10/28/26

COMP 435: *Computer Security Concepts*



Lecture 26: MACs, Digital Signatures



The University of North Carolina at Chapel Hill

https://tinyurl.com/comp435-fa26

Logistics



- Quiz next class
- Lab 3 is out
  - Due 11/5

Quiz Topics



- Cryptanalysis and attack models
- One-time pads: security requirements and key reuse
- Substitution, transposition, confusion, and diffusion
- Stream ciphers and block cipher modes: ECB, CBC, OFB, and CTR
- RSA encryption and decryption: key roles and basic modular arithmetic
- Diffie-Hellman key exchange, machine-in-the-middle attacks
- Cryptographic hashes: collision, pre-image, and second pre-image resistance
- MACs and digital signatures: integrity, authentication, and limitations
- RSA signatures: signing and verification
- Encrypt-then-sign: operation order and which keys to use
- Replay attacks and freshness: the concepts + definitions, how a replay attack could be carried out

Plan for Class Today

- 10:10-10:25 – Finish Collision-Resistant Hashing
- 10:35- 10:50 – MACs
- If time, introduce digital signatures

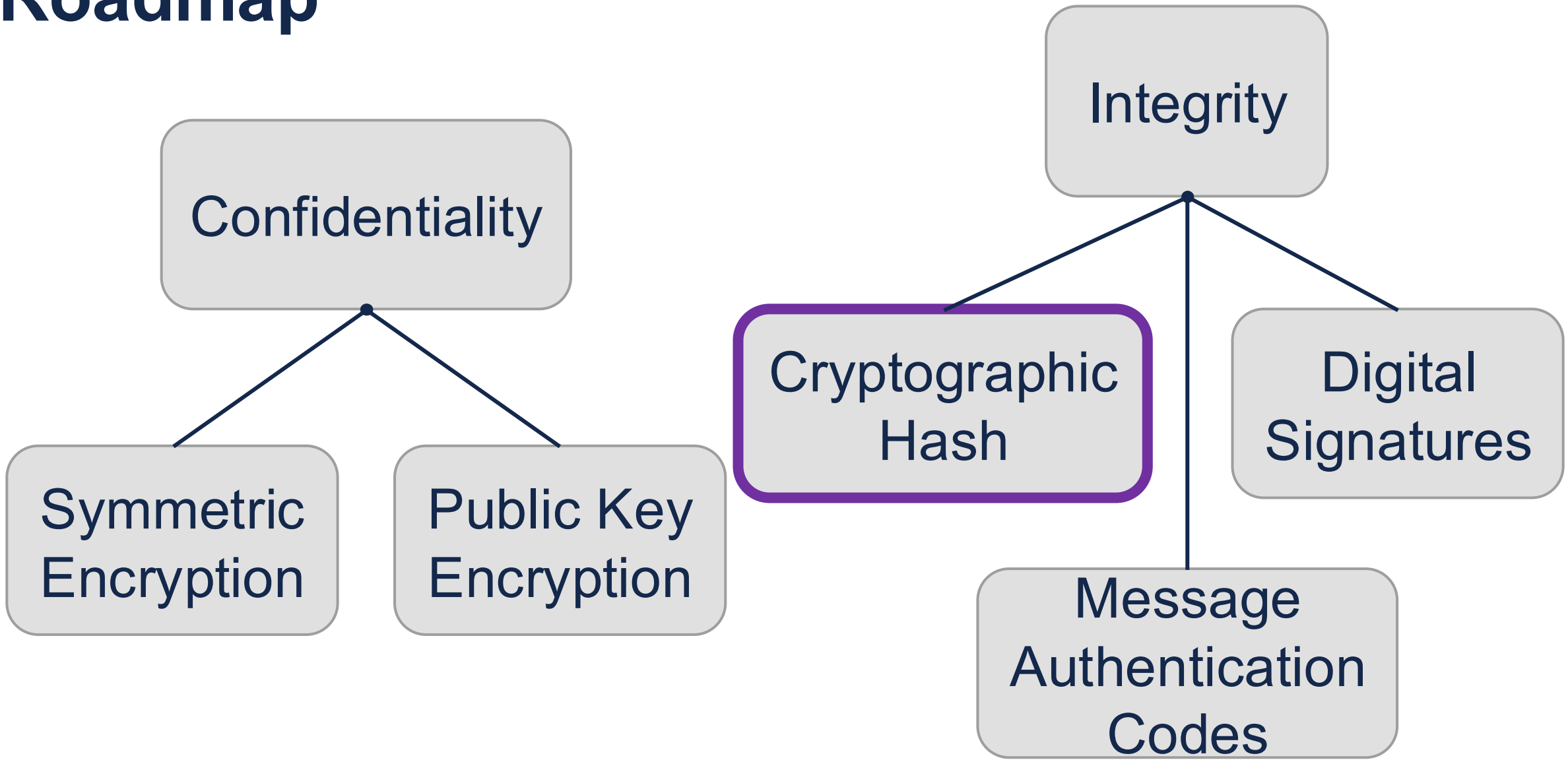
Guiding Question ☺

What cryptographic tools help us ensure the integrity of our data?



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



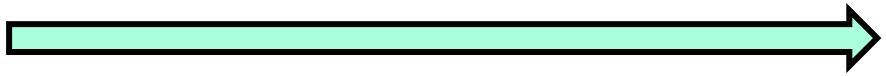




Dear Bob,
Do not trust Mallory!
Sincerely,
Alice

msg

msg



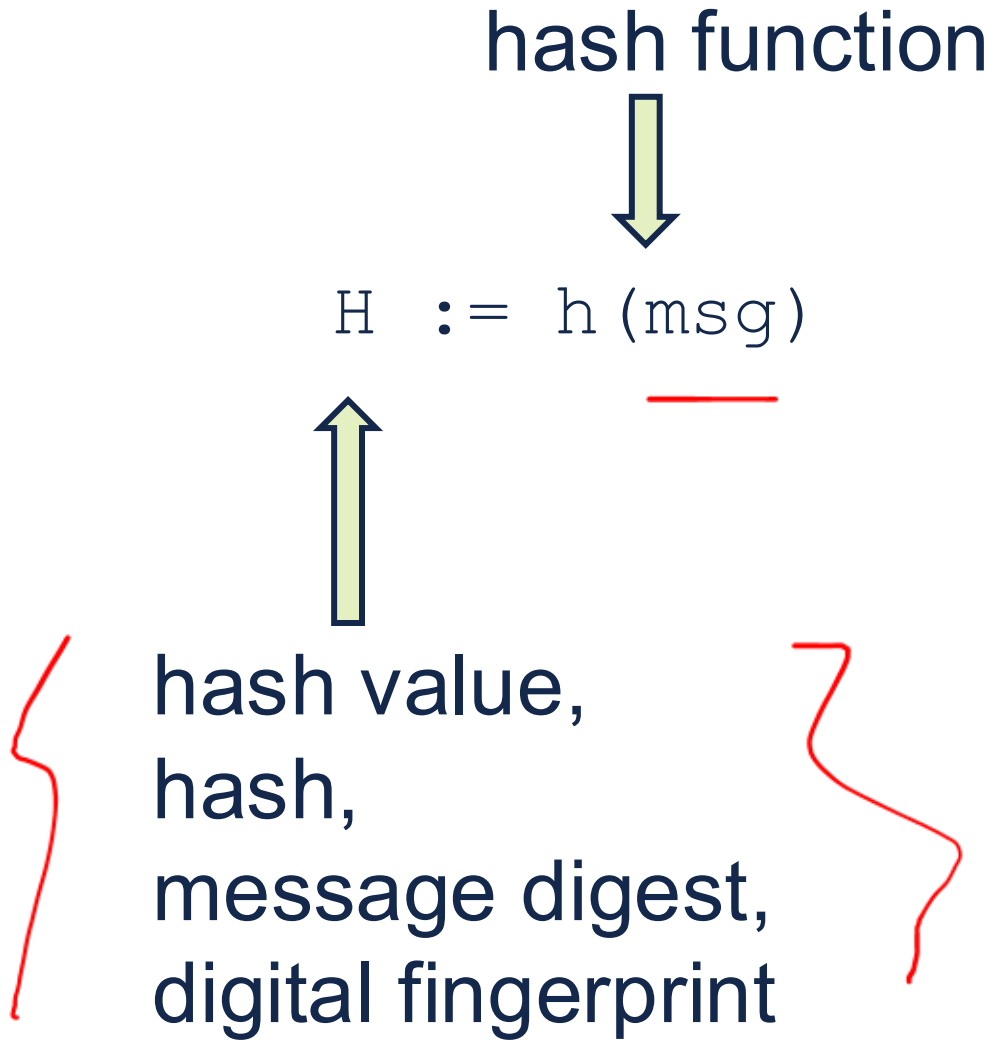
Dear Bob,
You can trust Mallory.
Sincerely,
Alice

