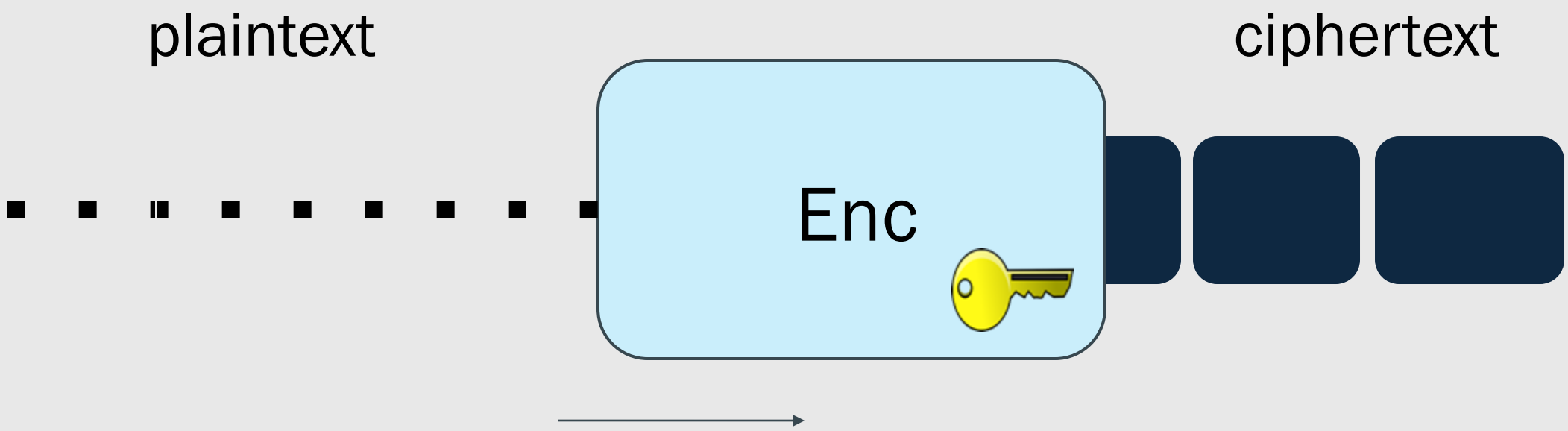
COMP435: *SECURITY CONCEPTS!*

Lecture 8: Public Key Encryption

tinyurl.com/comp435-fa25

BLOCK CIPHER MODES OF OPERATION

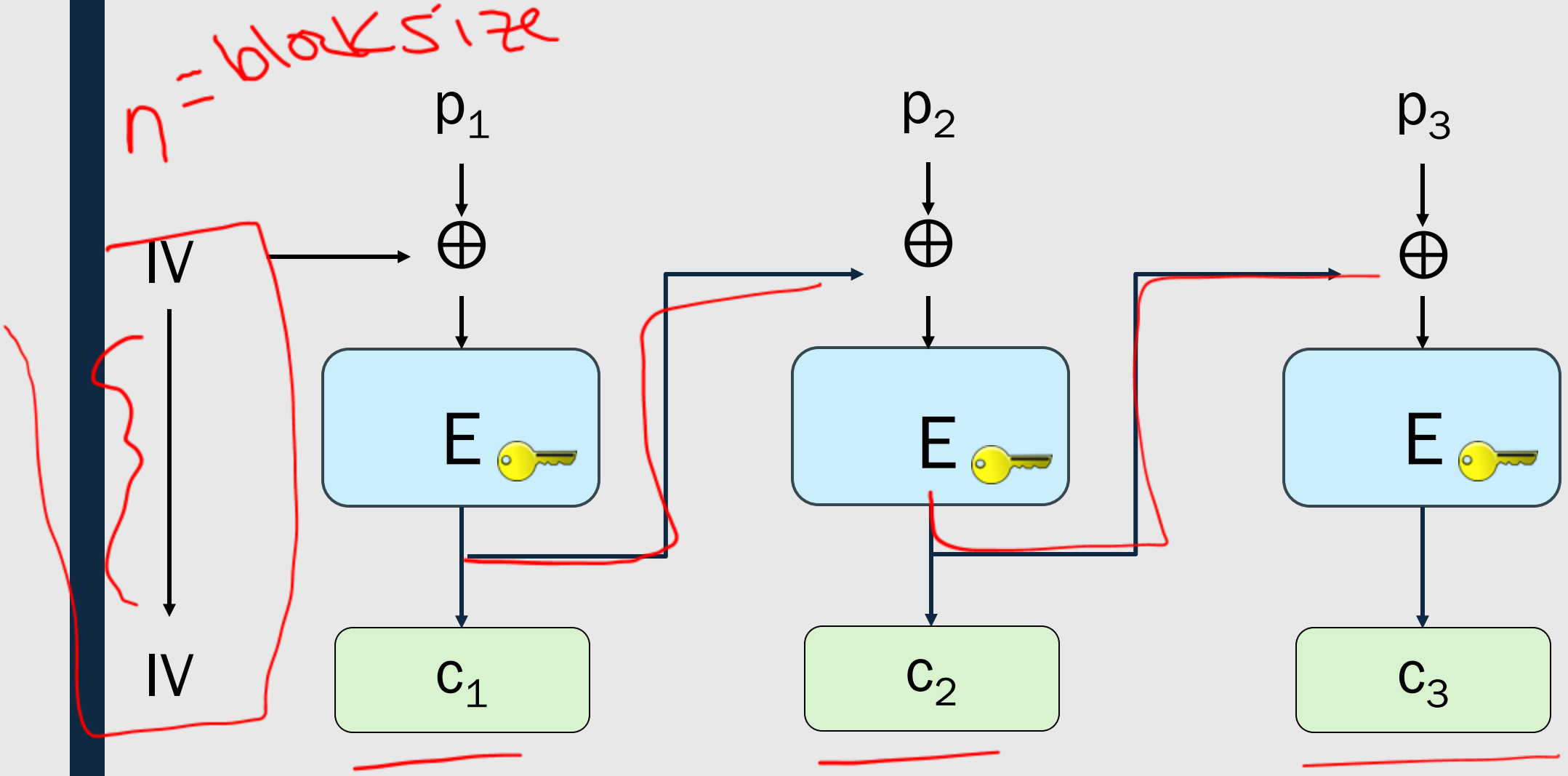
Block Ciphers



