COMP435: *SECURITY CONCEPTS!*

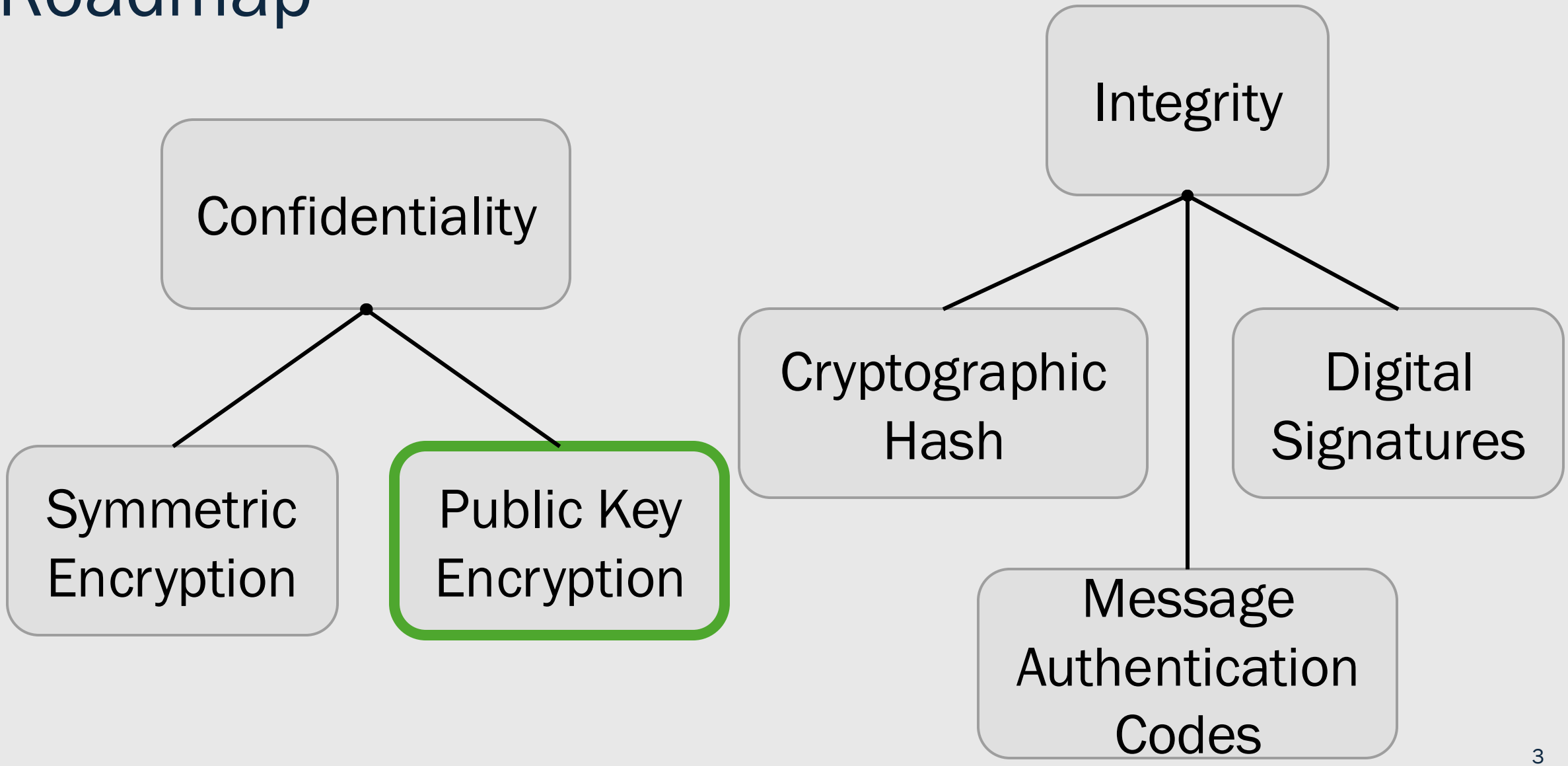
Lecture 9: Cryptographic Hash Functions

tinyurl.com/comp435-fa25

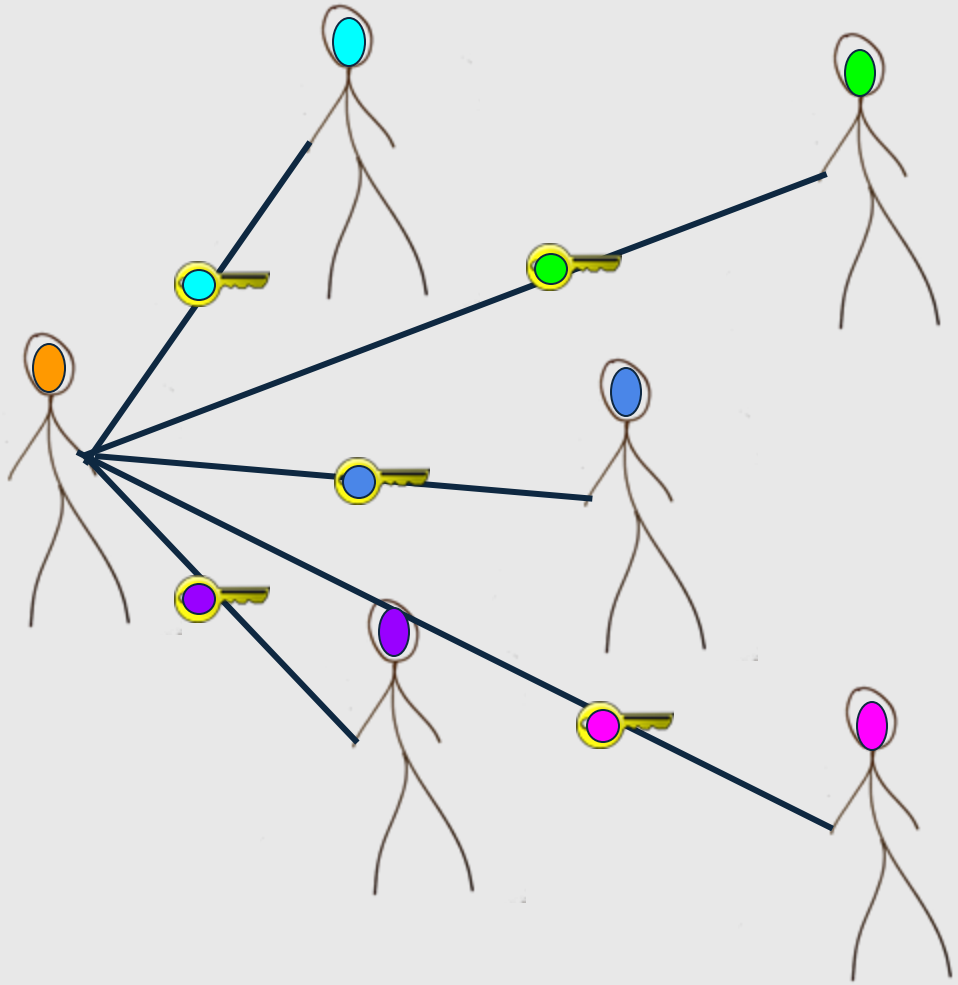
Logistics!

- Lab due tonight!
- Written Assignment due later this week
- Quiz on Wednesday. Review session tomorrow. 5pm in FB007
- Quiz Topics
  - *Passwords*
  - *Guessing Passwords, Keys, Lab 1*
  - *One-Time Pad Properties*
  - *Different Simple Ciphers (Mono/Poly Alphabetic, Caesar)*
  - *Block Ciphers, Modes of Operation – Encryption/Decryption*
  - *Biometrics, Templates & Reading Probability Density Curves*
  - *Symmetric vs. Public Key Cryptography*
  - *RSA: basic modular arithmetic*

