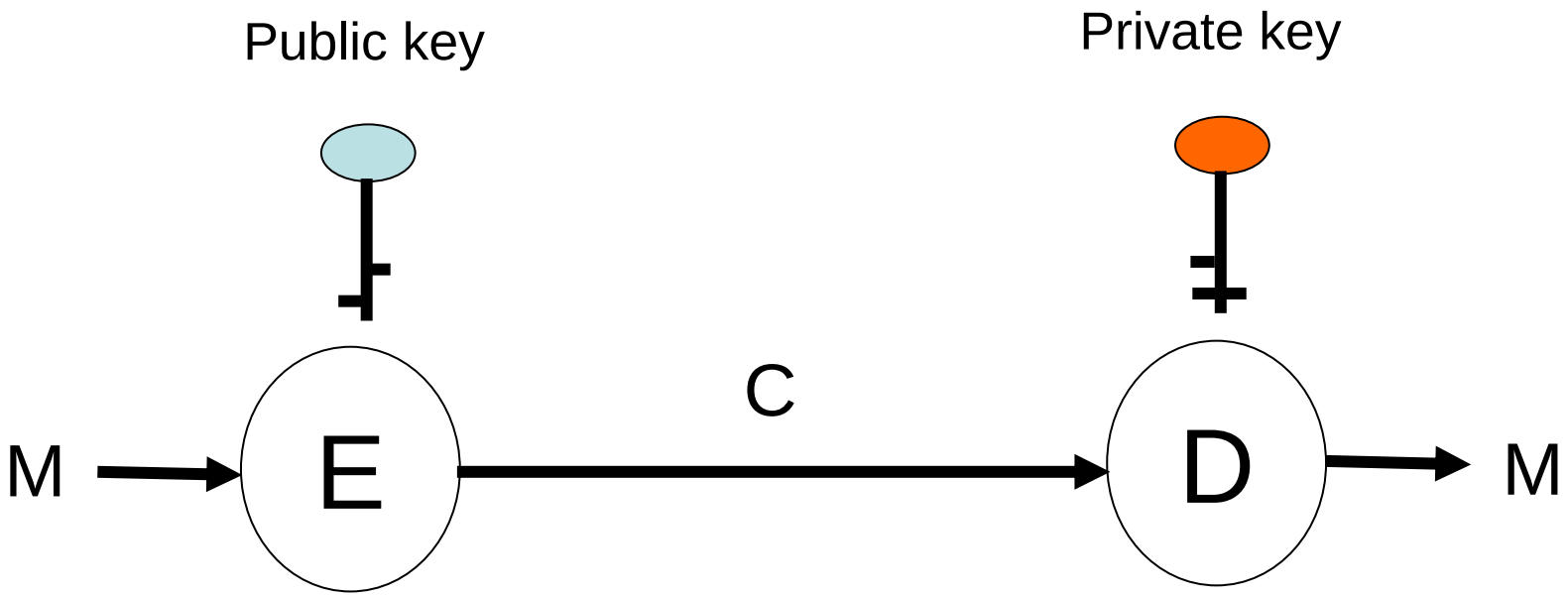


consists of a number of rounds, each round consists of XOR-ing the subkey (i.e., key mixing), substitutions, and a permutation

avalanche effect:

small changes in either plaintext or key should result in significant changes in the ciphertext.

Asymmetric-Key Encryption



Asymmetric Key Encryption

- A user generates a pair of public key and secret key and publishes his public key. The companion key is kept secret
- If Alice wants to send a message to Bob, she encrypts the message with Bob's public key.
- When Bob receives the message, he decrypts it with his secret key. Only Bob can decrypt it, because he is only party who has the secret key

Modular Arithmetic

- Define **modulo operator**

  $$b = a \bmod n$$

  to be the remainder when a is divided by n

- b is called the **residue** of a mod n
  - since it can be represented as: $a = qn + b$
- usually have $0 <= b <= n-1$
  - E.g. $-12 = -5 = 9 = 2 \pmod 7$