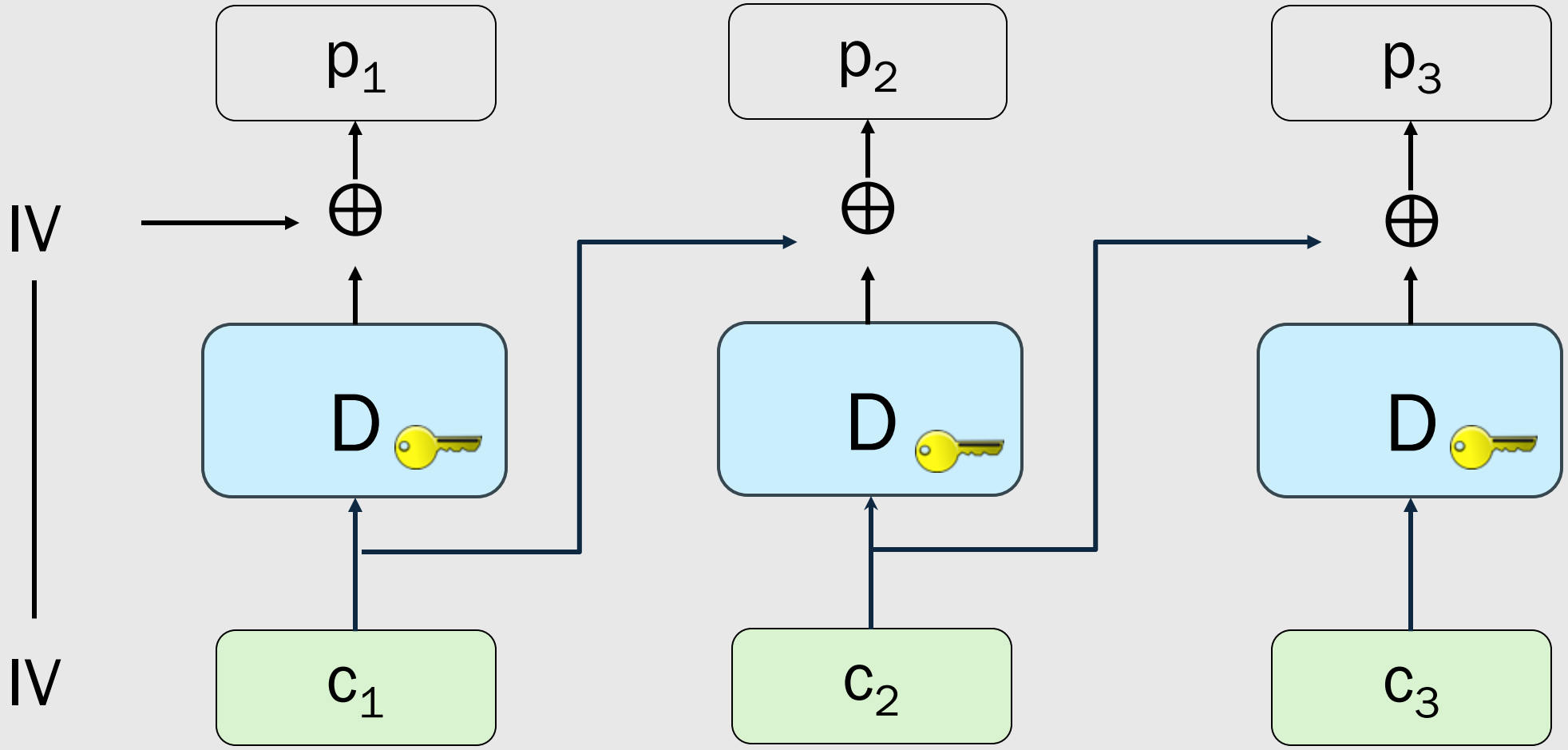
Modes of Operation

- Electronic Code Book Mode
- Cipher Block Chaining Mode
- Output Feedback Mode
- Counter Mode

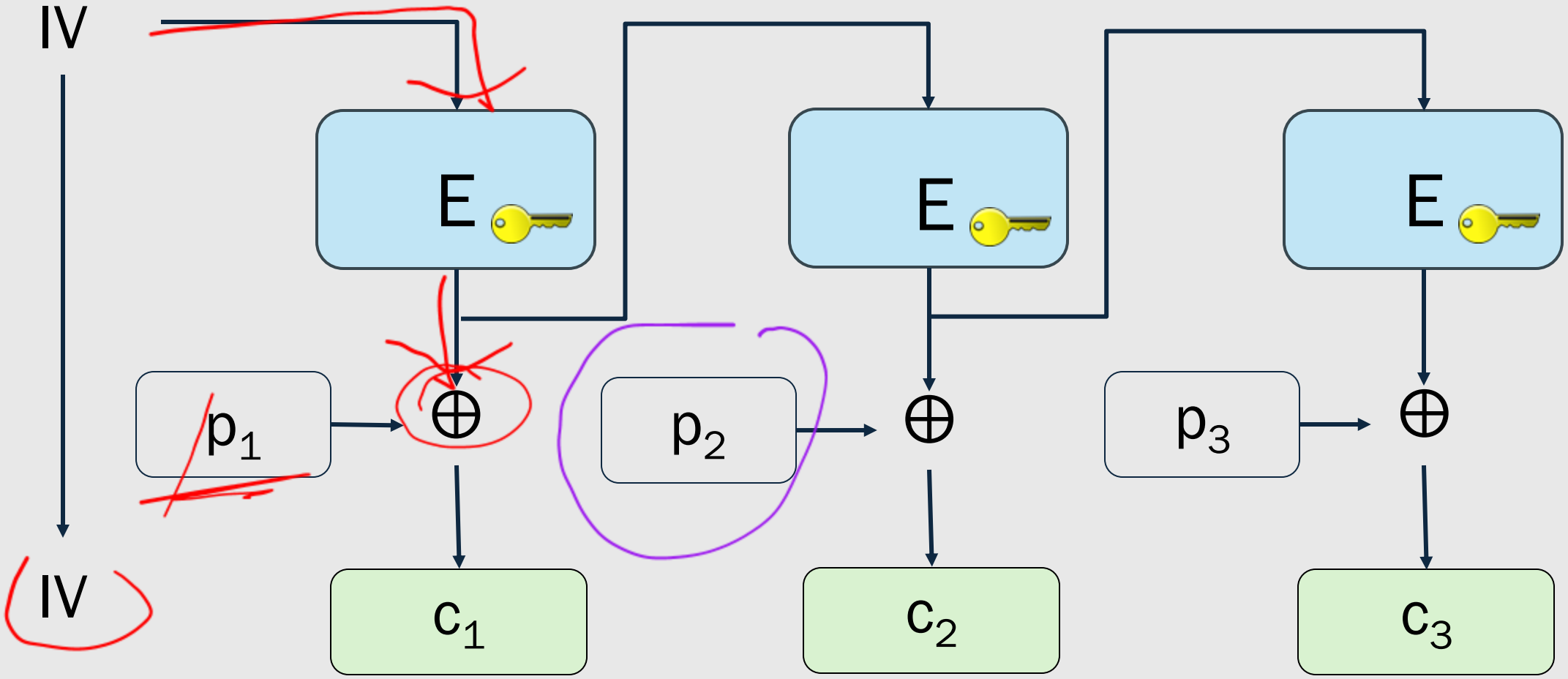
Cipher Block Chaining (CBC) Mode: Encryption