msg

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

Tool: cryptographic hash functions!

Terminology



Hash Function



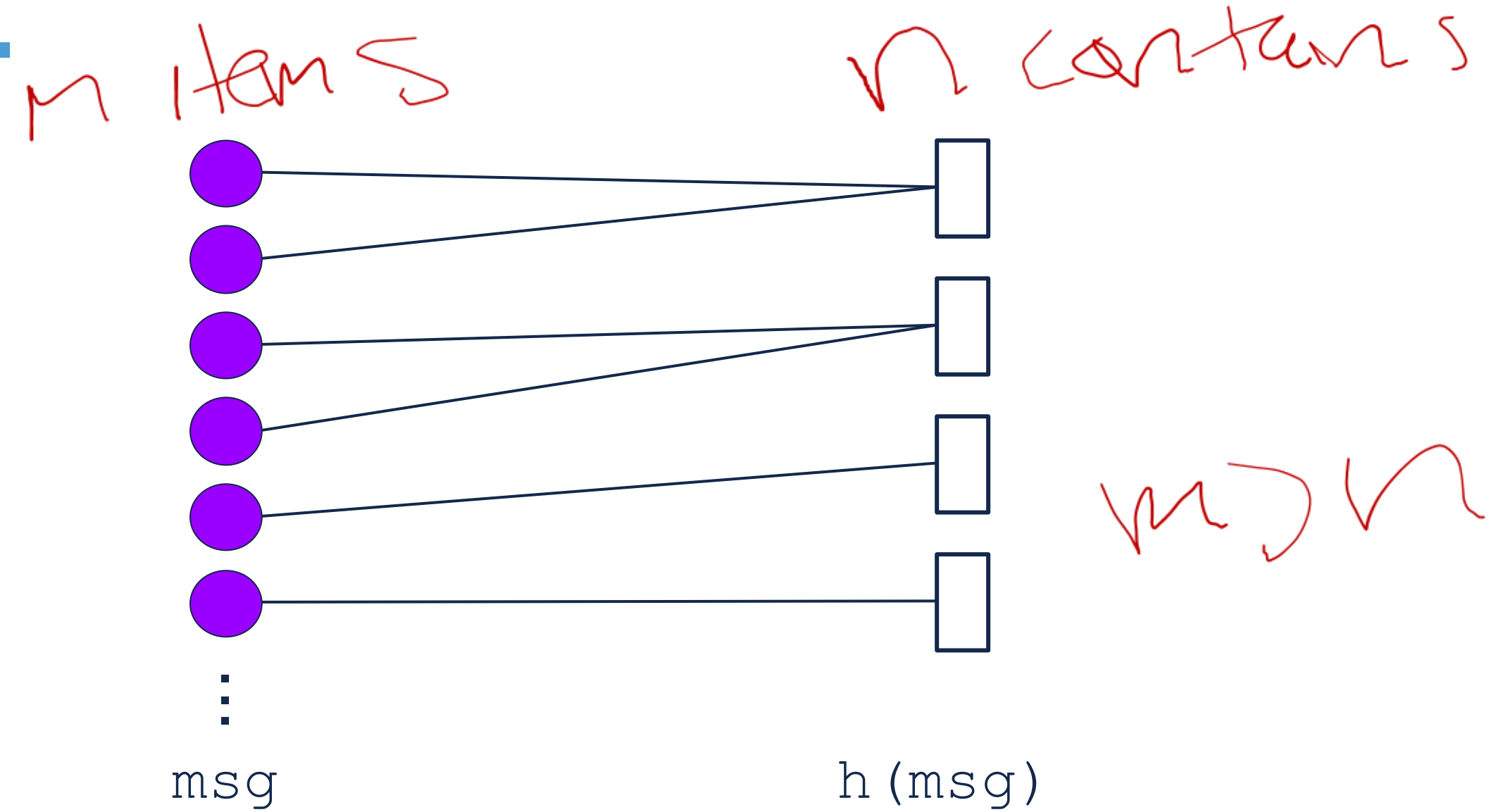
Def'n: a function that takes an arbitrary-length input and produces a fixed-length output