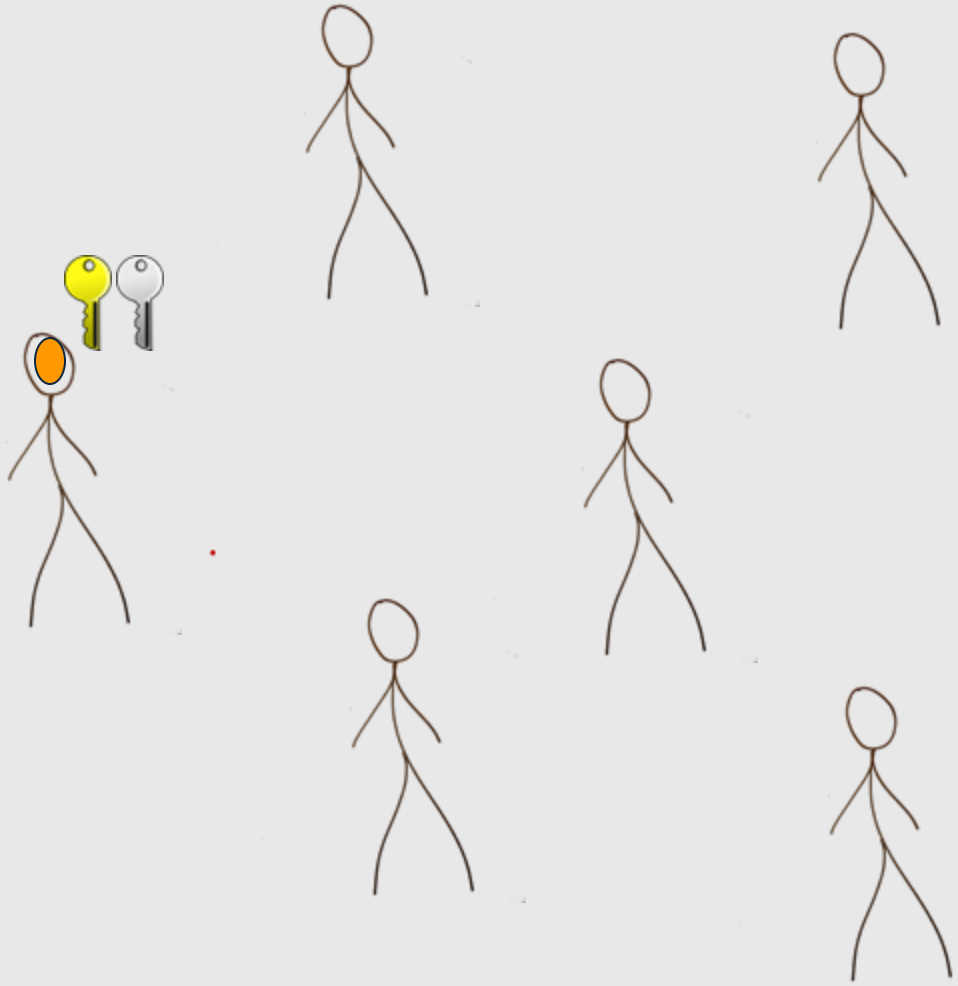
Roadmap


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

Symmetric Encryption



Public Key Encryption



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 Security

- $c = msg^e \bmod N$
- The attacker's challenge:
  Given N, e, msg^e, find msg
- If the attacker can factor N, they win

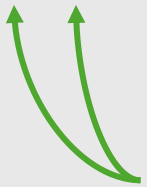
Given e and c, it is difficult to compute msg

Given e, c, and msg, it is difficult to compute d

RSA

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

$N = pq$

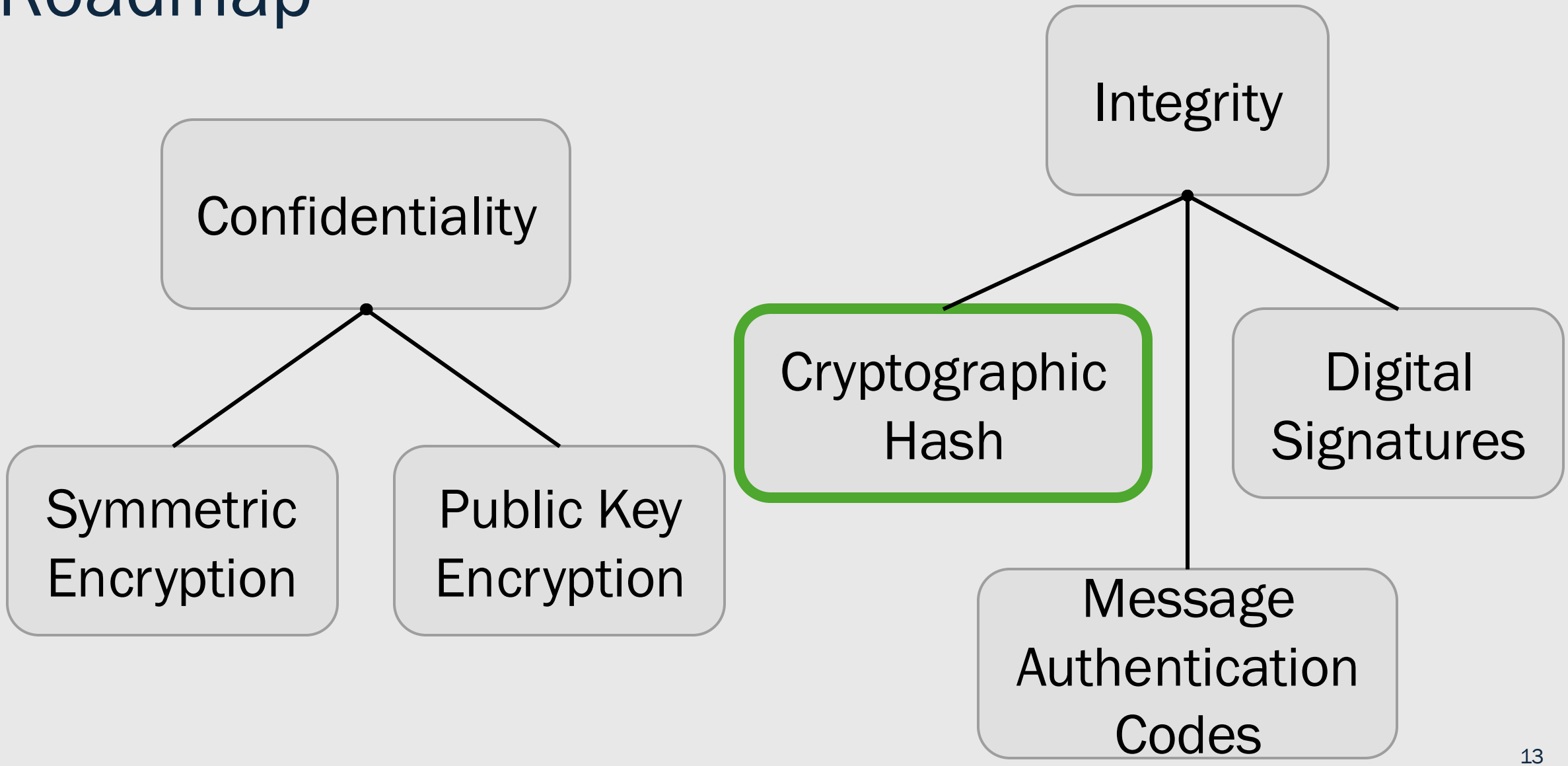


must remain secret!

RSA Practice

CRYPTOGRAPHIC HASH FUNCTIONS

Roadmap


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

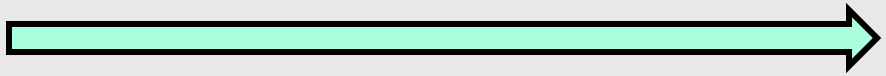
Alice

Bob

Dear Bob,
Do not trust Mallory!
Sincerely,
Alice

msg

msg



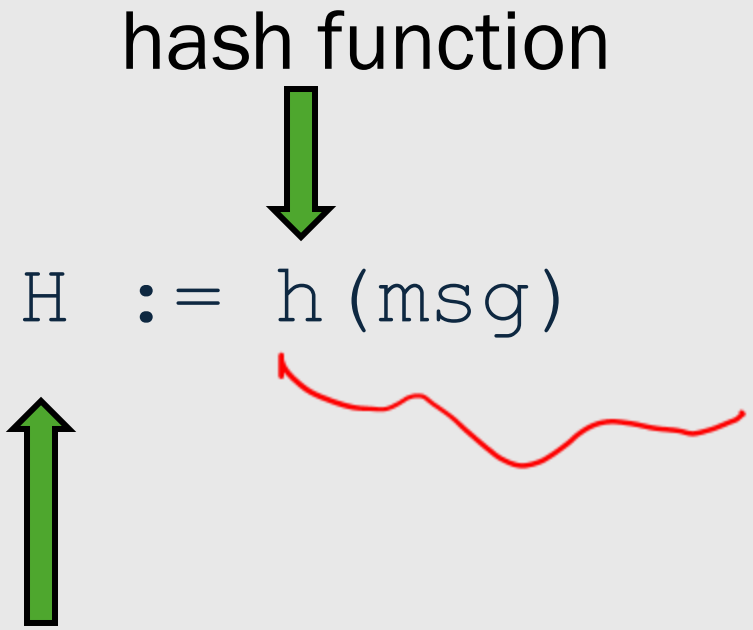
Dear Bob,
You can trust Mallory.
Sincerely,
Alice