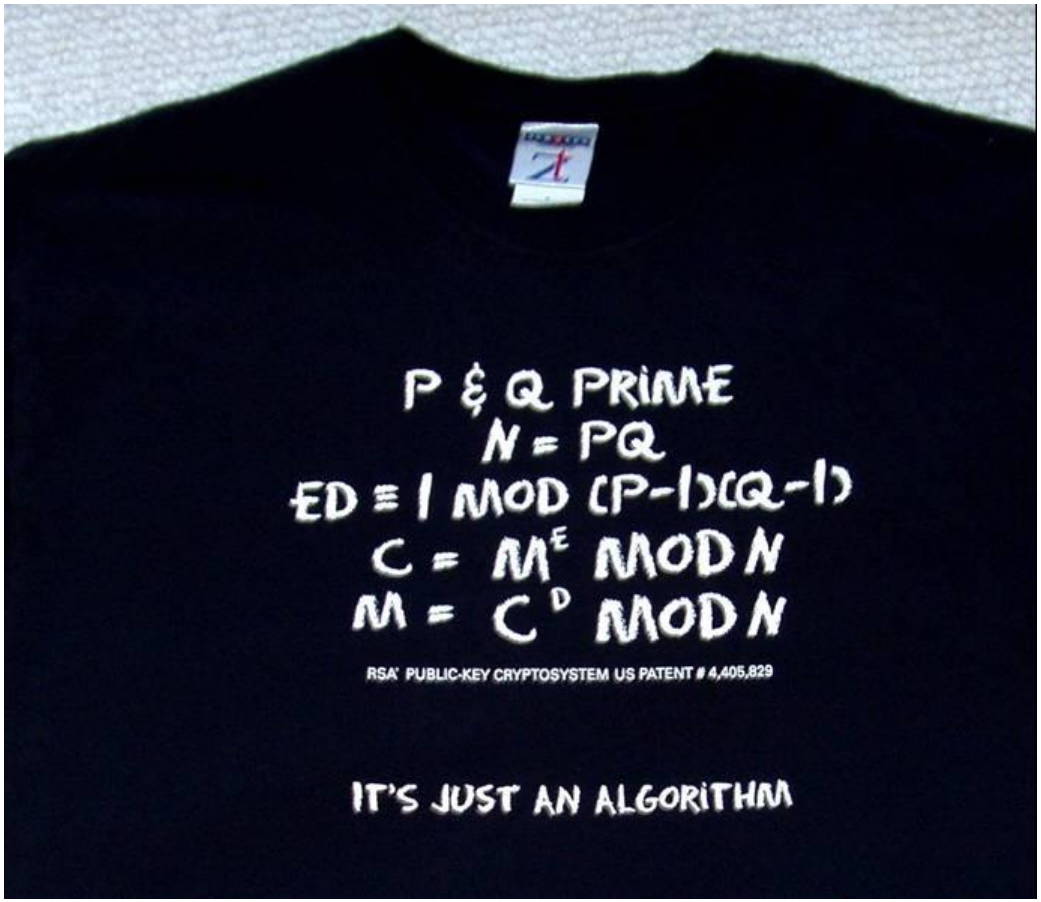
Modulo 7 Example

```
...
-21 -20 -19 -18 -17 -16 -15
-14 -13 -12 -11 -10  -9  -8
 -7  -6  -5  -4  -3  -2  -1
  0   1   2   3   4   5   6
  7   8   9  10  11  12  13
 14  15  16  17  18  19  20
 21  22  23  24  25  26  27
 28  29  30  31  32  33  34
...
```

RSA

- The RSA Public--Key Cryptosystem (**R**ivest, **S**hamir and **A**dleman (1978)) is the most popular and versatile PKC.

- RSA uses the knowledge that it is *easy* to find primes and multiply them together to construct composite numbers, but it is *difficult* to factor a composite number.


P & Q PRIME
N = PQ
ED ≡ 1 MOD (P-1)(Q-1)
C = M^E MOD N
M = C^D MOD N
RSA' PUBLIC-KEY CRYPTOSYSTEM US PATENT # 4,405,829
IT'S JUST AN ALGORITHM

The Textbook RSA

1. Choose two large primes p and q. Compute n = pq and $m=\phi(n)= (p-1)(q-1)$.
   - $\phi(n)$ is Euler's totient function: It is the number of positive integers less than n that are relatively prime to n.
2. Choose e, $1 \leq e \leq m - 1$, such that gcd(e,m)=1.
3. Finds d such that $ed=1 \bmod m$.
   - This is possible because of the choice of e.
   - d is the **multiplicative inverse** of e modulo m and can be found using the extended Euclidean (gcd) algorithm.
4. The **Public key** is **(e, n)**.

   The **Private key** is **(d, n)**.

RSA Encryption and decryption

- If Bob want to encrypt a message **X** for Alice. He uses Alice's public key and computes the ciphertext **Y** as

$$\mathbf{Y = X^e \bmod n}$$

- When Alice wants to decrypt **Y**, she uses the private key and calculates

$$\mathbf{X = Y^d \bmod n}$$

- **X** and **Y** are both integers in {0, 1, …, n-1}.