E.g., Modulo operation

3 % 10 = 3
3590224723940253 % 10 = 3

3 collision

Pigeonhole Principle



Functions

surjective

bijective

Recall:
*Bijective: every input has unique output AND
* Every output used
* AKA Injective + surjective

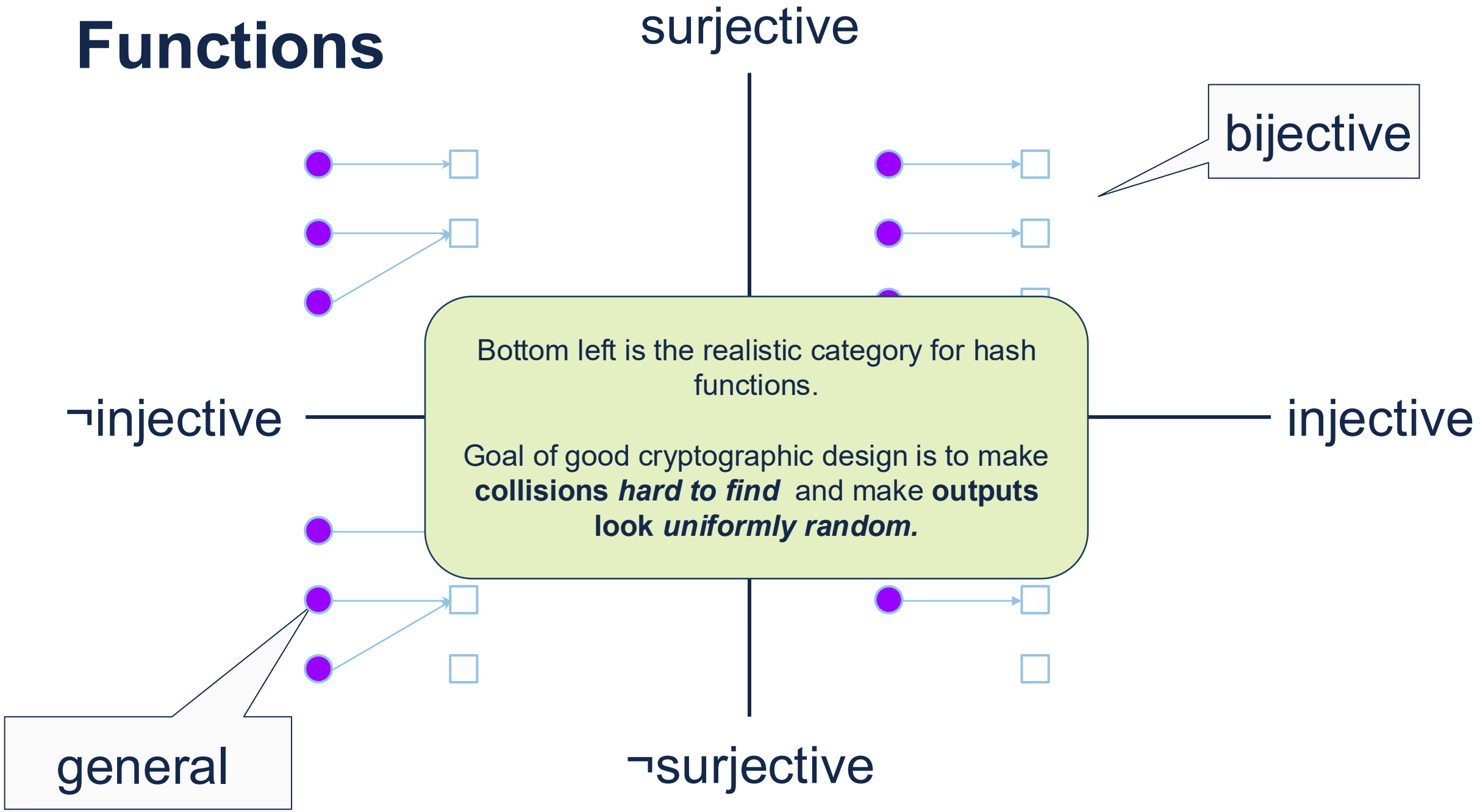