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

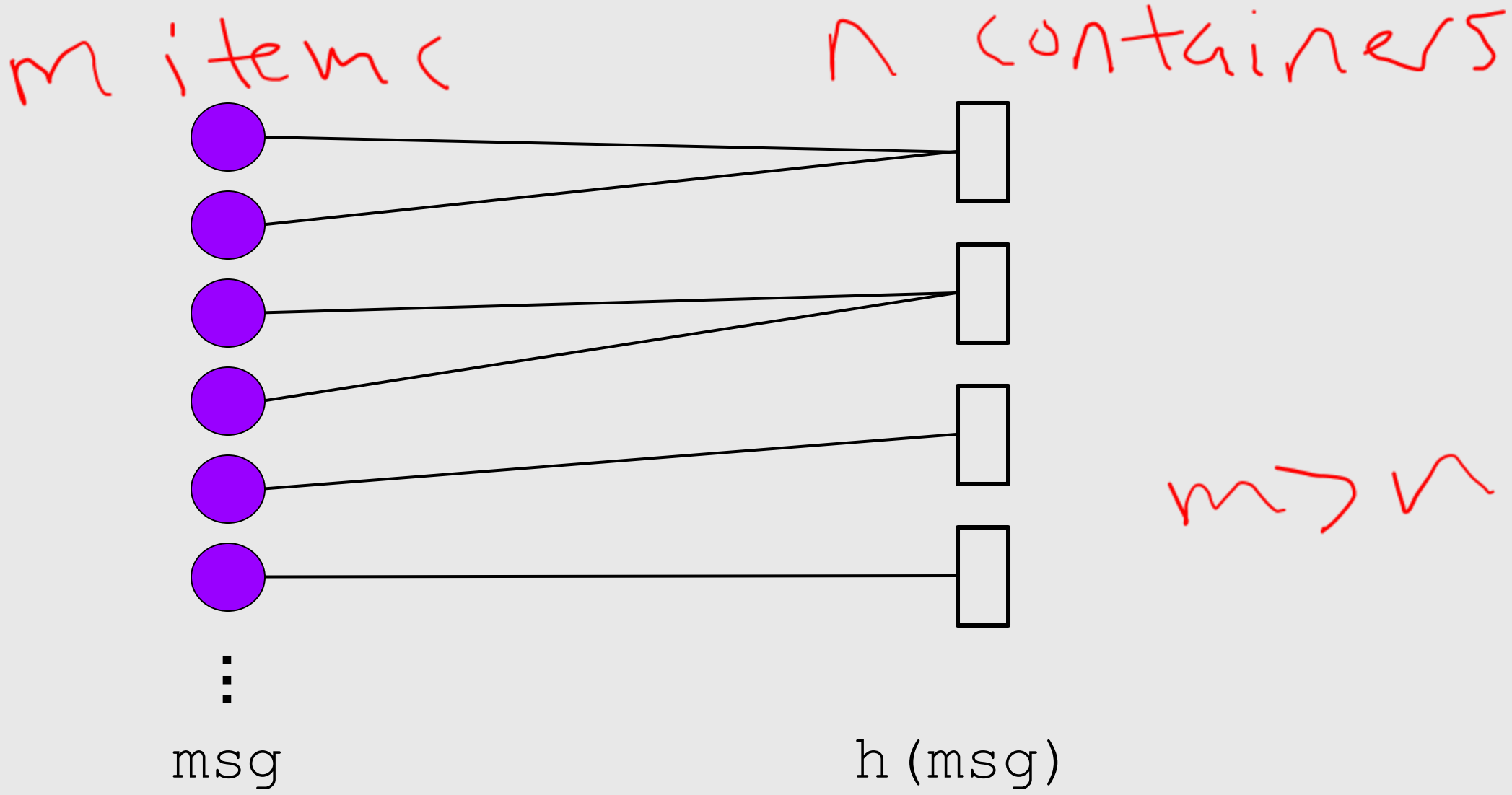
Hash Function

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

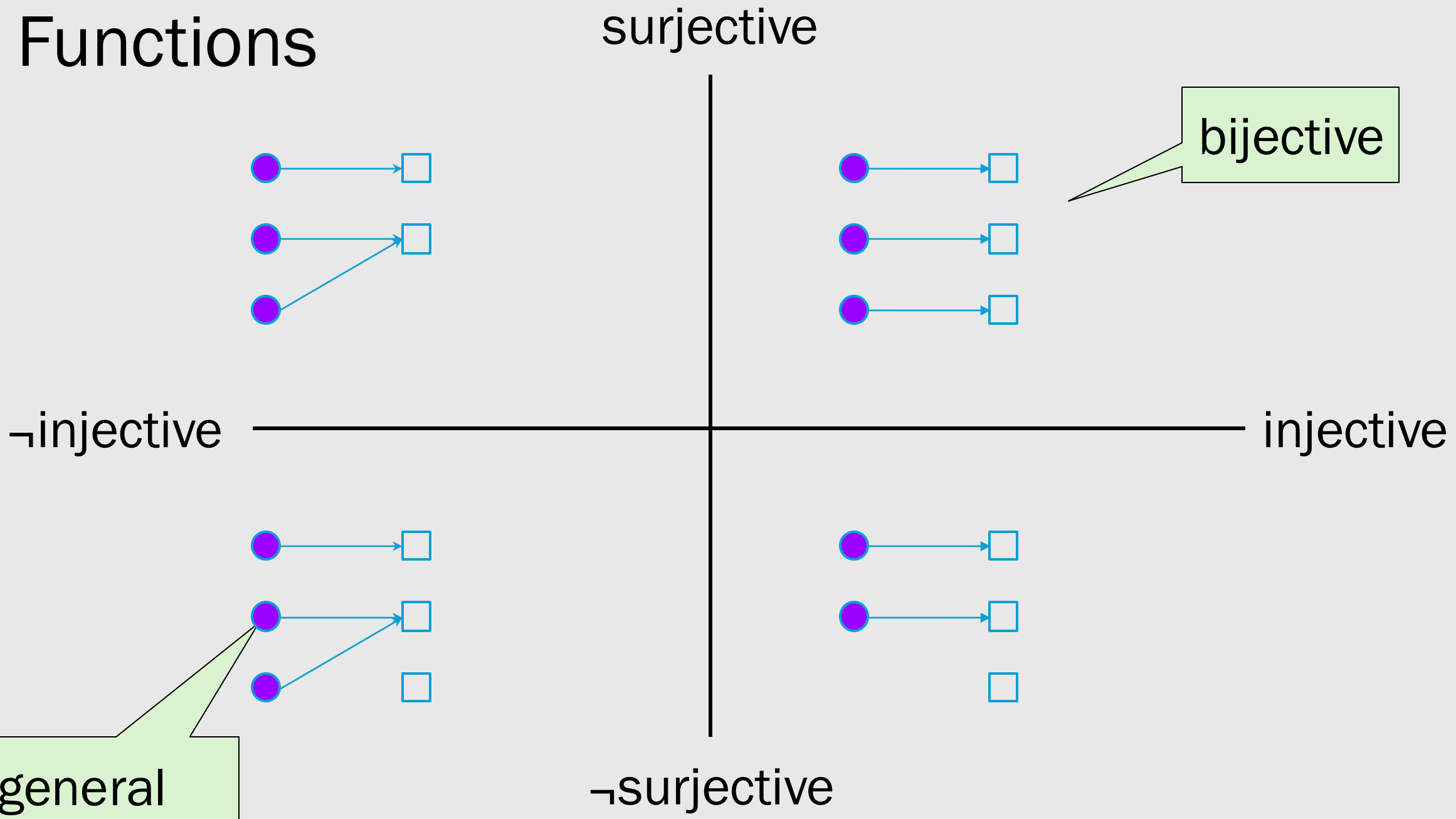
E.g., Modulo operation

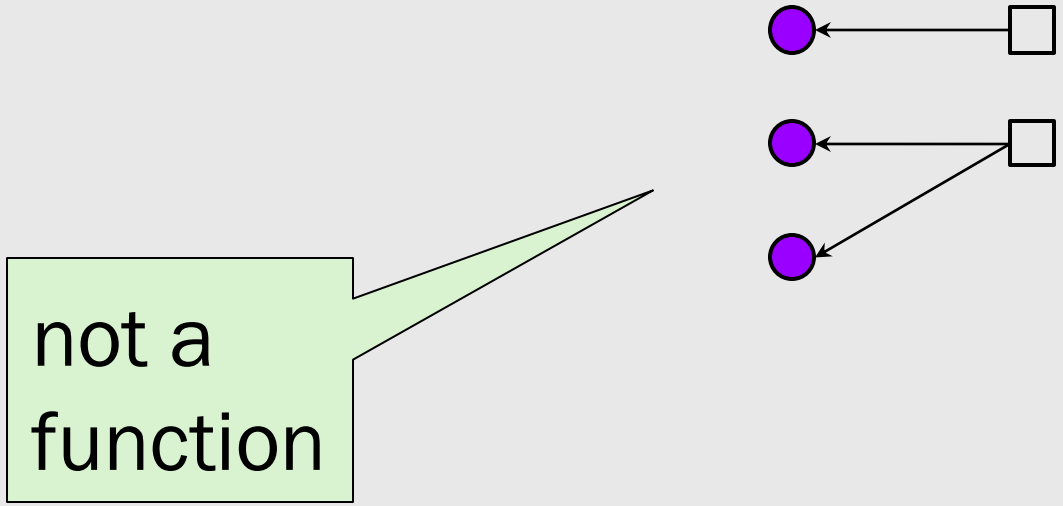
3 % 10 = 3
3590224723940253 % 10 = 3

Pigeonhole Principle



Functions


surjective
bijective
¬injective
injective
general
¬surjective

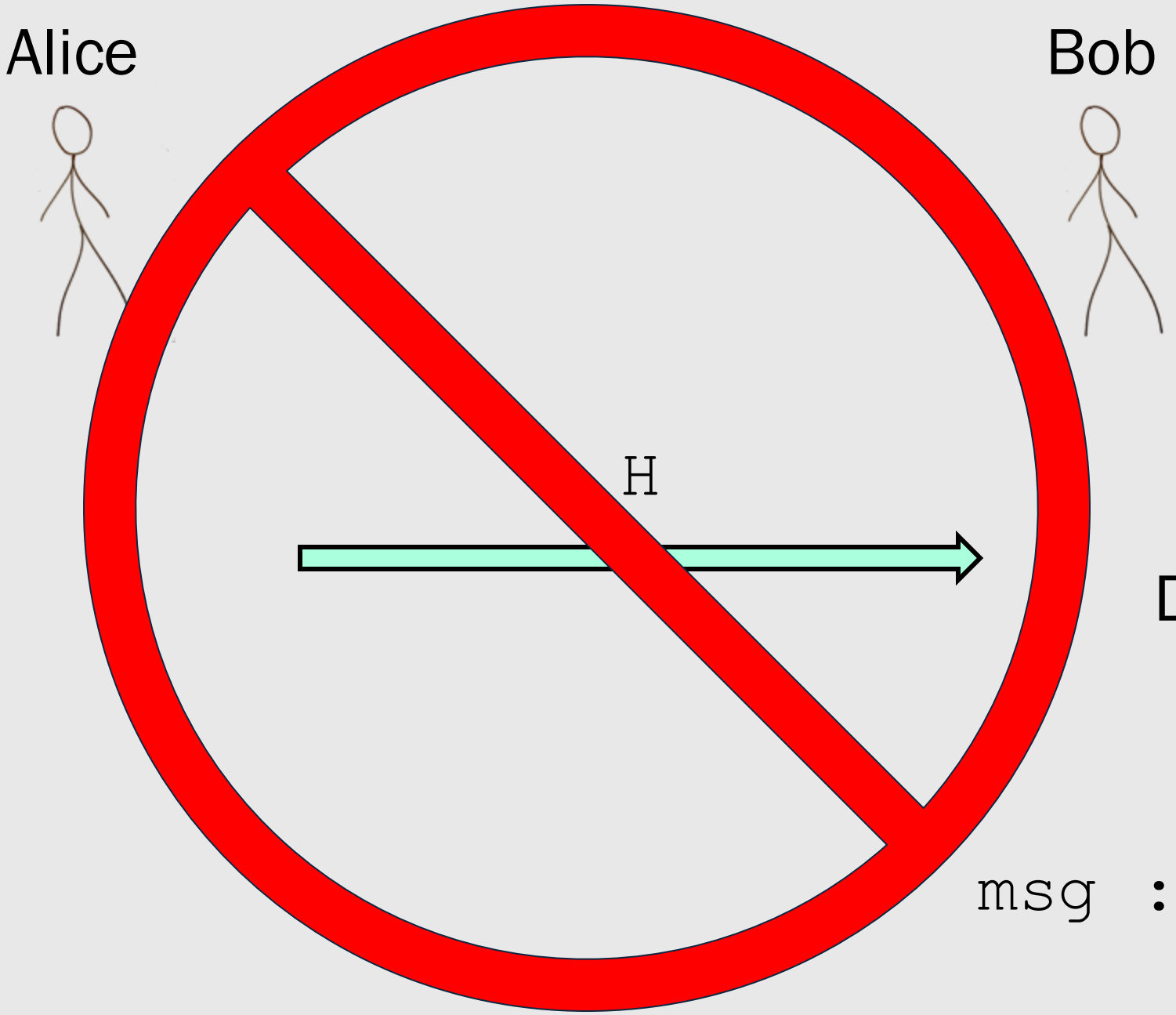


Hash functions not invertible!

One input mapping to two outputs.

Dear Bob,
Do not trust Mallory!
Sincerely,
Alice

msg



Does not exist!

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

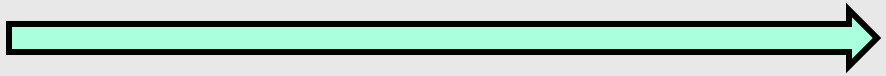
Alice

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')`

Second Pre-image Resistant

Given `msg`, hard to find `msg'` such that
`msg ≠ msg'` and
`h(msg) = h(msg')`

Pre-image Resistant

Let `H := h(msg)`
Given `H`, hard to find `msg'` such that
`H = h(msg')`

Cryptographic Hash Functions

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

- collision resistant
- second pre-image resistant
- pre-image resistant

COLLISION RESISTANT & THE BIRTHDAY PARADOX

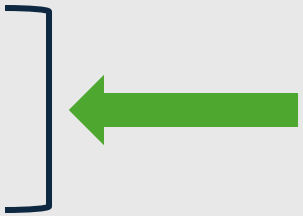
Number of people	Probability of a birthday collision
5	3%
20	41%
50	97%
100	99.99%