- **Example**: Choose p=11 and q=13.

  n=11*13=143

  m=(p-1)(q-1)=10*12=120

  e=37 ➔ gcd(37,120)=1

  Use the extended gcd algorithm we find d such that ed=1 mod 120:

  d=13 → de=481 = 1 mod 120.

- To encrypt a message X = 3

$$Y = X^e \bmod n = 3^{37} \bmod 143 = 42$$

- To decrypt Y

$$X = Y^d \bmod n = 42^{13} \bmod 143 = 3$$

The ElGamal Encryption

- The security of this system relies on another hard problem
  - Discrete log problem
- Operates on group Z_p^* = {1, 2, …, p-1} where p is a large prime number

Generator of Z_p^*

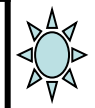
- An element α is a generator (or primitive root) of Z_p^* if

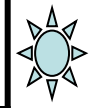
  $\alpha^i \pmod p$ for $0 < i \leq p-1$

  generates all numbers *1,.. p-1*

An example : Z_{11}^*

	1	2	3	4	5	6	7	8	9	10
1	1	1								1
2	**2**	**4**	**8**	**5**	**10**	**9**	**7**	**3**	**6**	1
3	3	9	5	4	1	3				1
4	4	5	9	3	1	4				1
5	5	3	4	9	1	5				1
6	**6**	**3**	**7**	**9**	**10**	**5**	**8**	**4**	**2**	1
7	**7**	**5**	**2**	**3**	**10**	**4**	**6**	**9**	**8**	1
8	**8**	**9**	**6**	**4**	**10**	**3**	**2**	**5**	**7**	1
9	9	4	3	5	1	9				1
10	10	1	10							1






Discrete Logarithm Problem

INPUT:

- Z_p^*
- g *in* Z_p^* , a generator of Z_p^*
- h in Z_p^*

Find the unique number $a<p$ such that
$h=g^a \bmod p$

- **DL Assumption:** There is no efficient algorithm to solve DL problem.
- It is widely believed that this assumption holds.

The ElGamal Cryptosystem

- **Key generation**:
  - Alice chooses a prime **p** and two random numbers **g** and **u**, both less than **p**, where **g is** a generator of $\mathbf{Z_p^*}$.
  - Then she finds:

$$\mathbf{y = g^u} \bmod \mathbf{p}$$

Alice's public key is **(p, g, y)**, her secret key is **u**.

- To encrypt a message **X** for Alice, Bob chooses a random number **k** < p - 1. Then he calculates:

$$a = g^k \bmod p$$
$$b = y^k \times X \bmod p$$

- The ciphertext is **(a,b)**
- The length is twice the length of the plaintext.