¬injective

injective

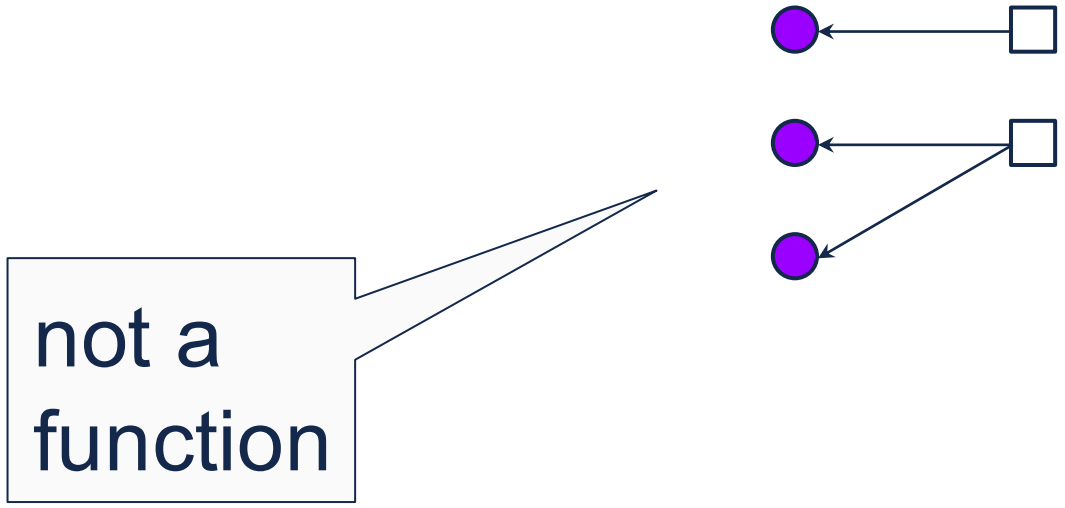
general

¬surjective

Functions

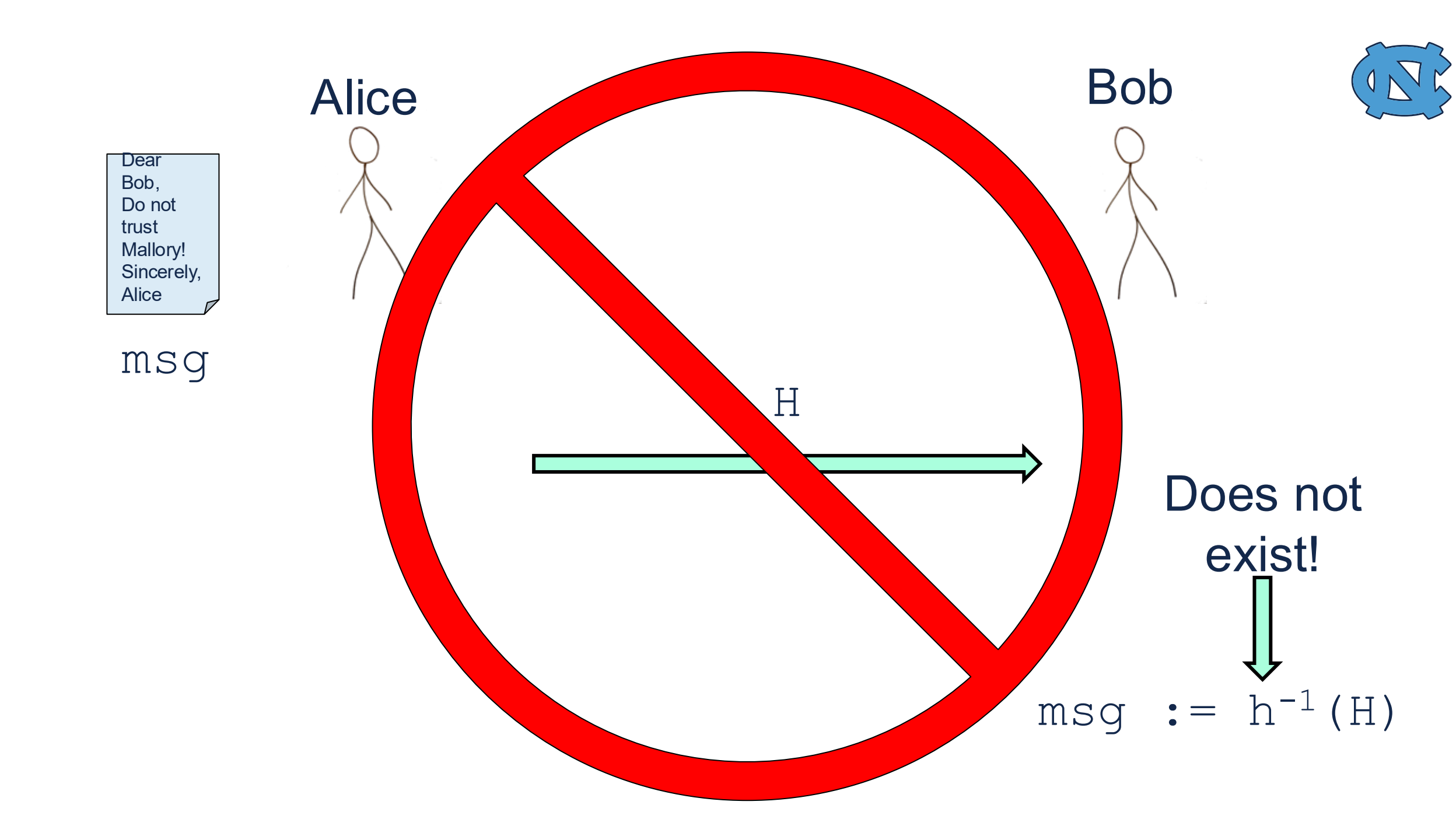

surjective
bijective
¬injective
injective
Bottom left is the realistic category for hash functions.
Goal of good cryptographic design is to make **collisions *hard to find*** and make **outputs look *uniformly random.***
general
¬surjective

Inverse of a hash function?



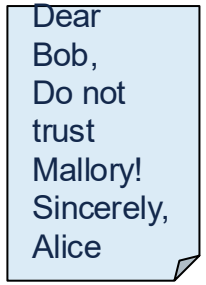
Hash functions not invertible!

One input mapping to two outputs.