Cryptographic Hash Functions

- MD5
- SHA-1
- SHA-2
- SHA-3

Cryptographic Hash Functions

- MD5
- SHA-1
- SHA-2
- SHA-3



deprecated

Cryptographic Hash Functions

- MD5
- SHA-1
- SHA-2
- SHA-3

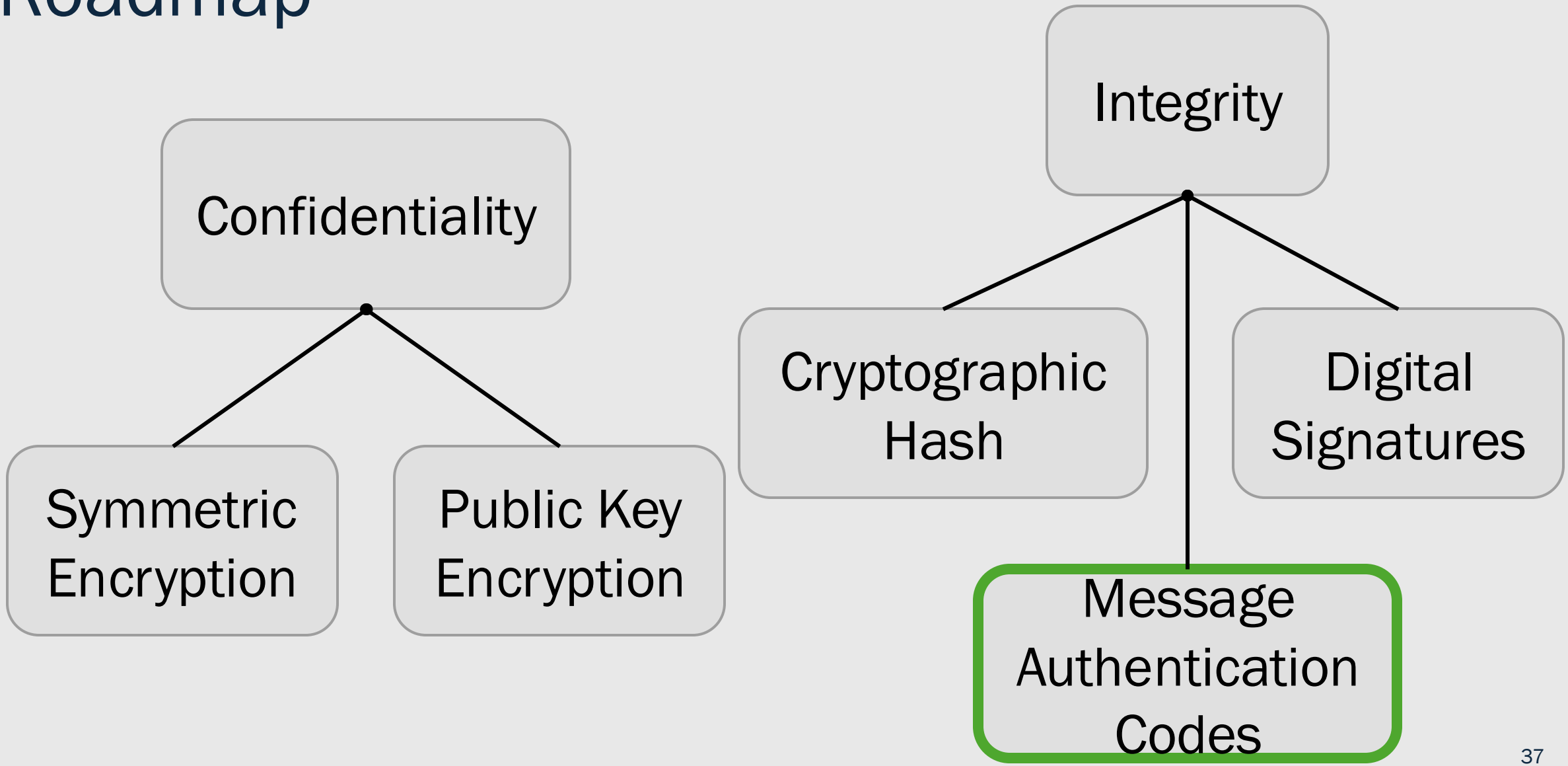
SHA-256

Cryptographic Hash Functions

- MD5
- SHA-1
- SHA-2
- SHA-3  Multiple Variants

MESSAGE AUTHENTICATION CODES

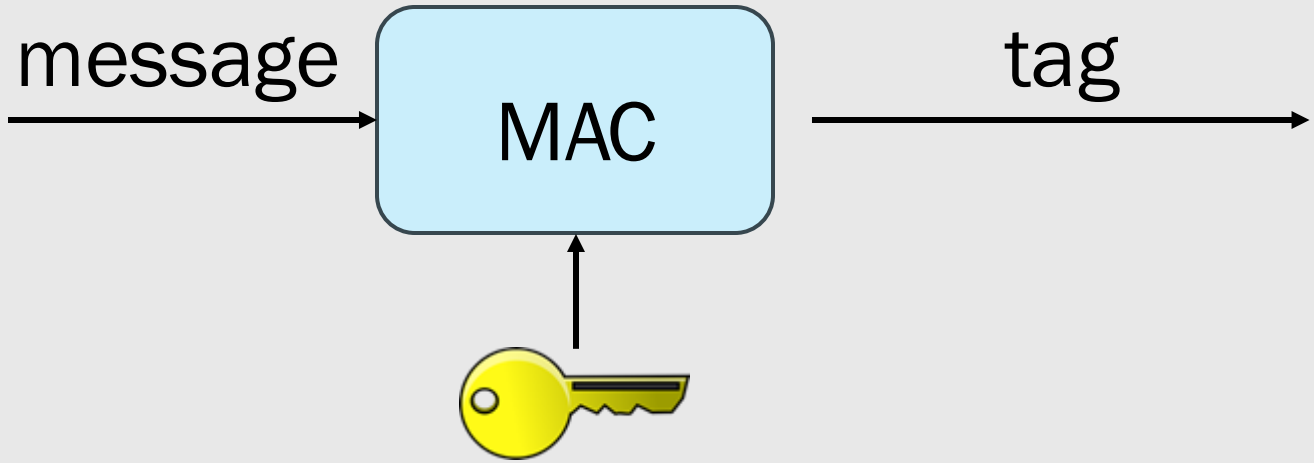
Roadmap

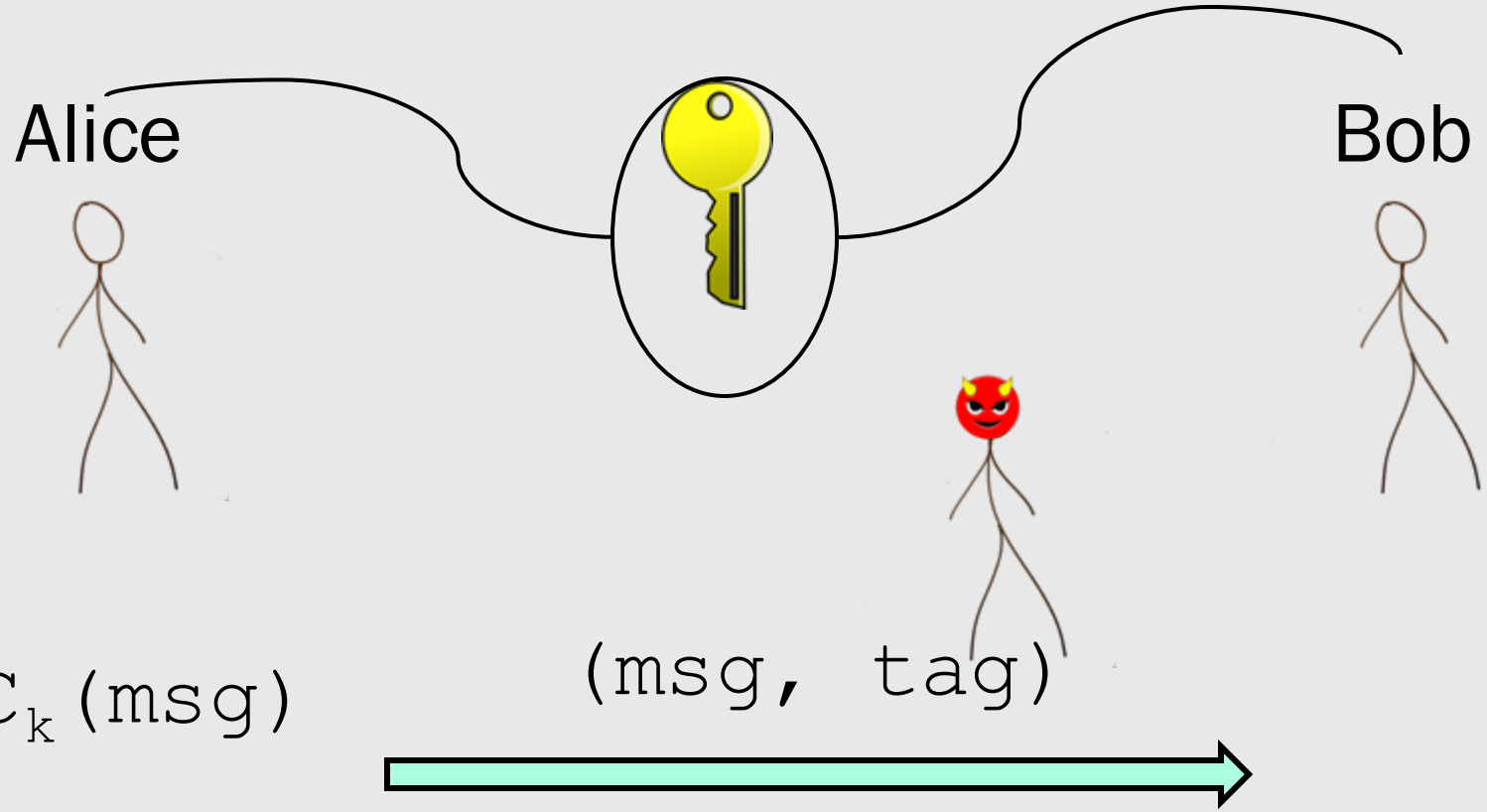

Confidentiality
Symmetric Encryption
Public Key Encryption
Integrity
Cryptographic Hash
Digital Signatures
Message Authentication Codes

Message Authentication Code (MAC)

Def’n: a keyed cryptographic hash function

Message Authentication Codes





Dear Bob,
Do not trust
Mallory!
Sincerely,
Alice

msg

tag:= $MAC_k(msg)$

(msg, tag)

Dear Bob,
You can trust
Mallory.
Sincerely,
Alice

$tag \stackrel{?}{=} MAC_k(msg)$

Properties of MACs

- Unforgeability
- Integrity

Properties of MACs

- **Unforgeability**  *Authentication of data origin*
- Integrity

Properties of MACs

- Unforgeability
- **Integrity**  *Integrity of Data*

Message Authentication Codes

- HMAC
- CBC-MAC
- CMAC

Message Authentication Codes

- **HMAC** 
- CBC-MAC
- CMAC

a construction that keys an underlying collision-resistant cryptographic hash function

Message Authentication Codes

- HMAC
- **CBC-MAC**
- **CMAC**

combines a block cipher with a secret key

MESSAGE AUTHENTICATION & CONFIDENTIALITY

Message Authentication and Confidentiality

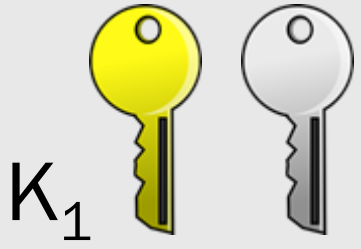
Encrypt-and-Authenticate

Authenticate-then-Encrypt

Encrypt-then-Authenticate

Encrypt and Authenticate

Alice



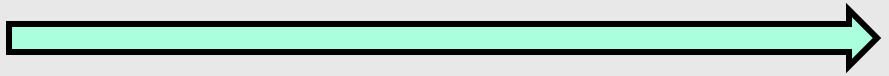
Bob

msg

$c := Enc_{k1}(msg)$

$t := MAC_{k2}(msg)$

(c,t)

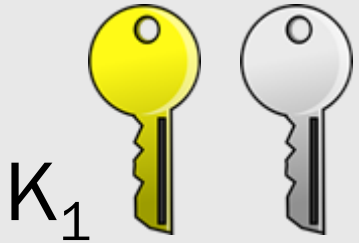


$msg := Dec_{k1}(c)$

$t \stackrel{?}{=} MAC_{k2}(msg)$

Authenticate-then-Encrypt

Alice



Bob

msg

$t := MAC_{k2}(msg)$

$c := Enc_{k1}(msg|t)$

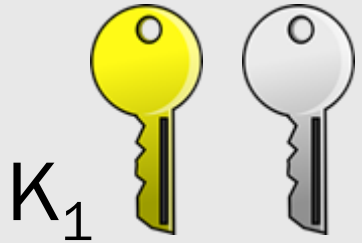
(c)

$msg|t := Dec_{k1}(c)$

$t \stackrel{?}{=} MAC_{k2}(msg)$

Encrypt-then-Authenticate

Alice



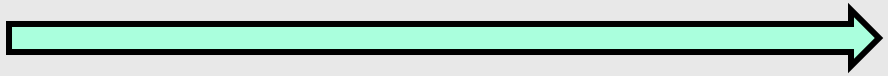
Bob

msg

$c := Enc_{k1}(msg)$

$t := MAC_{k2}(c)$

(c,t)



?