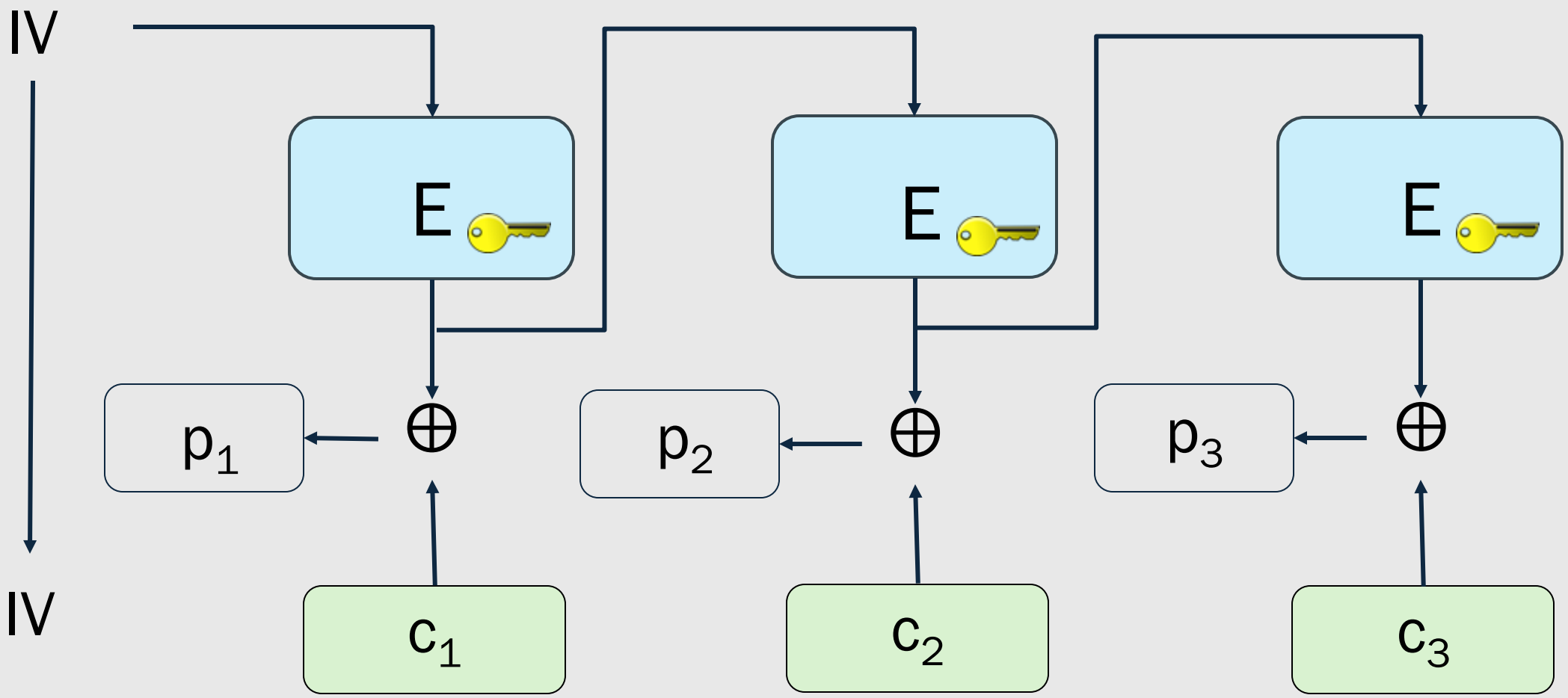
CBC Mode Decryption



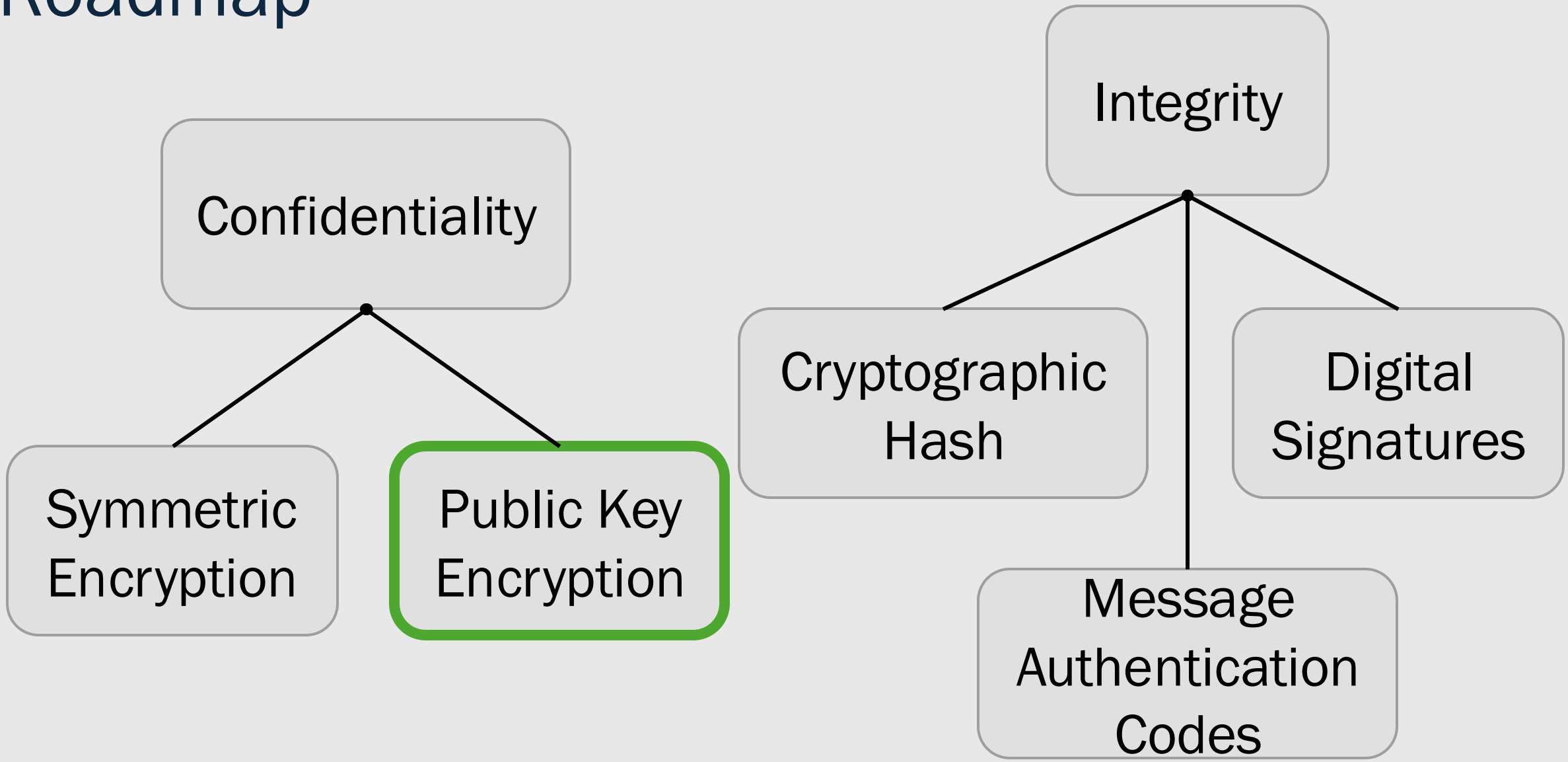
Output Feedback (OFB) Mode: Encryption



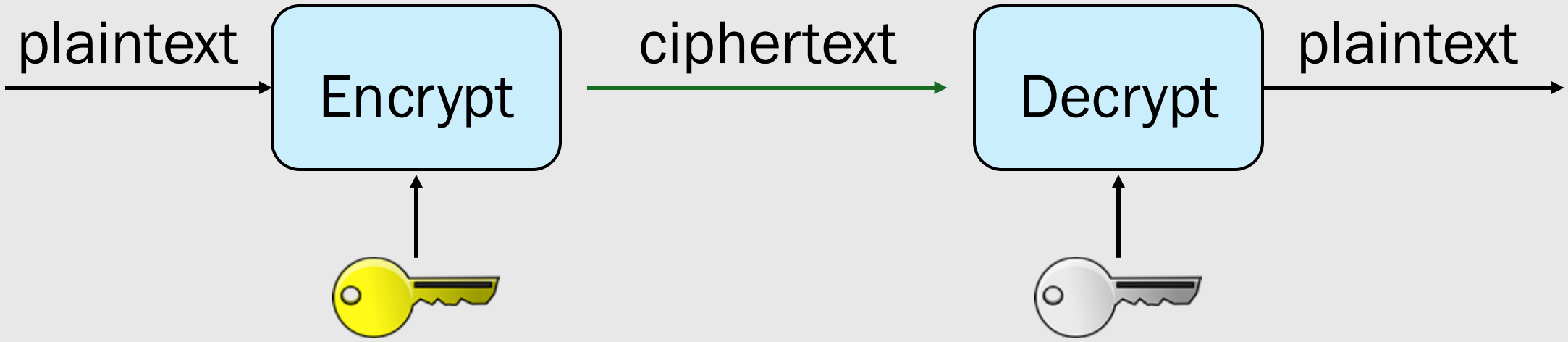
Output Feedback (OFB) Mode: Decryption



Roadmap

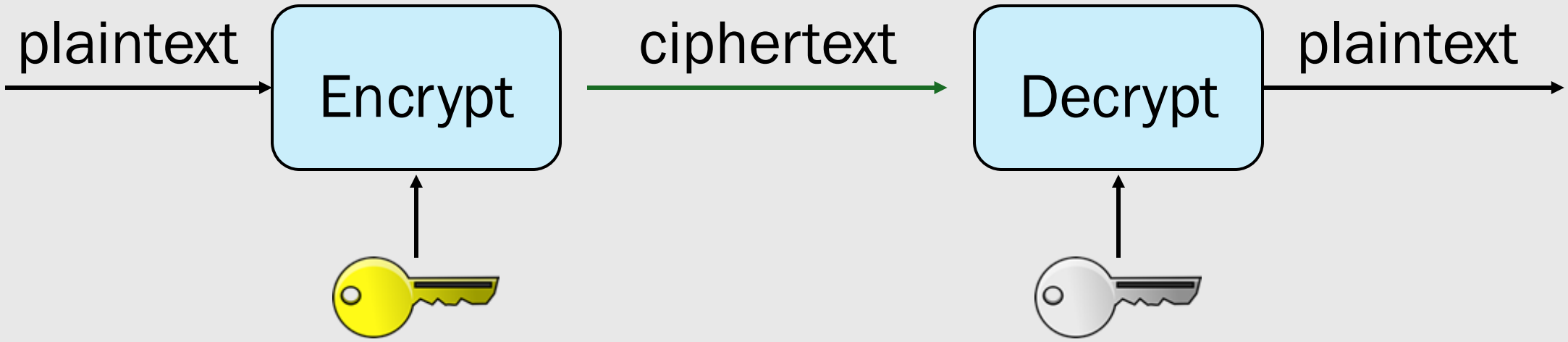

Confidentiality
Symmetric Encryption
Public Key Encryption
Integrity
Cryptographic Hash
Digital Signatures
Message Authentication Codes