Alice
Bob
Dear Bob, Do not trust Mallory! Sincerely, Alice
msg
H
Does not exist!
msg := h^-1(H)

Alice

Bob

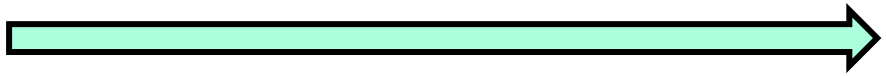






msg

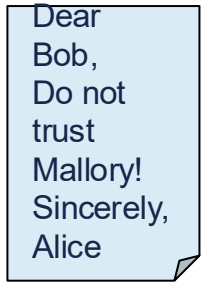
(msg, H)



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

Alice

Bob

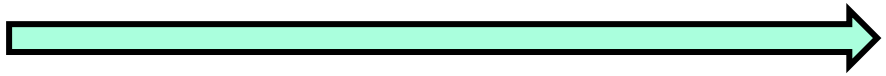






msg

(msg, H)



h ≜ msg % 256

Many collisons! ☹

H ?= 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

These are in decreasing order of strength!

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')`



Why is finding a collision easier for the attacker???

Collision resistant & the birthday paradox

Why Collisions Are Easier to Find

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

Cryptographic Hash Functions

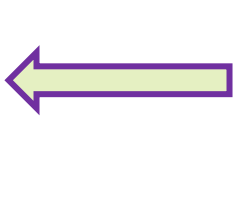


- MD5
- SHA-1
- SHA-2
- SHA-3

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



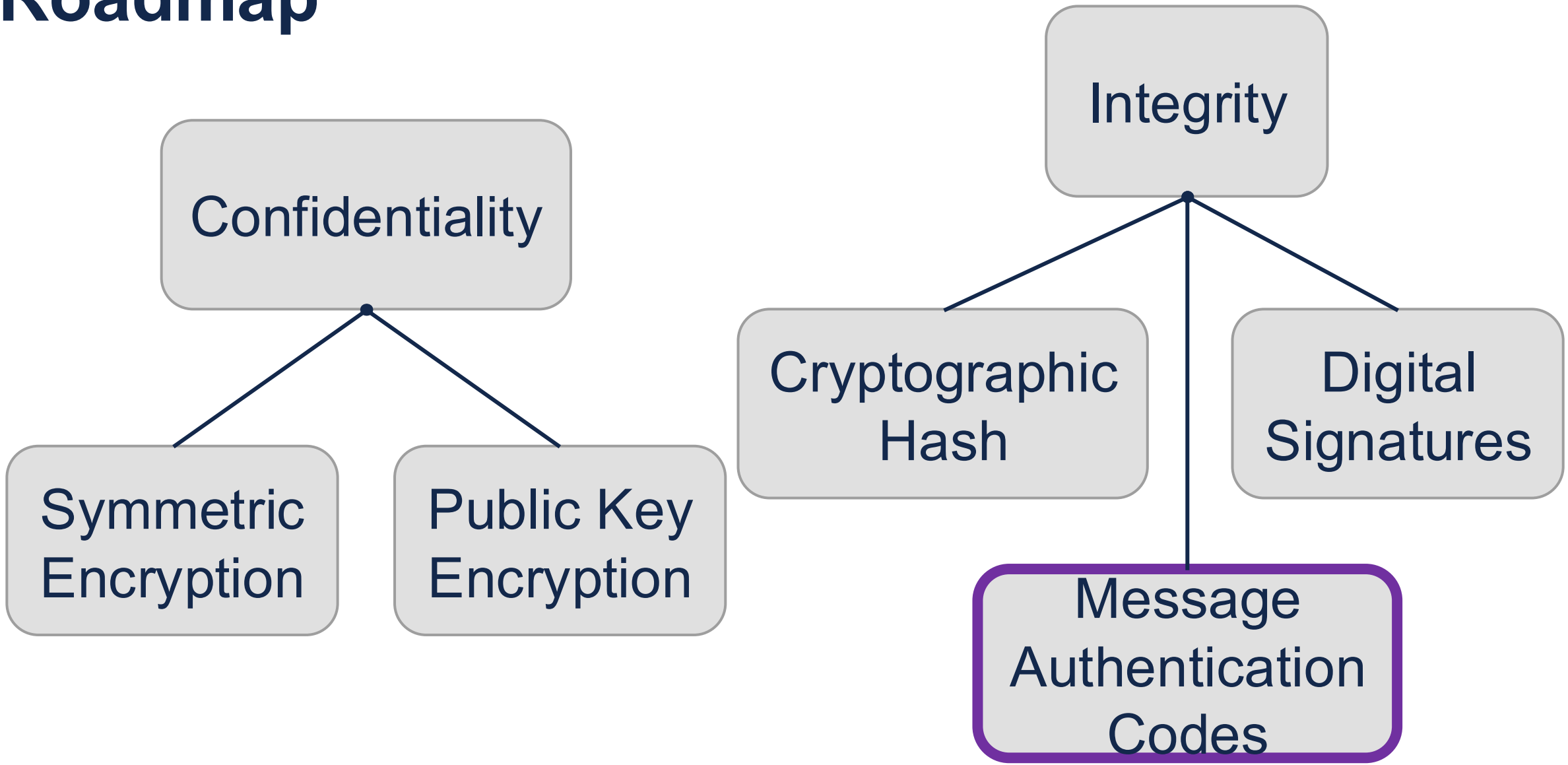
Multiple Variants (different block sizes)