$t = MAC_{k2}(c)$

$msg := Dec_{k1}(c)$

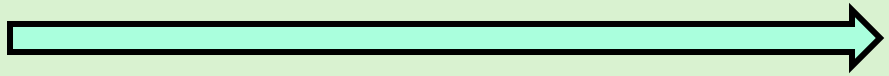
Alice

Bob

msg

H:= h(msg)

(msg, H)



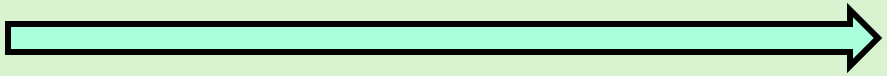
H ?= h(msg)

---

msg

tag:= MAC_k(msg)

(msg, tag)



tag ?= MAC_k(msg)

Alice

Bob

MAC with a shared key: gives **integrity + authenticity !**

msg

H:= h(msg)

? = h(msg)

msg

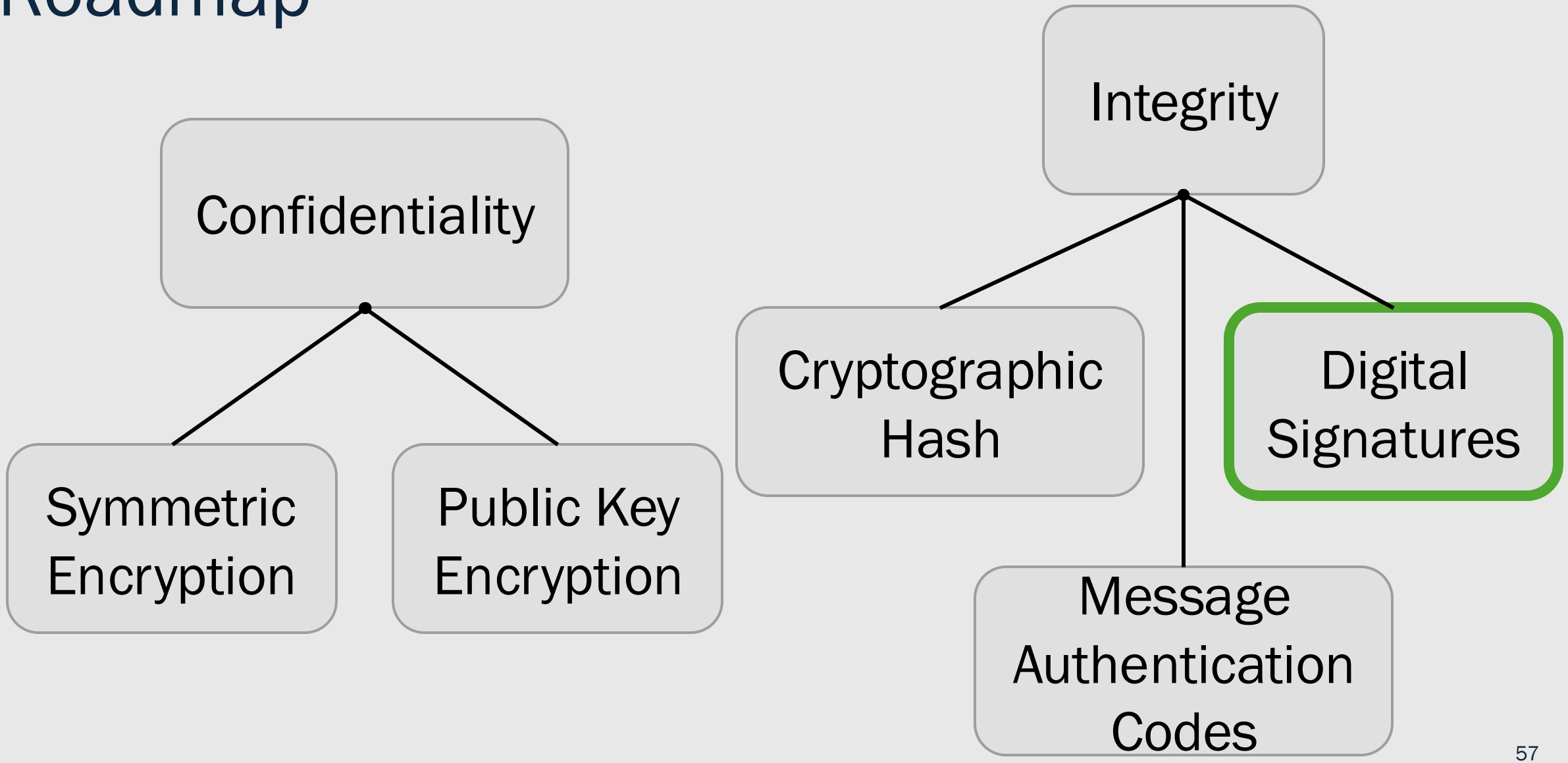
tag:= MAC_k(msg)

(msg, tag)

? tag = MAC_k(msg)

DIGITAL SIGNATURES

Roadmap


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

Digital Signature

Def’n: a means to ensure a given message comes from a given source.

Properties of a Digital Signature

- Authentication
- Integrity
- Accountability

Properties of a Digital Signature

- **Authentication**  Authentication of data origin
- Integrity
- Accountability

Properties of a Digital Signature

- Authentication
- **Integrity**  Integrity of data
- Accountability

Properties of a Digital Signature

- Authentication
- Integrity
- **Accountability**

Non-repudiation. A signer cannot deny having signed.

Digital Signature Requirements

- Unforgeable
- Authentic
- Unalterable
- Not reusable

Digital Signature Requirements

- **Unforgeable** 
- Authentic
- Unalterable
- Not reusable

No party can produce a viable signature by outside means

Digital Signature Requirements

- Unforgeable
- **Authentic**
- Unalterable
- Not reusable

A signature is tied to a single party

Digital Signature Requirements

- Unforgeable
- Authentic
- **Unalterable**
- Not reusable

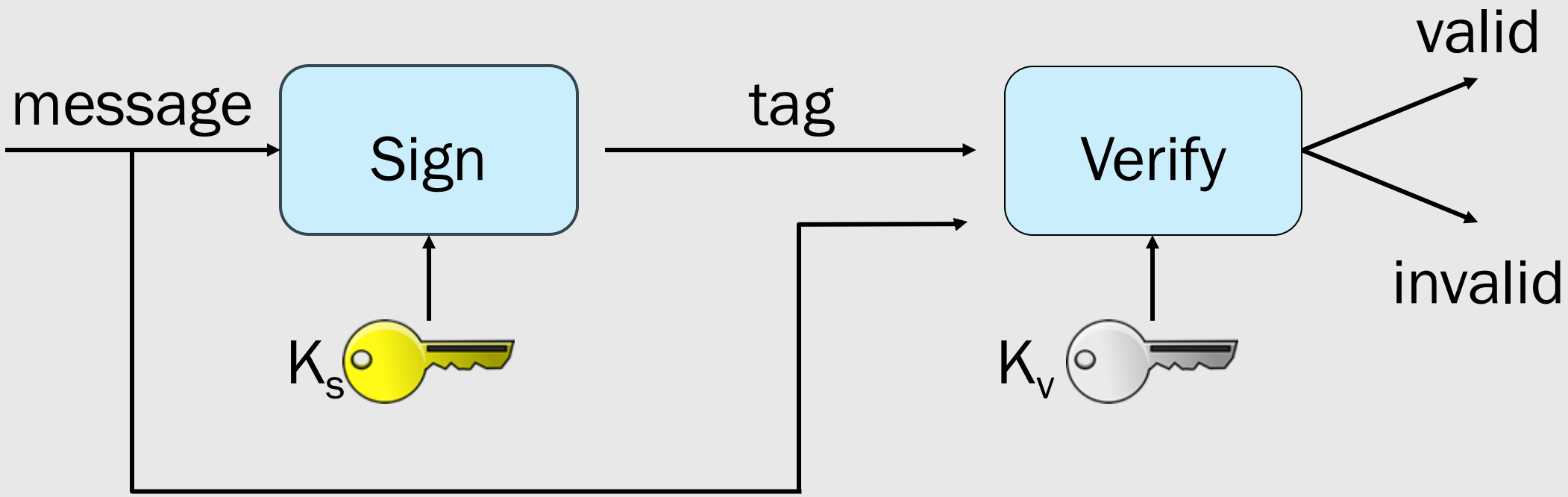
A signed message cannot be altered without invalidating the signature

Digital Signature Requirements

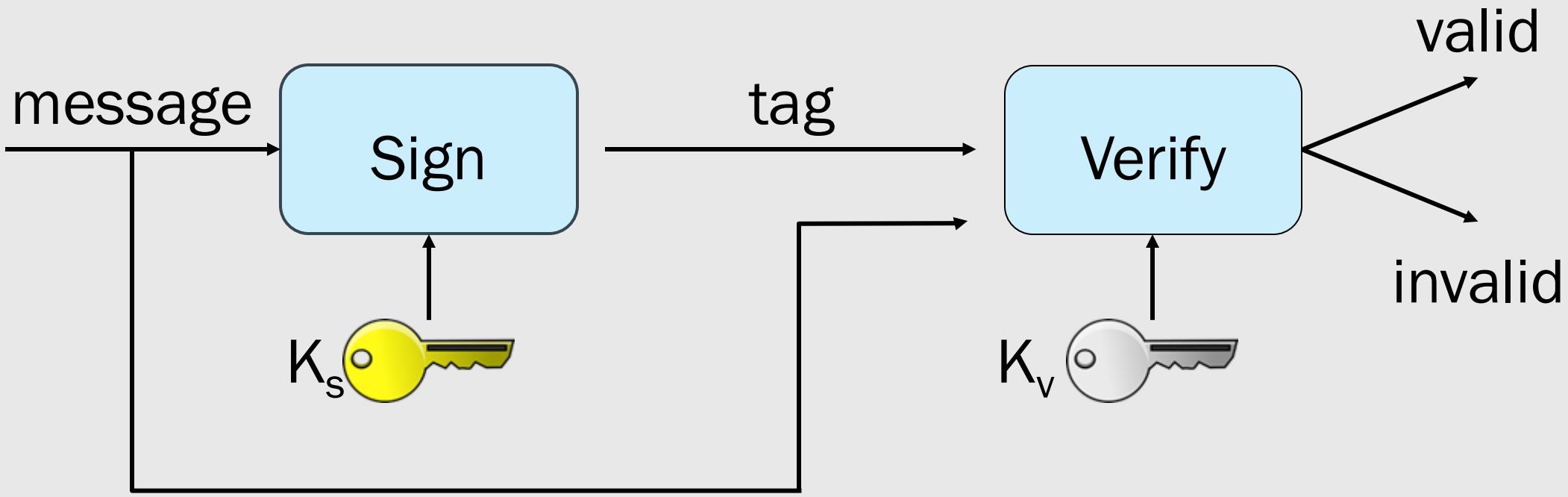
- Unforgeable
- Authentic
- Unalterable
- **Not reusable**