Public Key Encryption



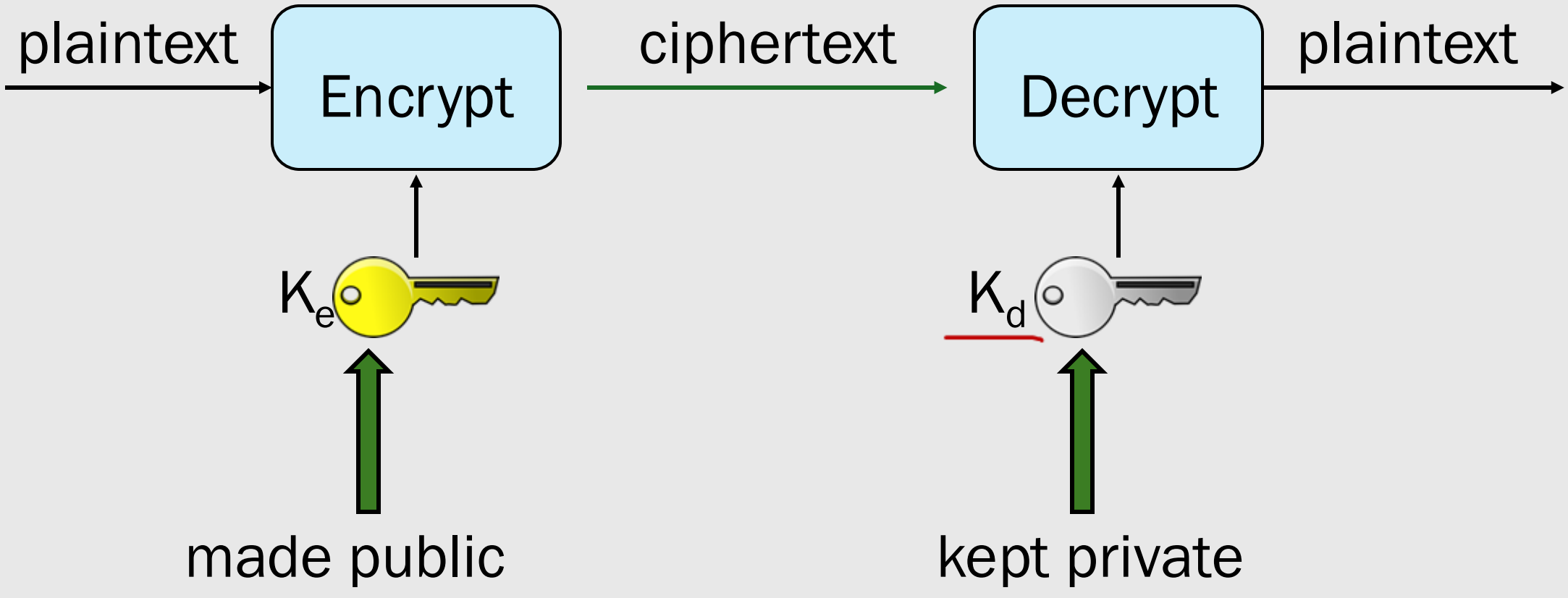
Public Key Encryption

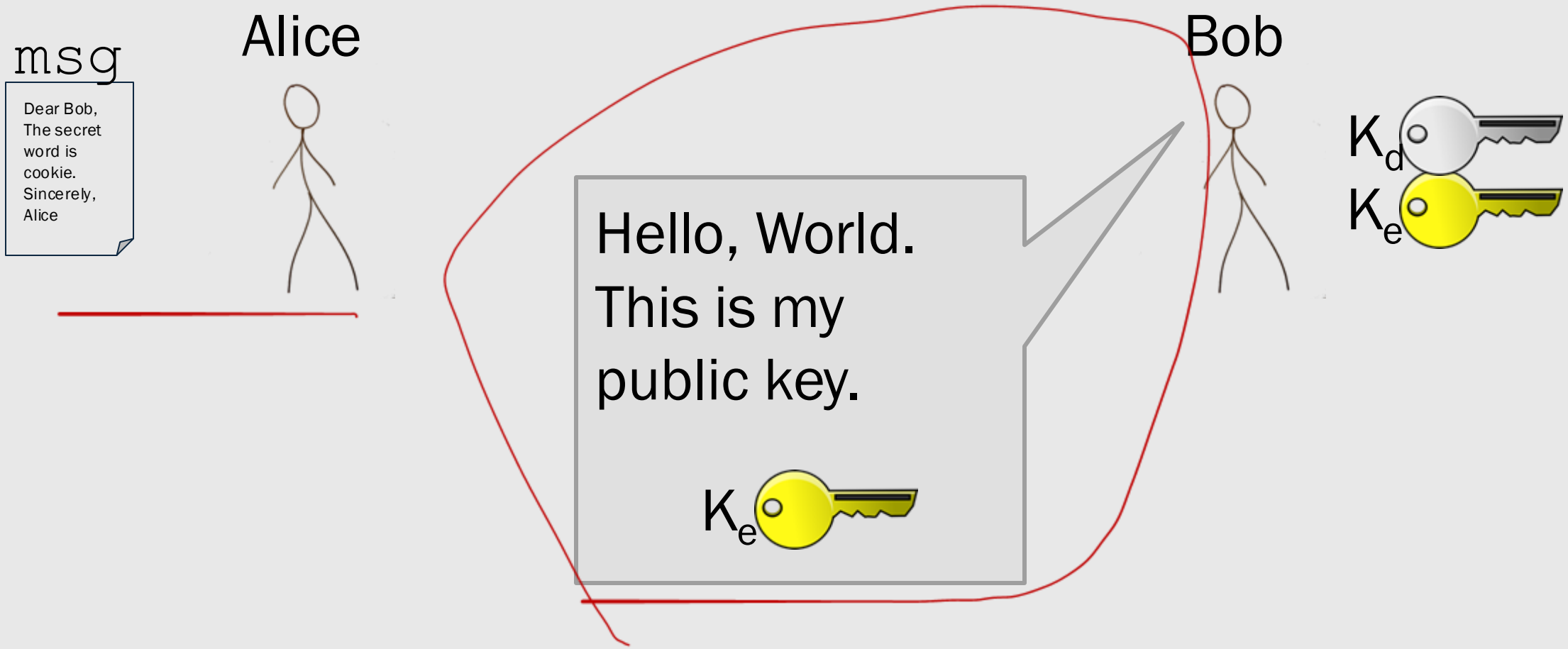
$$c = Enc_{Ke}(msg)$$
$$msg = Dec_{Kd}(c)$$



Public Key Encryption

$$c = Enc_{Ke}(msg)$$
$$msg = Dec_{Kd}(c)$$



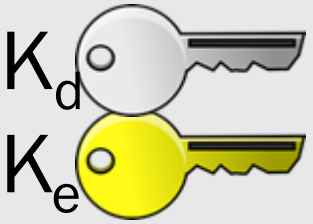

msg
Dear Bob,
The secret
word is
cookie.
Sincerely,
Alice
Alice
Bob
Hello, World.
This is my
public key.
K_e
K_d
K_e