- To decrypt **(a,b)** Alice calculates

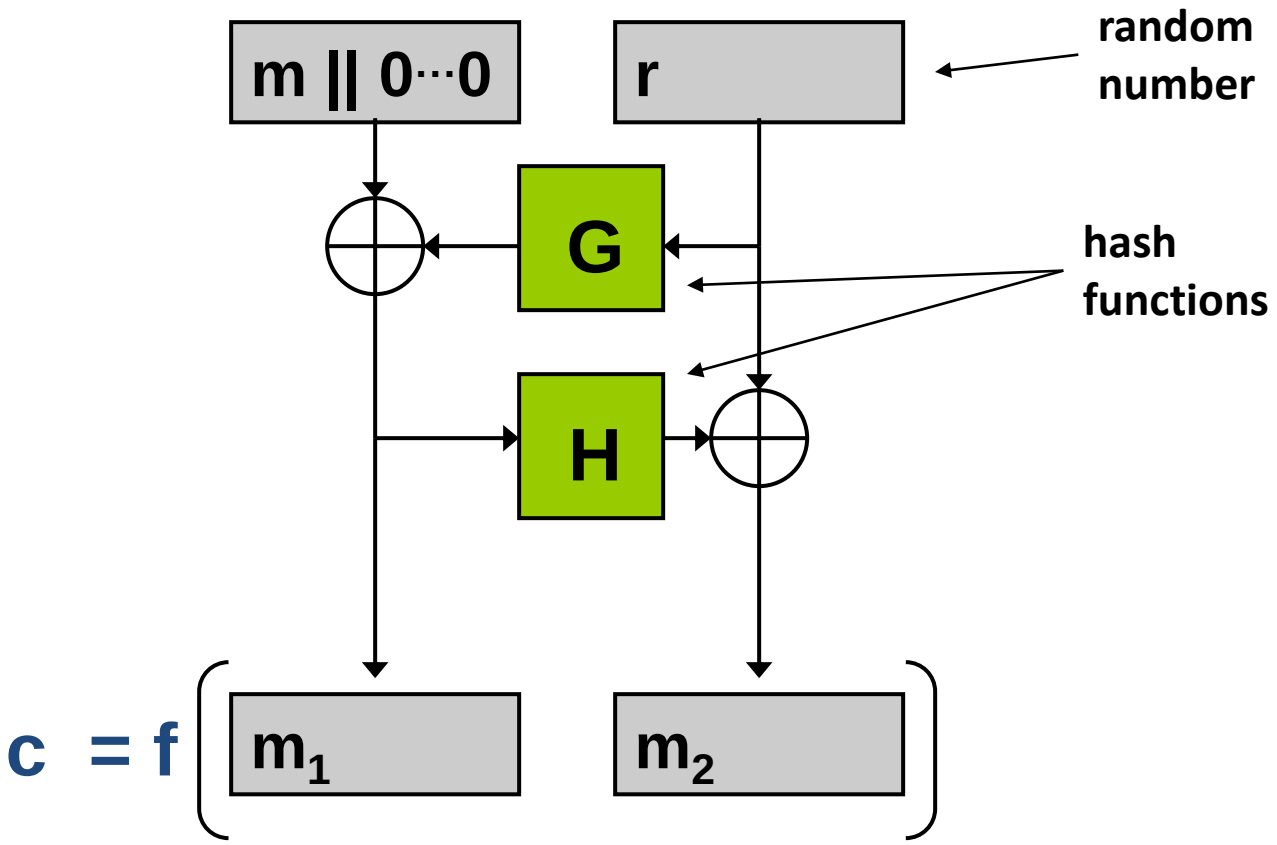
$$X = \frac{b}{a^u} \bmod p$$

Division means calculating the inverse mod p

RSA vs ElGamal

- What are the differences between these two cryptosystems?
- Which one is more suitable when encrypting a message from a small domain (e.g., 0 - 9999)?

RSA in Reality
Optimal Asymmetric Encryption Padding (OAEP)



f: RSA Encryption function

TOOLS FOR INTEGRITY AND AUTHENTICITY

Message Authentication Code

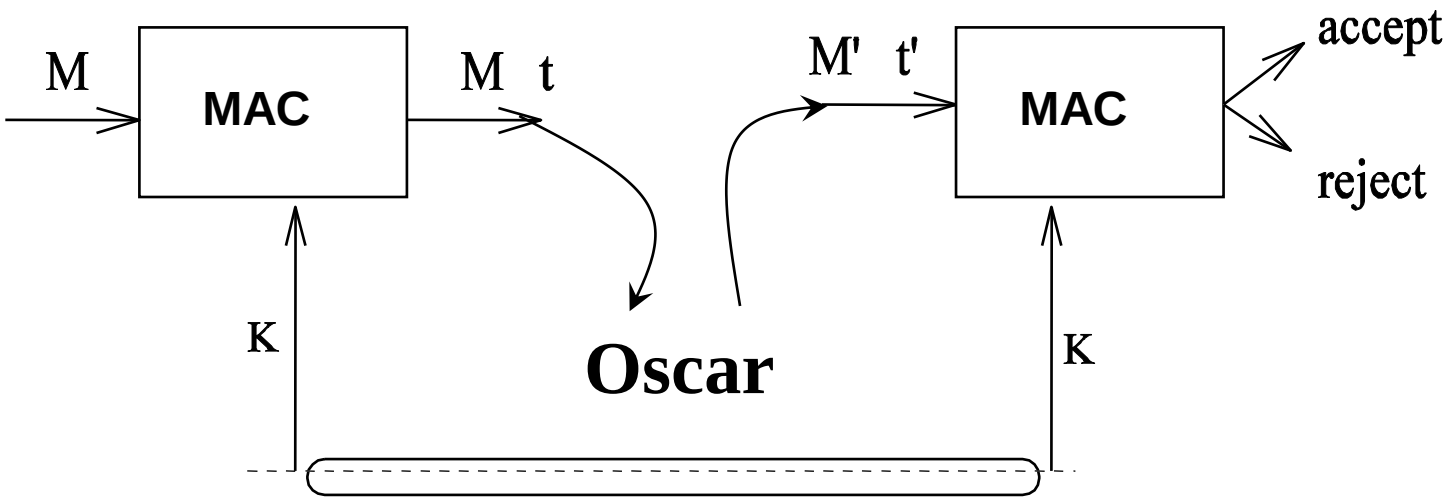
- Symmetric tool for message integrity and authenticity
- Produce a cryptographic checksum
- Common constructions
  - Hash function based
  - Block cipher based