A signature is tied to the signed message

Public Key Signatures



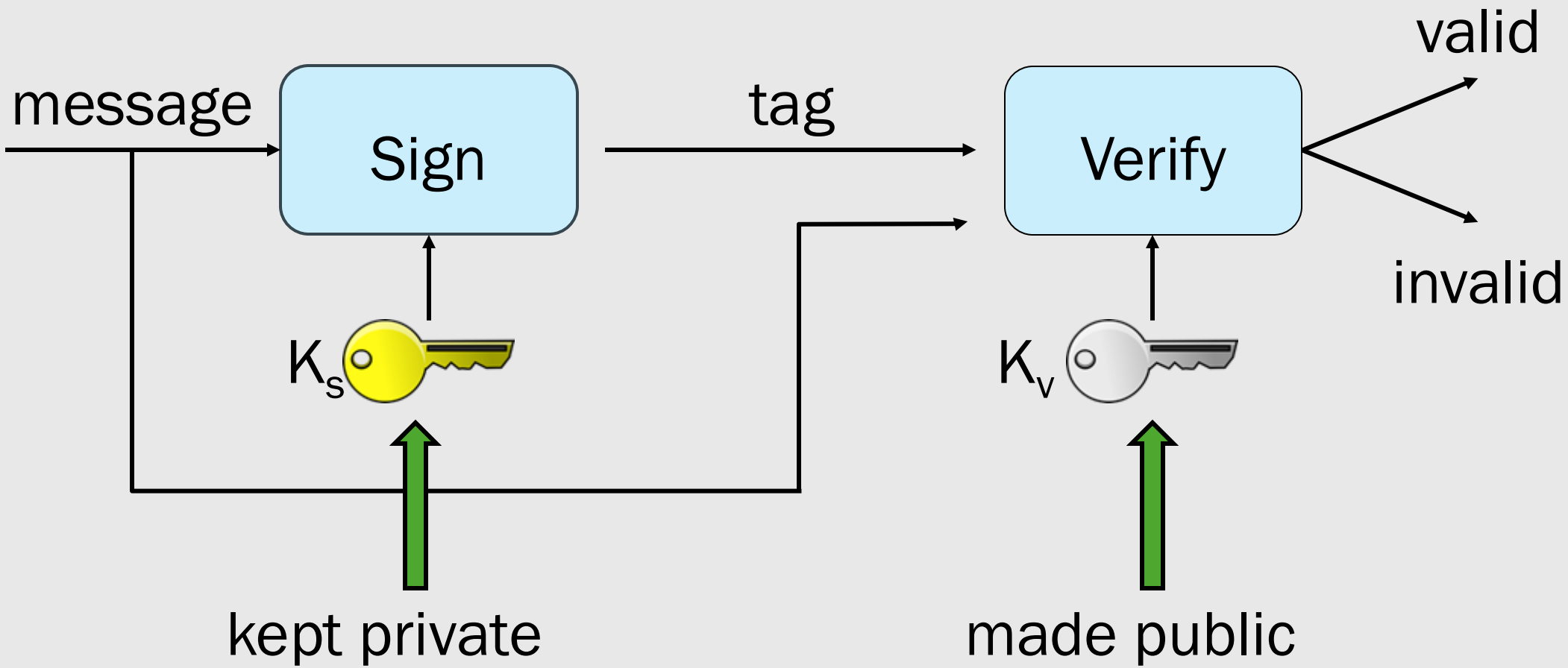
Public Key Signatures

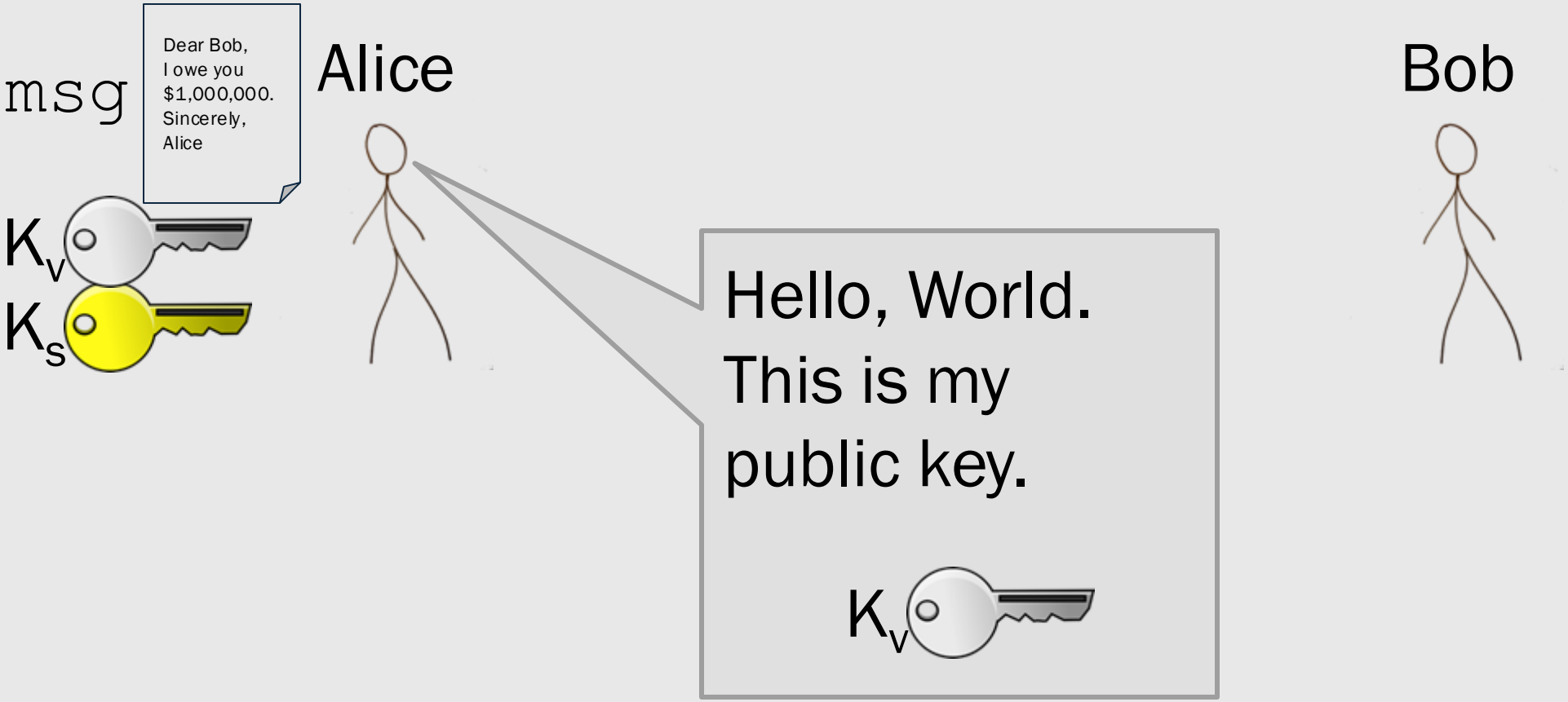


$$tag = Sign_{Ks}(msg)$$

$$isValid = Verify_{Kv}(msg, tag)$$

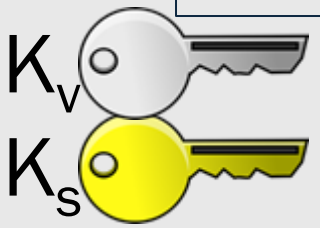
Public Key Signatures




msg
Dear Bob,
I owe you
$1,000,000.
Sincerely,
Alice
Alice
Bob
K_v
K_s
Hello, World.
This is my
public key.
K_v

msg

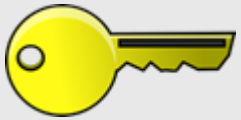
Dear Bob,
I owe you
$1,000,000.
Sincerely,
Alice