msg

Dear Bob,
The secret word is cookie.
Sincerely,
Alice

Alice

Bob



$c = Enc_{Ke}(msg)$

c

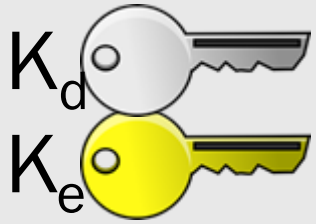
$msg = Dec_{Kd}(c)$

msg

Dear Bob,
The secret word is cookie.
Sincerely,
Alice

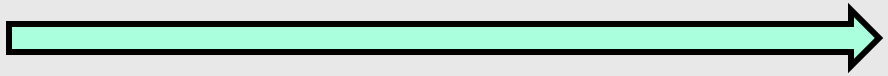
Alice

Bob



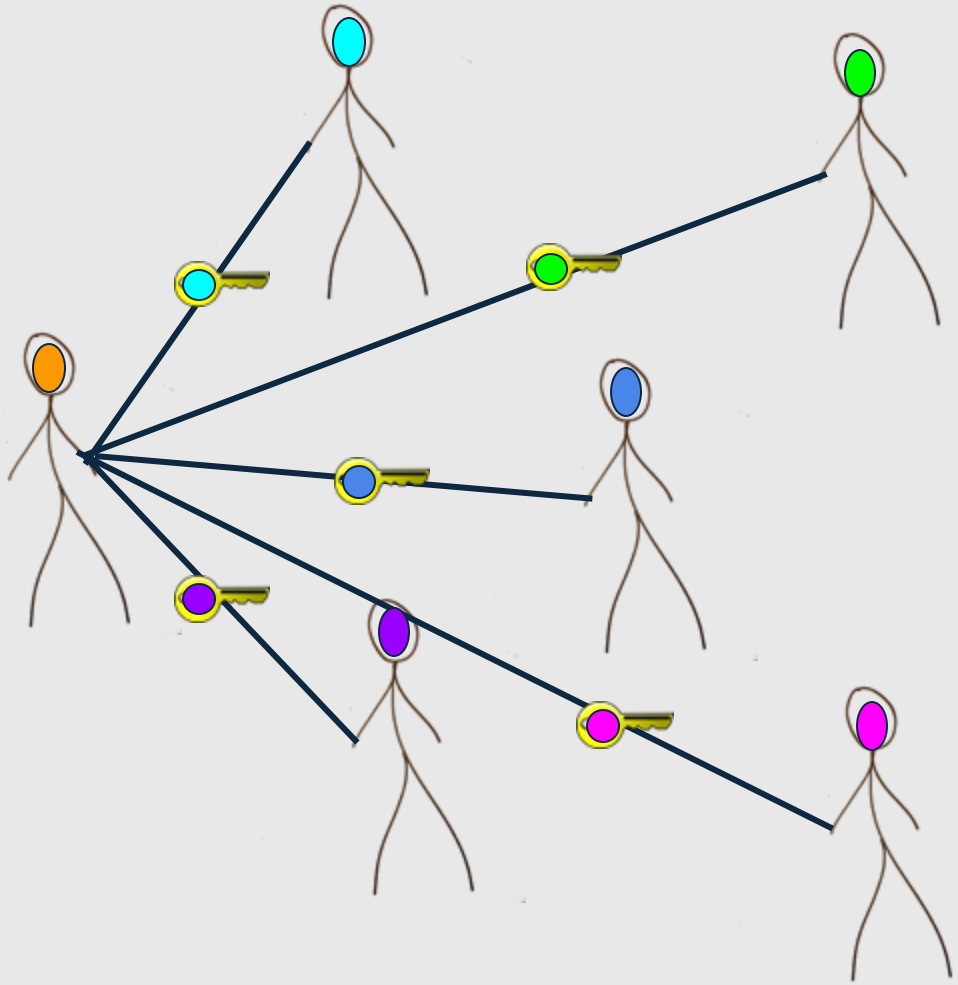
$c = Enc_{Ke}(msg)$

c

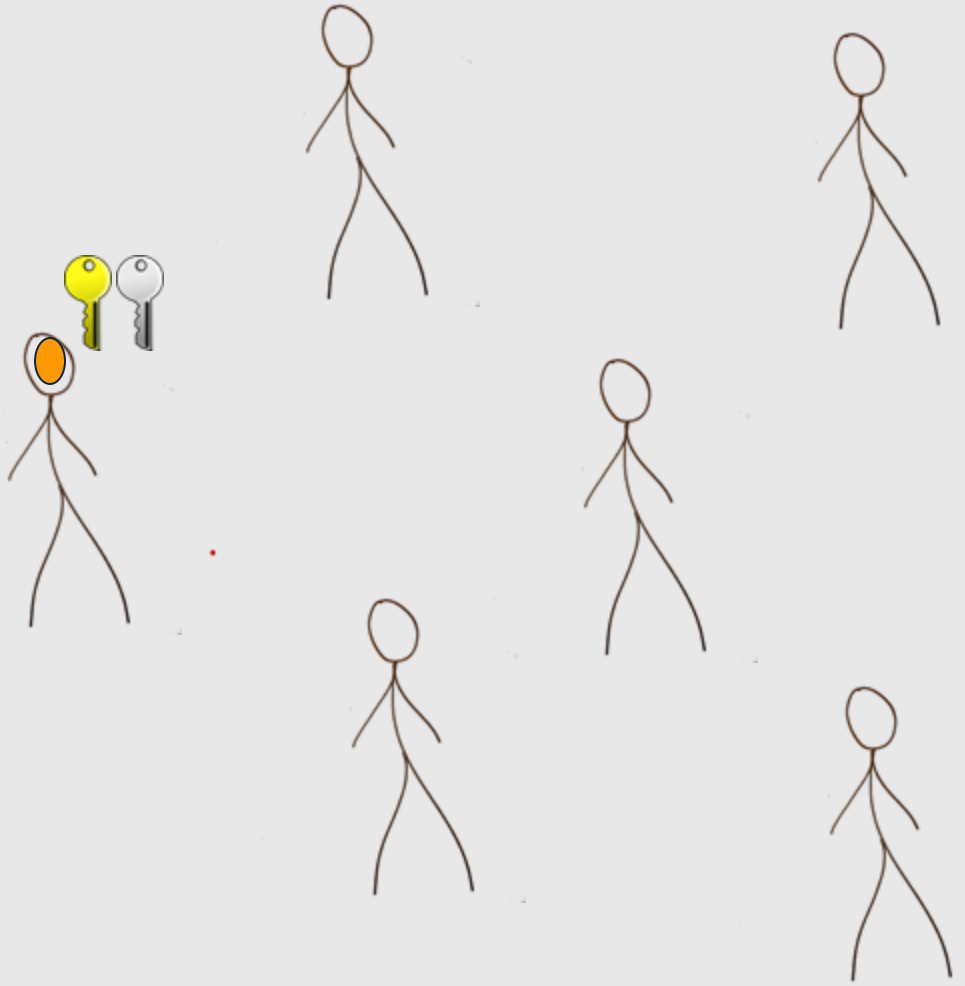


$msg = Dec_{Kd}(c)$

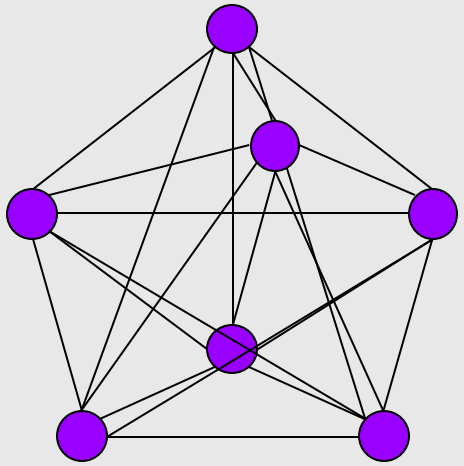
Symmetric Encryption



Public Key Encryption



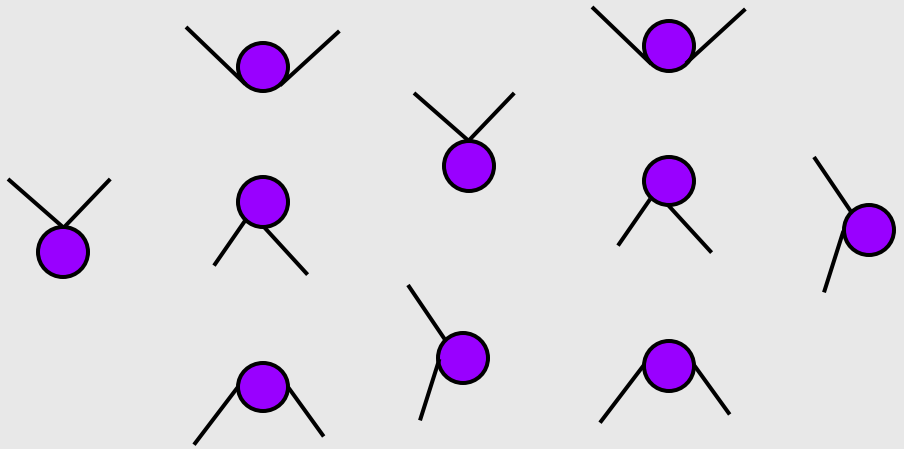
Symmetric Encryption



$$\frac{n(n-1)}{2} \text{ keys}$$

$O(n^2)$

Public Key Encryption



2n keys

$O(n)$

RSA

- Ron Rivest, Adi Shamir, Leonard Adleman
- 1978
- Turing Award (2002)
- Based on the hardness of factoring
- Keys are 1024-, 2048-bits long

RSA Key Generation

1. Input parameters: Given *l* >2 and odd e > 2:
2. Generate prime *p* s.t.
   - *p is prime, is l-bits longs, and* $gcd(e, p-1) = 1$
3. Generate prime *q* s.t.
   - $q \neq p$, *q is prime, is l-bits longs, and* $gcd(e, q-1) = 1$
4. Compute N, where N := *pq*. N is the RSA modulus
5. Compute private exponent, d, s.t. $e*d \equiv 1 \bmod (p-1)(q-1)$

$$e \times d \equiv 1 \bmod (p-1)(q-1)$$

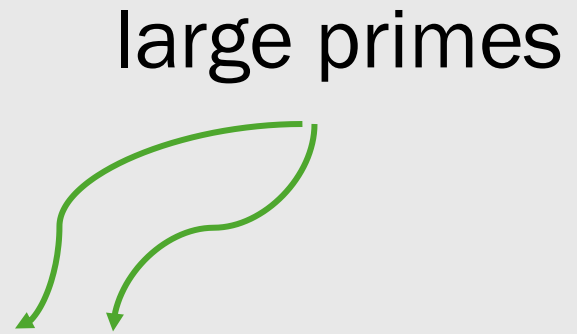
RSA Key Generation

1. Input parameters: Given *l* >2 and odd e > 2:
2. Generate prime *p* s.t.
   - *p is prime, is l-bits longs, and gcd(e, p-1) = 1*
3. Generate prime *q* s.t.
   - *q ≠ p, q is prime, is l-bits longs, and gcd(e, q-1) = 1*
4. Compute N, where N := *pq*. N is the RSA modulus
5. Compute private exponent, d s.t. e*d ≡ 1 mod (*p*-1)(*q*-1)

The public key is (N, e)

The private key is (N, d)