What is message integrity

- Preventing unauthorised modification of data
- Different from error detection
  - This is for unintentional modification of data (e.g., due to noise)
- Both involve a checksum
  - Integrity check value is based on the message and a secret key
  - Error correction/detection code does not use secret key

Message Authentication Code

- Transmitter and receiver share a secret key **K**. To transmit **M**, the transmitter calculates a **MAC** and appends it to **M**, thus $\mathbf{t=MAC_K(M)}$.



- The receiver receives a message **(M, t)**. It uses the key **K** and **M** to calculate $\mathbf{MAC_K(M)}$ and compare it with **t**. If the two match, the received message is accepted as authentic.

- The **MAC** is also called a **cryptographic checksum**.

HMAC

- HMAC uses cryptographic hash functions such as SHA-2/3, etc.
- Treat a **hash function** as a “black box”
  - Existing implementation of hash functions can be used
  - It is easy to replace a hash function (if it is not secure)

Cryptographic hash functions

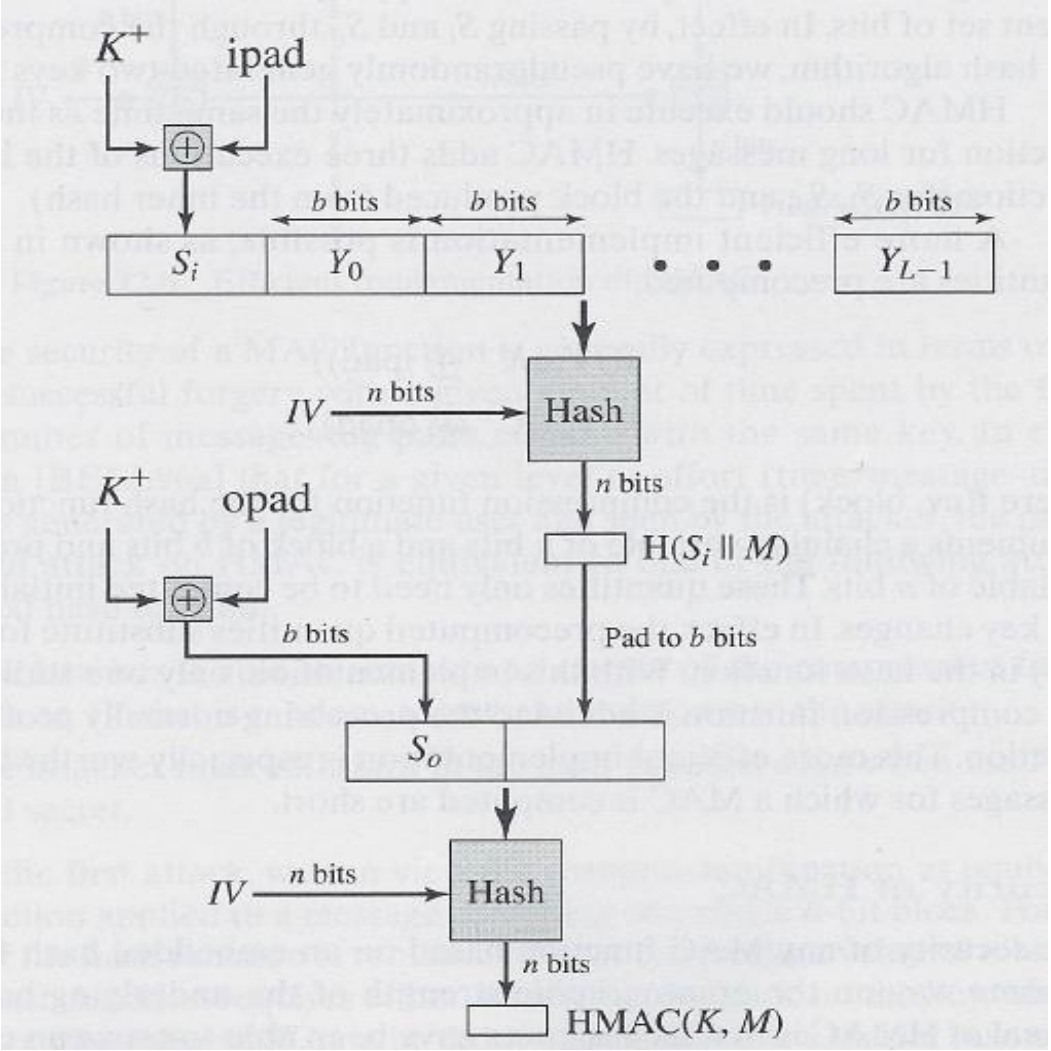
- We require the following properties for cryptographic hash functions:

1. It can be applied to any size input.
2. The output must be of fixed size.
3. Easy to compute.

4. **Pre-image resistant**: For any given Y, it is difficult to find an X such that H(X)=Y.
5. **Collision resistant**: It is computationally infeasible to find messages X and Y with X≠Y such that H(X)=H(Y).

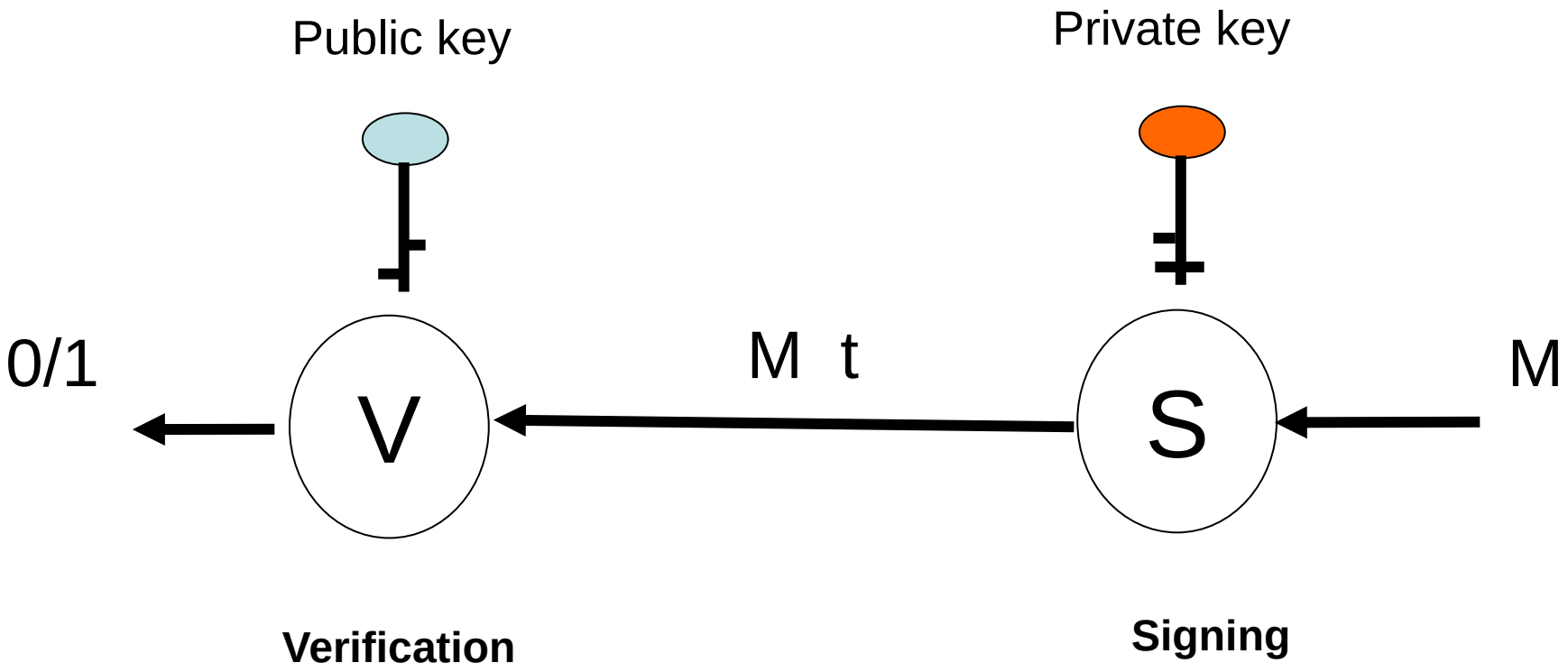
HMAC(K,M) = H(K ⊕ opad || H((K ⊕ ipad) || M))

- Y_i is ith block of M
- b is number of bits in a block
- K^+ is K padded with zeros on the left
- ipad = [0x36 * blocksize]
- opad = [0x5c * blocksize]