Alice

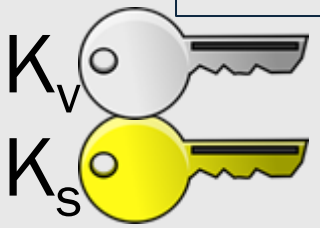
Bob

$tag := Sign_{Ks}(msg)$



msg

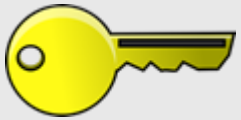
Dear Bob,
I owe you
$1,000,000.
Sincerely,
Alice



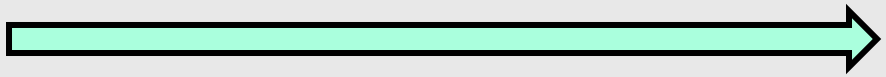
Alice

Bob

$tag := Sign_{Ks}(msg)$



(tag, msg)

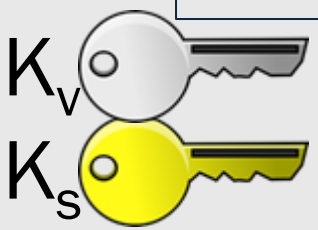


msg

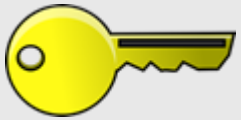
Dear Bob,
I owe you
$1,000,000.
Sincerely,
Alice

Alice

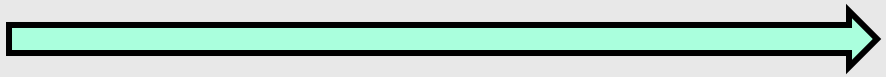
Bob



$tag := Sign_{Ks}(msg)$



(tag, msg)



$isValid = Verify_{Kv}(msg, tag)$

RSA FOR DIGITAL SIGNATURES

RSA for digital signatures

sign: $sig = m^s \bmod N$

verify: $m \overset{?}{=} sig^v \bmod N$

RSA for digital signatures

sign: $sig = m^s \bmod N$

verify: $m \overset{?}{=} sig^v \bmod N$

(N, v) → K_v

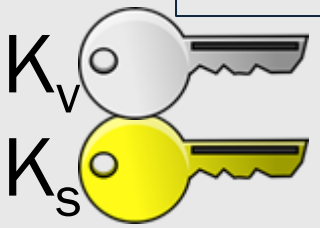
(N, s) → K_s

```
sig := Rand(N)
msg := sig^v mod N
```

RSA does not provide unforgeability

msg

Dear Bob,
I owe you
$1,000,000.
Sincerely,
Alice

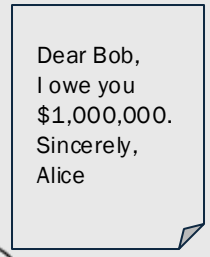


Alice

Hello, World.
This is my
public key.

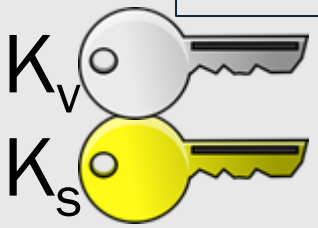
Bob

msg

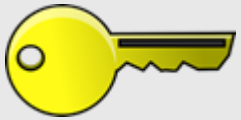


Alice

Bob

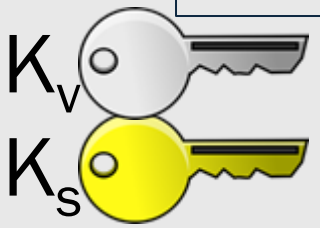


$tag := Sign_{K_s}(h(msg))$



msg

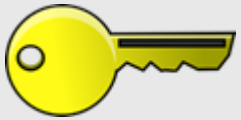
Dear Bob,
I owe you
$1,000,000.
Sincerely,
Alice



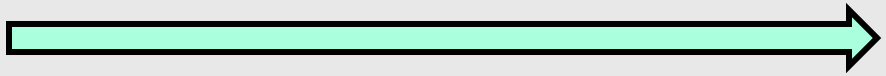
Alice

Bob

$tag := Sign_{Ks}(h(msg))$



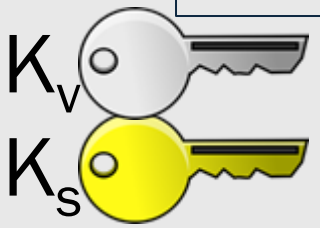
(tag, msg)



msg

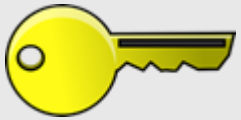
Dear Bob,
I owe you
$1,000,000.
Sincerely,
Alice

Alice

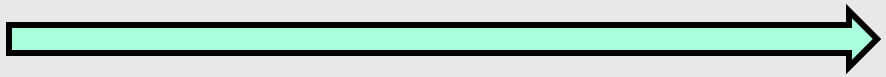


Bob

$tag := Sign_{Ks}(h(msg))$

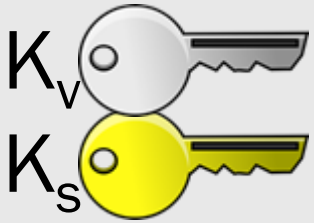


(tag, msg)



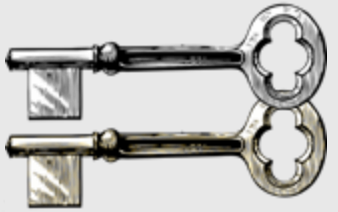
$isValid = Verify_{Kv}(h(msg), tag)$

Encrypt then Sign



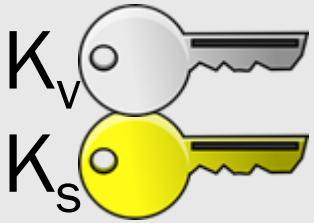
Alice

Bob



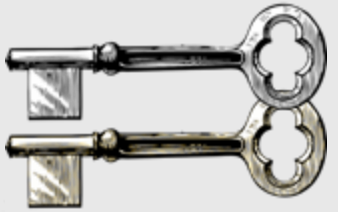
Hello, World. This is my public key for verifying my signature.

K_v 

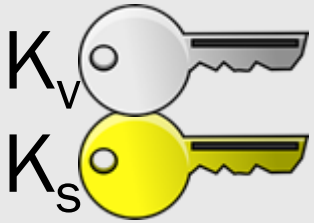


Alice

Bob

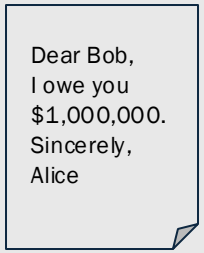


Hello, World.
This is my public key for encrypting.

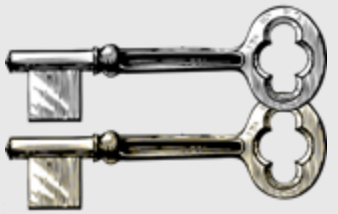


Alice

msg



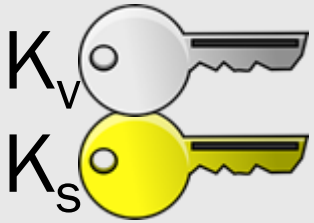
Bob



K_e
K_d

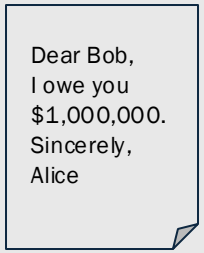
$$c := Enc_{Ke}(msg)$$

$$tag := Sign_{ks}(c)$$

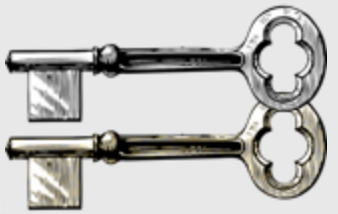


Alice

msg



Bob

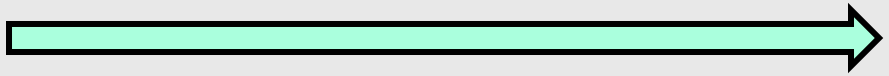


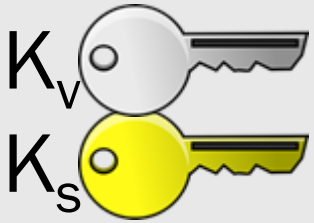
K_e
K_d

$c := Enc_{Ke}(msg)$

$tag := Sign_{ks}(c)$

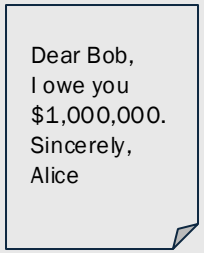
(c,tag)



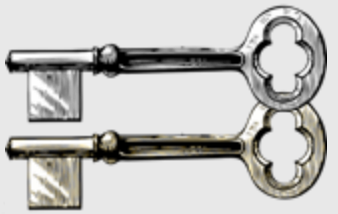


Alice

msg



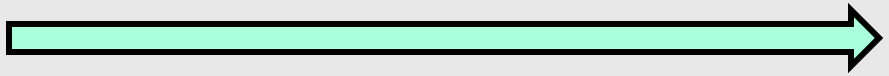
Bob



$c := Enc_{Ke}(msg)$

$tag := Sign_{ks}(c)$

(c,tag)



$isValid := Verify_{Kv}(c,tag)$

$msg := Dec_{kd}(c)$

A Note of Caution

- Beware of determinism
- Use unique keys for encryption, signing, generating MACs
- Use appropriate algorithms for the desired property
- Do not roll your own