RSA



public key $\longrightarrow K_e = (e, N)$

private key $\longrightarrow K_d = (d, N)$

RSA

public key → $K_e = (e, N)$

private key → $K_d = (d, N)$

$ed \equiv 1 \mod \phi(N)$

RSA Encryption and Decryption

- Encryption: $c = msg^e \bmod N$
- Decryption: $msg = c^d \bmod N$

Given N, e and msg^e, it is difficult to compute msg

Given N, e, msg^e, and msg, it is difficult to compute d

RSA Correctness

- $msg = (msg^e)^d \bmod N$
- Thm.
  For all primes p, q, $p \neq q$,
  Let $N = pq$,
  $ed \equiv 1 \pmod{(p-1)(q-1)} \rightarrow$
  $\forall x \ni \mathbb{Z}, x^{ed} \equiv x \pmod{N}$.

https://toc.cryptobook.us/book.pdf

One-Way Functions

- a function F: X ⟶ Y is one-way if F is easy to compute but hard to invert

- multiplying two large primes is one-way
  - *multiplying pq = N is easy to compute*
  - *factoring N to find p, q is hard*

Trap-Door Functions

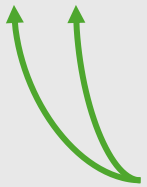
- a one-way function F: X ⟶ Y is a trap-door function if there exists a secret to make inversion easy

- inverting x^e mod N is hard
  - *d is the trap door*
  - *computing x^{ed} mod N is easy*

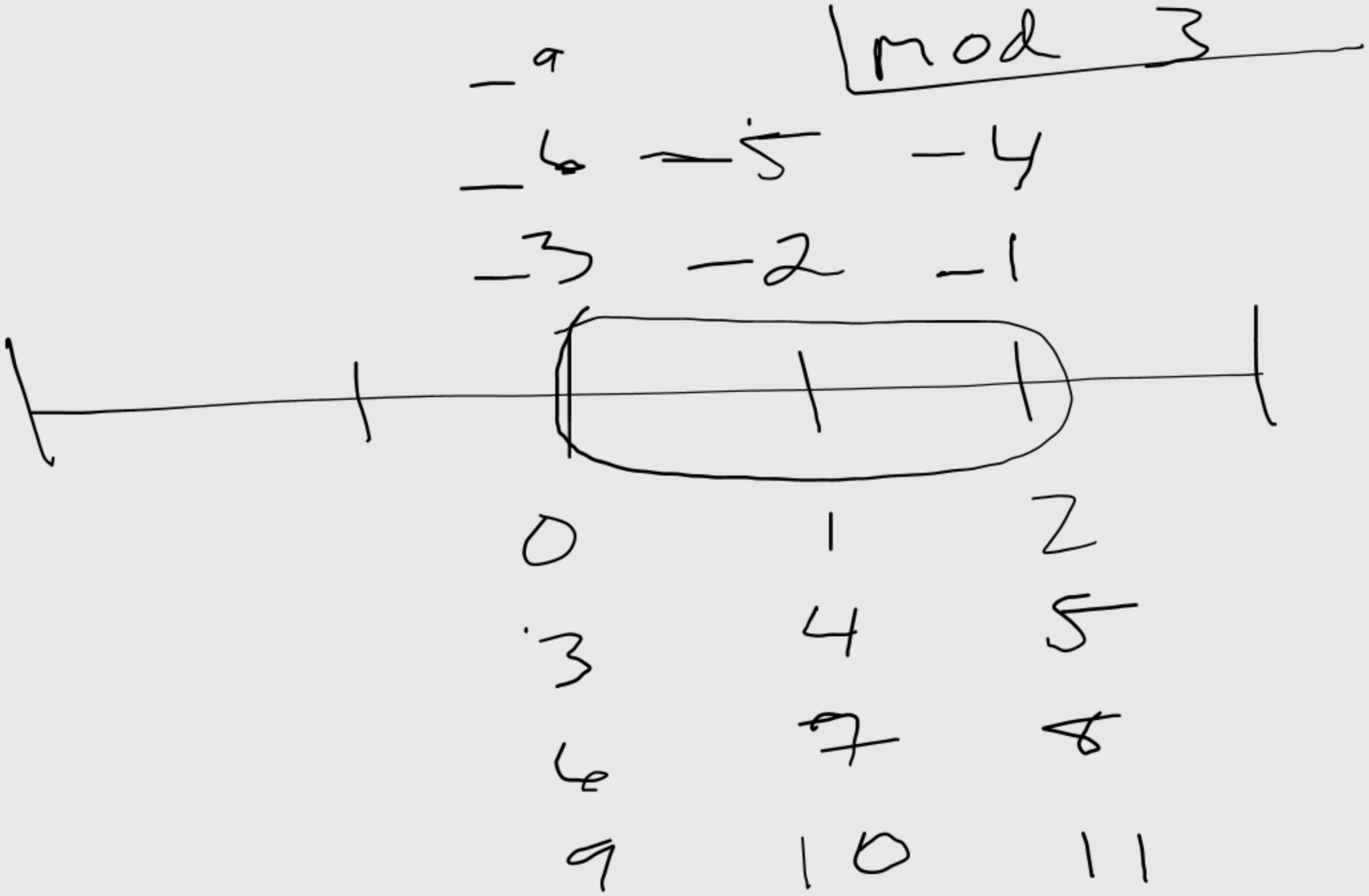
RSA

Public Key K_e: (N, e) Private Key K_d: (N, d)

$N = pq$



must remain secret!