Digital Signature

Public key analogy of message authentication codes



RSA Signature Scheme

- Key Generation:
  - Generate primes P and Q, compute N = PQ
  - Generate d and e such that de = 1 mod (P-1)(Q-1)
  - Public Key (N, e)
  - Private Key d
- SIGN:
  - Given message m, compute s = m^d mod N
- VER:
  - Given message m, signature s, check if m = s^e mod N

Example

- Key Generation:
  - P = 13, Q = 17
  - N = P*Q = 221
  - Let’s say e = 5.
    - d = 77
  - Public Key (221, 5)
  - Private Key (221, 77)

Example

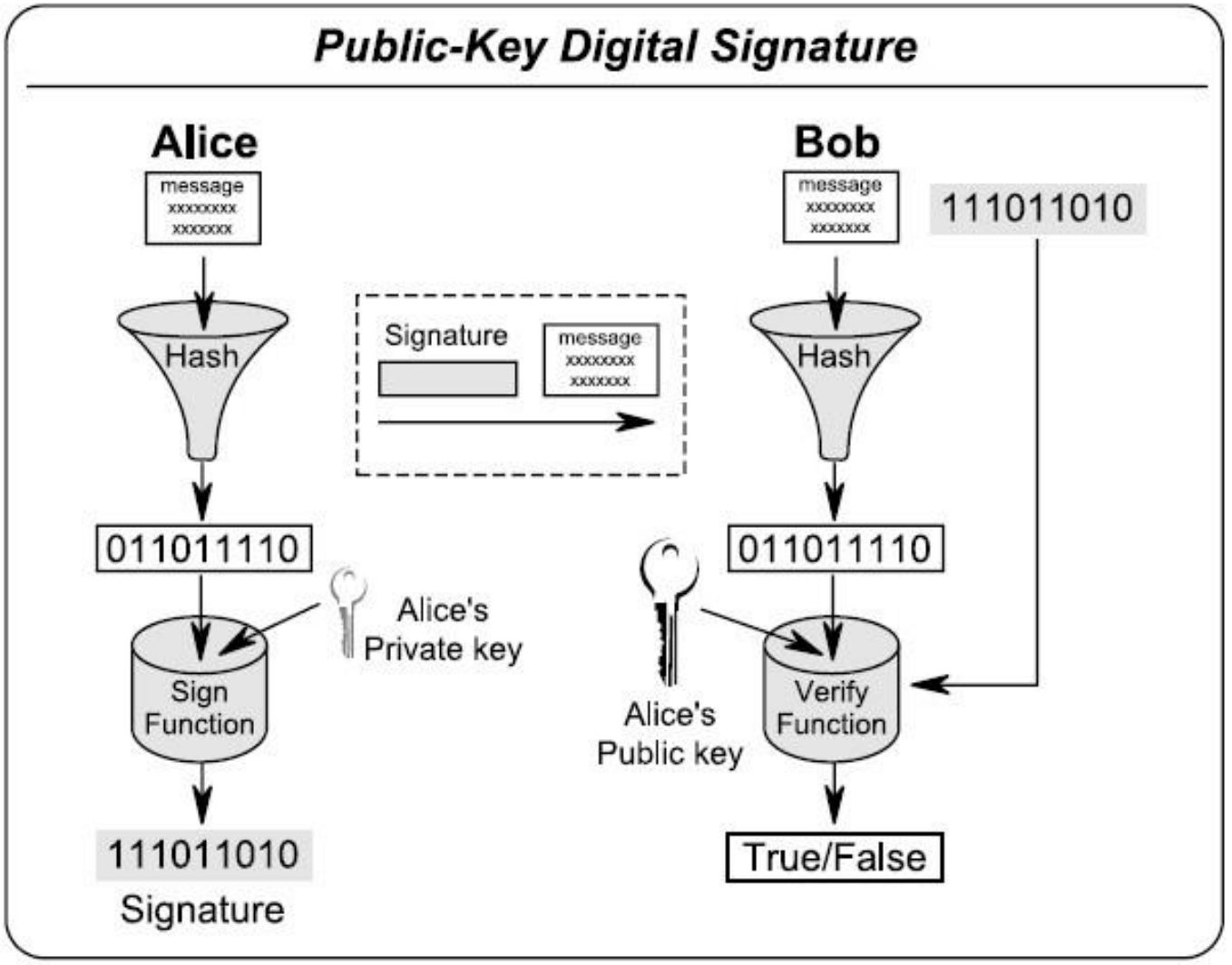
- SIGN:
  - Suppose we would like to sign a message (124)
  - $s = 124^{77} \bmod 221$
  - $s = 37 \bmod 221$
  - The message-signature pair is (124, 37)