mod 3
-6 -5 -4
-3 -2 -1
0 1 2
3 4 5
6 7 8
9 10 11

An Example

$p = 5$

$q = 7$

$N = 35$

$(p-1)(q-1) = \phi(N) = 24$

$e = 5$

$d = 5$

message: 2

$E_{ke} = 2^5 \bmod N = 32$

ciphertext: 32

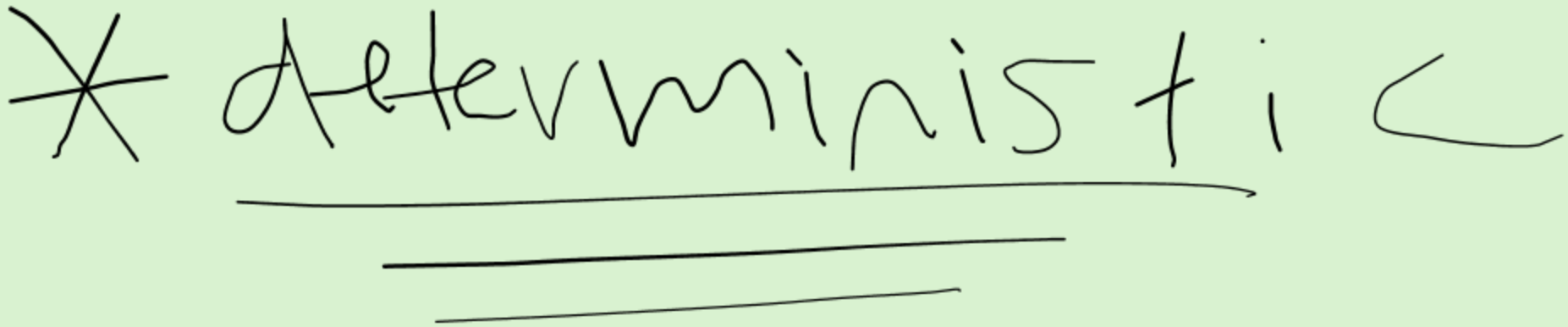
$E_{kd} = 32^5 \bmod N = 2$

- $\frac{n(n-1)}{2}$ keys
- key distribution
+ fast algorithms

Symmetric Encryption

+ $2n$ keys
+ publish public key
- slow algorithms

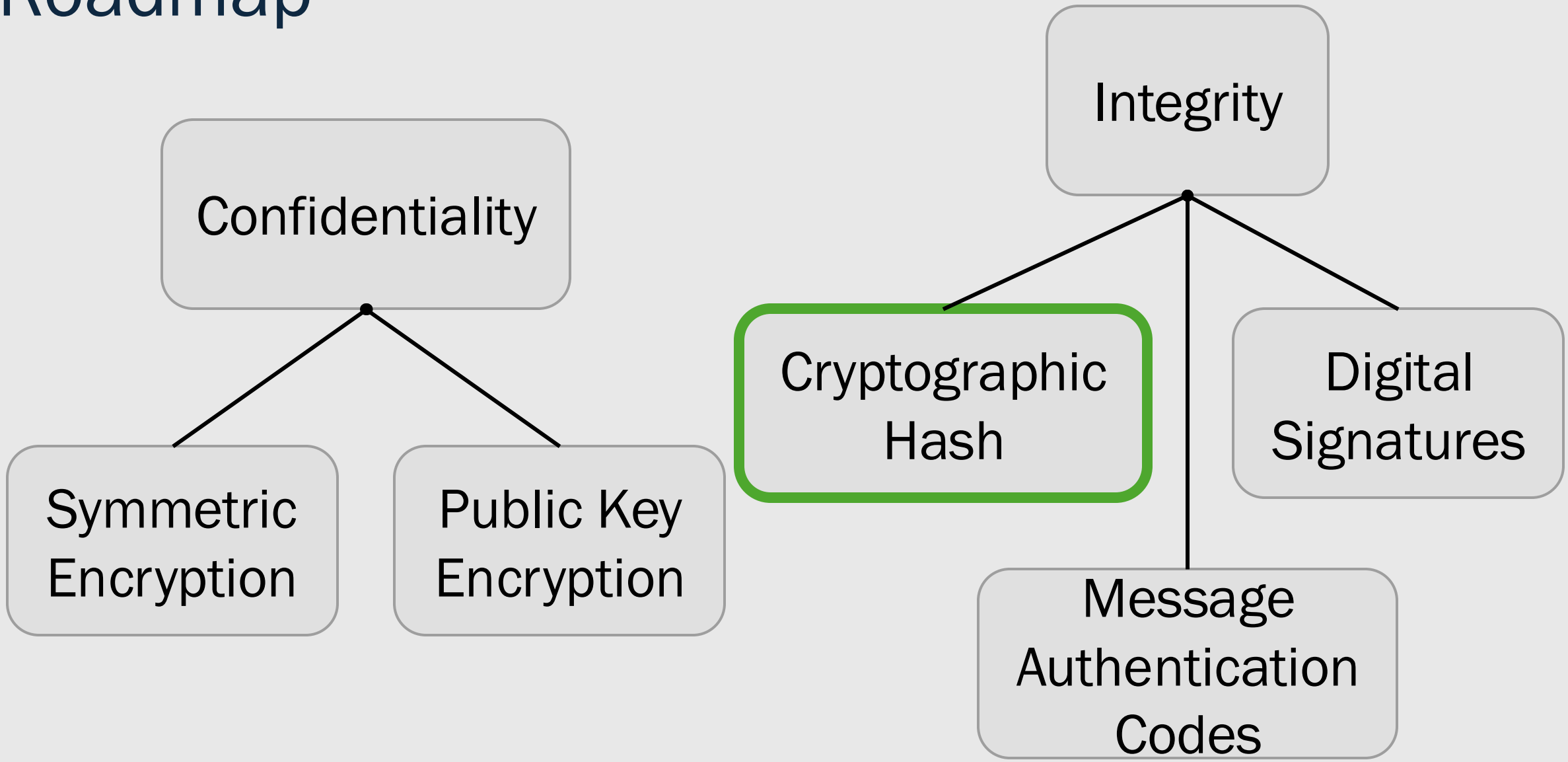
Public Key Encryption



WS Q1

CRYPTOGRAPHIC HASH FUNCTIONS

Roadmap


Integrity
Confidentiality
Cryptographic Hash
Digital Signatures
Symmetric Encryption
Public Key Encryption
Message Authentication Codes

Message Integrity

Def’n: message has not been modified

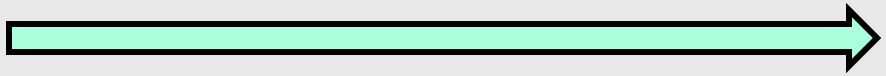
Alice

Bob

Dear Bob,
Do not trust Mallory!
Sincerely,
Alice

msg

msg



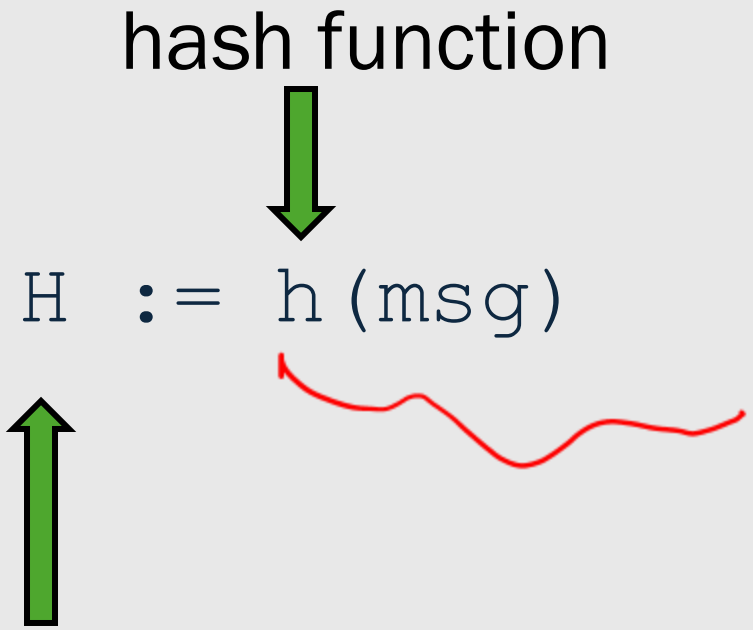
Dear Bob,
You can trust Mallory.
Sincerely,
Alice

msg

Goal: Bob would like to know that the message has been modified.

Tool: cryptographic hash functions!

Terminology



hash value,
hash,
message digest,
digital fingerprint

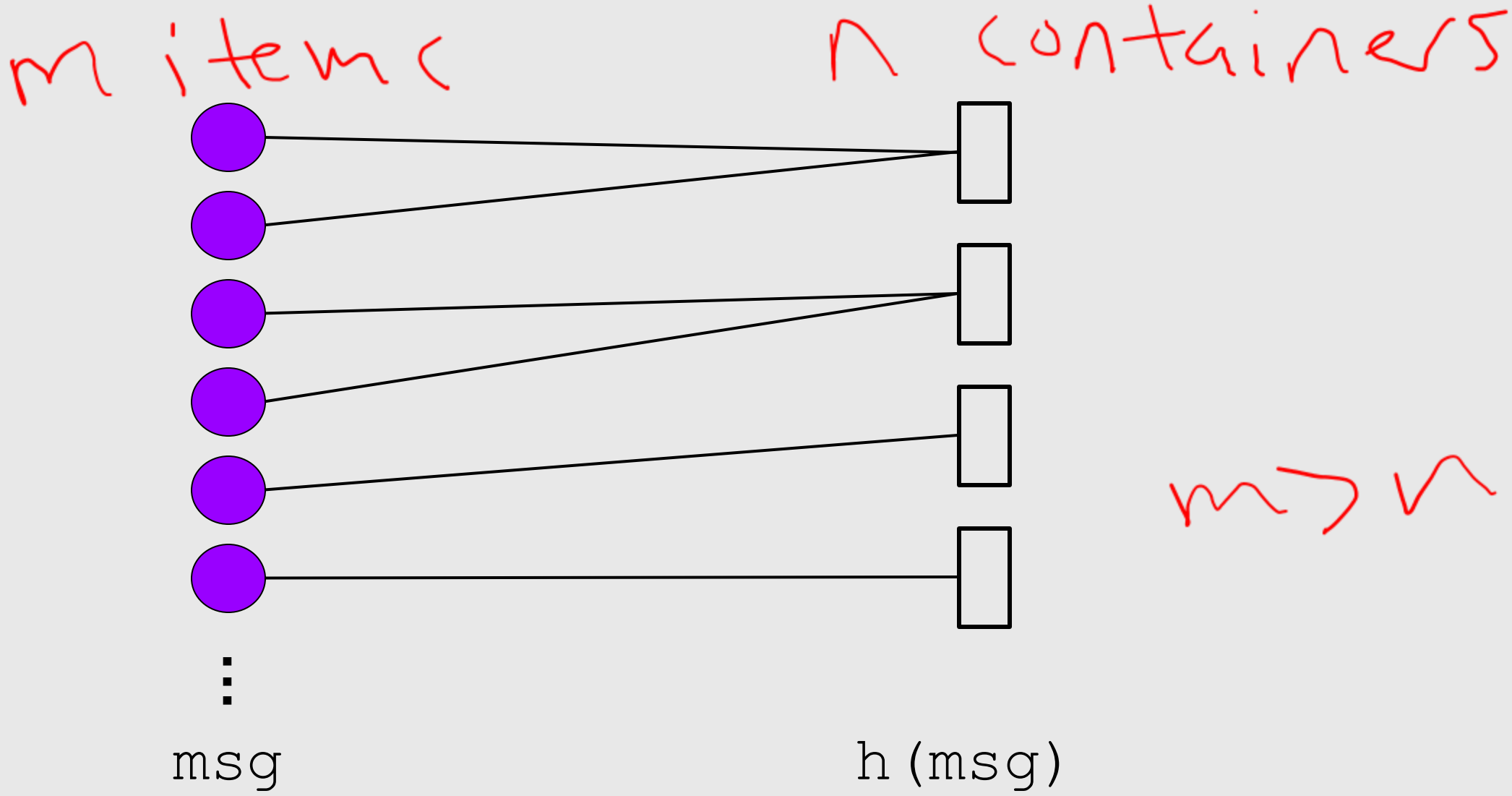
Hash Function

Def'n: a function that takes an arbitrary-length input and produces a fixed-length output