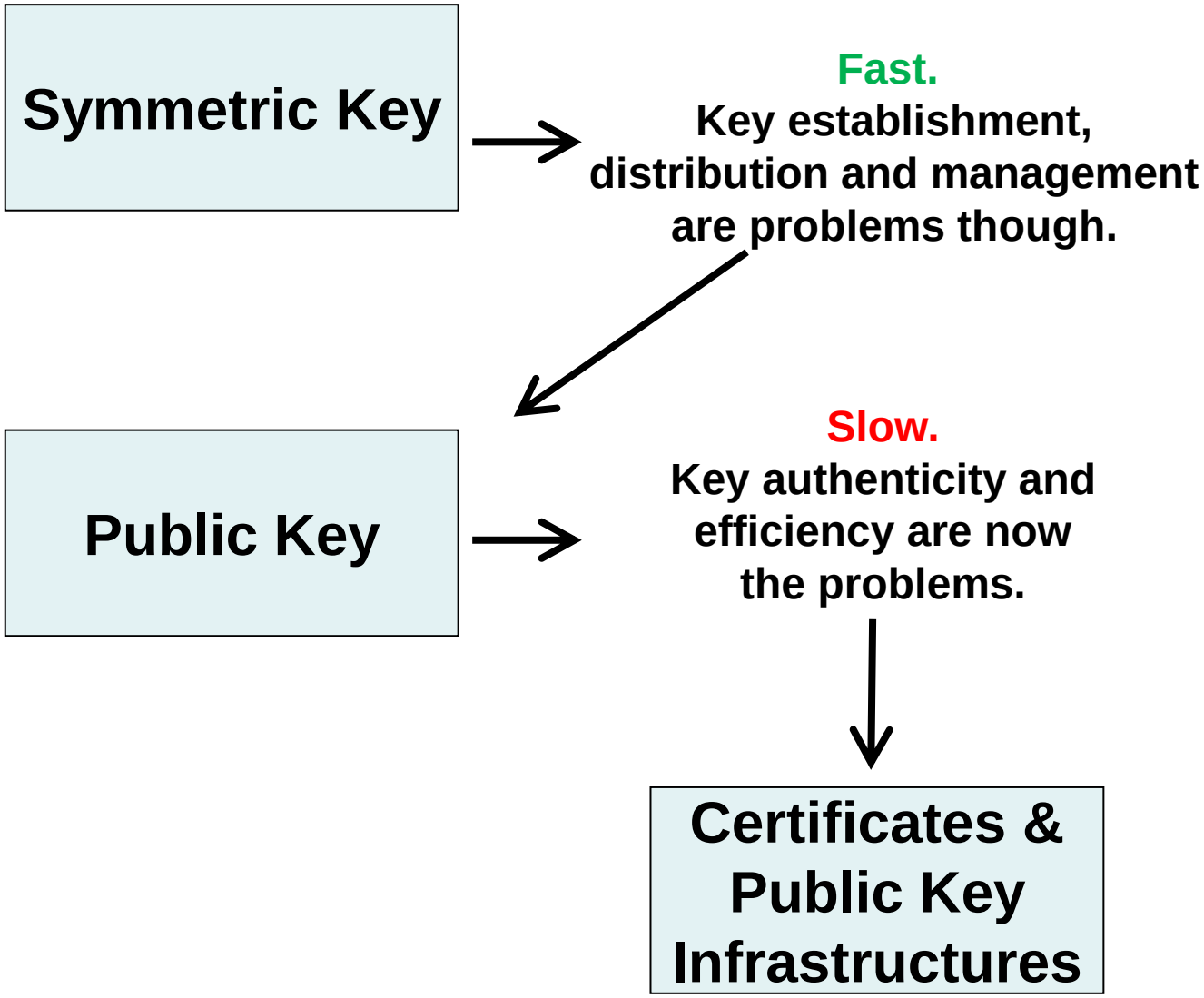
Example

- VER:
  - Given a message-signature pair is (124, 37)
  - Anyone can verify the signature by computing
  - $s^e \mod N$ and see if it is equal to m.
  - $37^5 \mod 221 = 124 \mod 221$
  - Return True.

Question: how do you sign a long message?

Hash-then-Sign





Hybrid System: PKC (with PKI) + one-time Symm Key