E.g., Modulo operation

3 % 10 = 3
3590224723940253 % 10 = 3

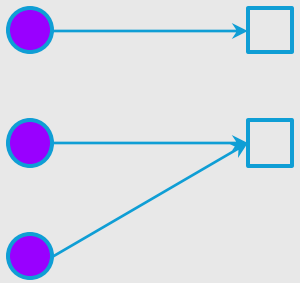
Pigeonhole Principle

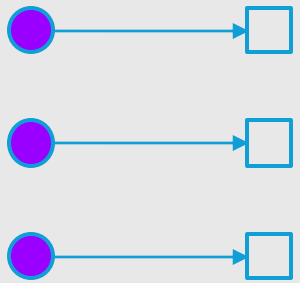


Functions

surjective

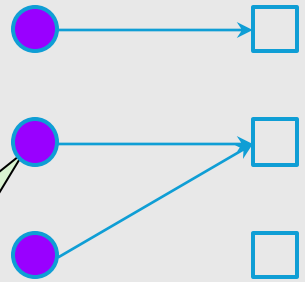
bijective

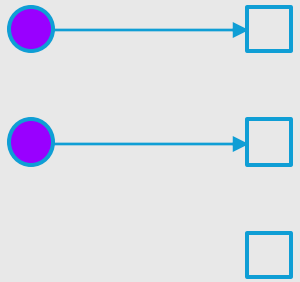




¬injective

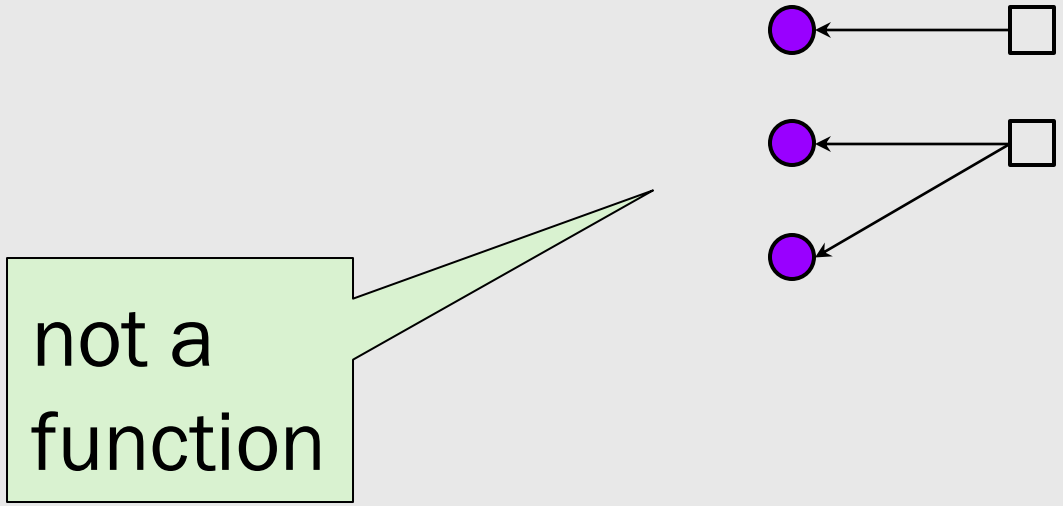
injective





general

¬surjective



Hash functions not invertible!

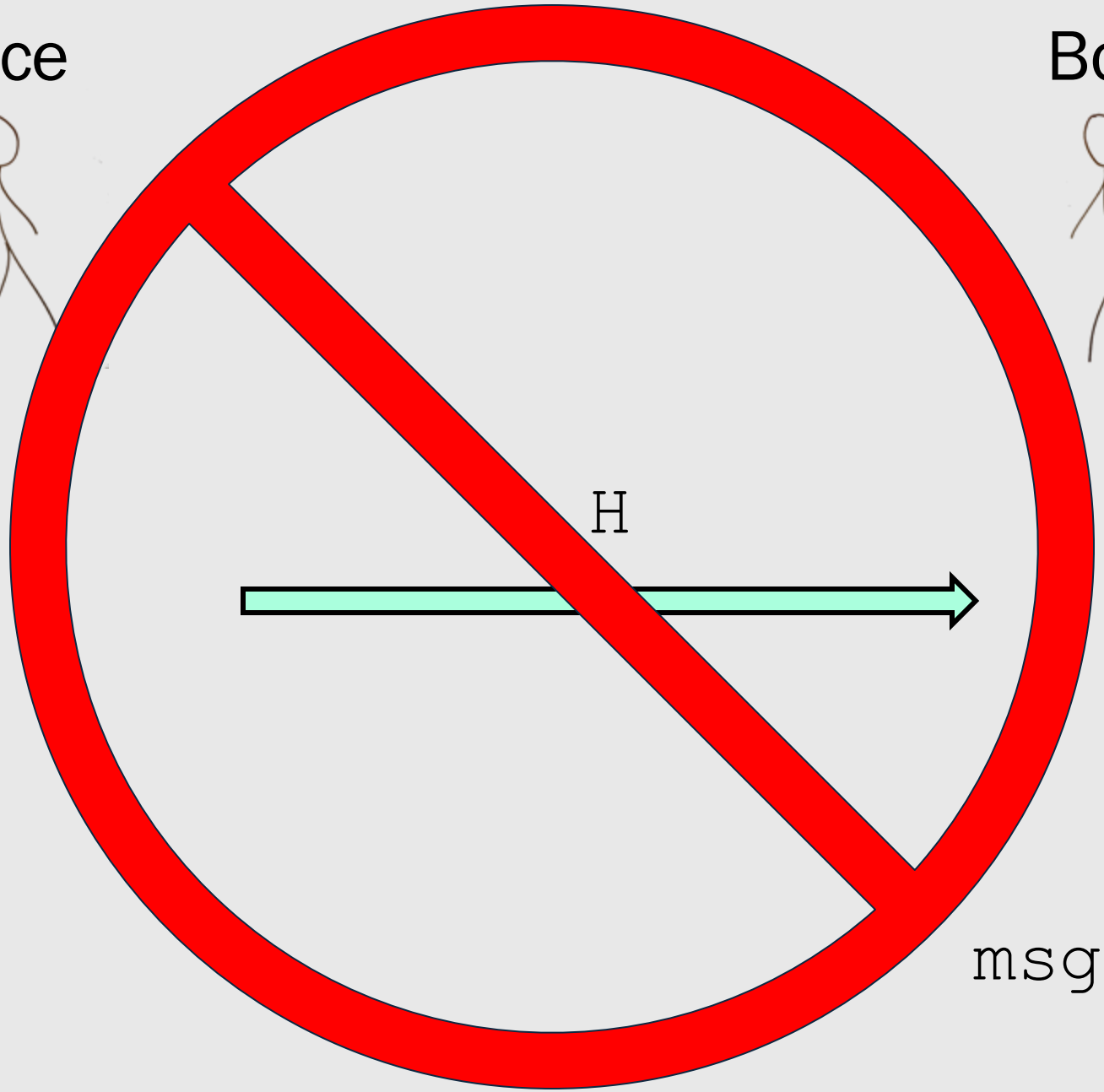
One input mapping to two outputs.

Alice

Bob

Dear Bob,
Do not trust Mallory!
Sincerely,
Alice

msg



Does not exist!

$msg := h^{-1}(H)$

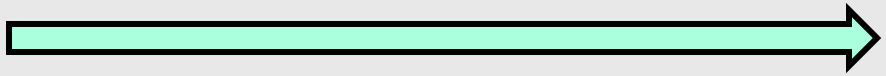
Alice

Bob

Dear Bob,
Do not trust Mallory!
Sincerely,
Alice

msg

(msg, H)



$H \overset{?}{=} h(msg)$

Alice

Bob

Dear Bob,
Do not trust Mallory!
Sincerely,
Alice

msg

(msg, H)

h ≜ msg % 256

Many collisons! ☹

$H \overset{?}{=} h(msg)$

*** A hash can reveal modifications, but cannot guarantee that no modifications exist! ***

CRYPTOGRAPHIC HASH FUNCTIONS

Cryptographic Hash Functions

Def'n: a hash function that is strong enough to resist collisions

- collision resistant
  - *should be hard to find **any two different** inputs that hash to the same output*
- second pre-image resistant
  - *can't find a different input with same hash*
- pre-image resistant:
  - *can't go backwards*

Collision Resistant

Hard to find `msg`, `msg'` such that

`msg ≠ msg'` and

`h(msg) = h(msg')`