WS Questions! (from lec 25)



Message Authentication Codes (MACs)

Roadmap

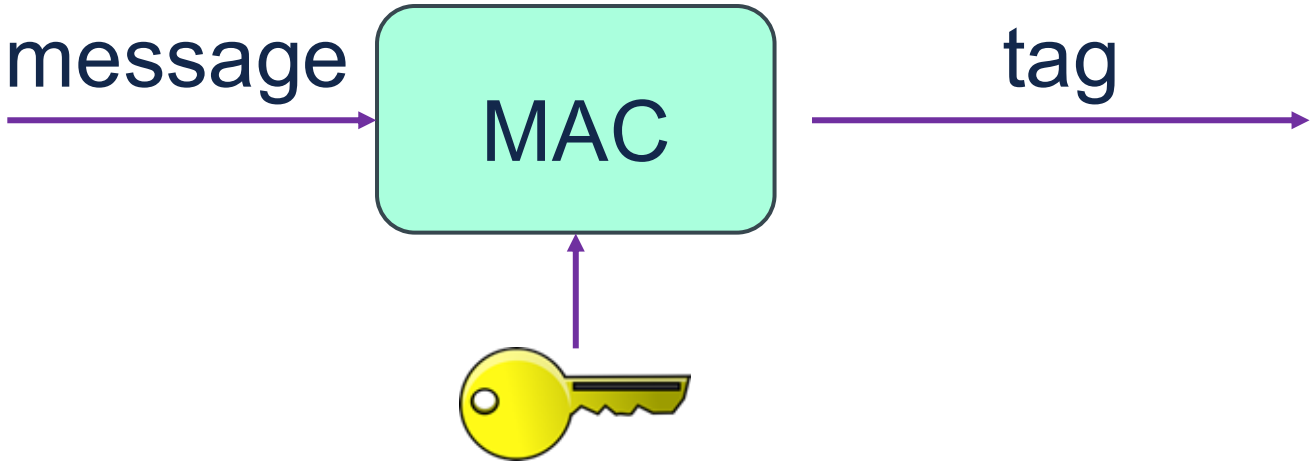

Confidentiality
Symmetric Encryption
Public Key Encryption
Integrity
Cryptographic Hash
Digital Signatures
Message Authentication Codes

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

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

If attacker modifies the message, the MAC that bob computes won't equal the tag

Without the secret key, attacker won't be able to produce valid tag after modifying msg

Properties of MACs



- Unforgeability
- Integrity

Properties of MACs



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

This is provided by the secret key!! This was not previously provided by unkeyed cryptographic hash fns

Properties of MACs



- Unforgeability
- ***Integrity***



Integrity of data

Message Authentication Codes



- HMAC
- CBC-MAC
- CMAC

Message Authentication Codes



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

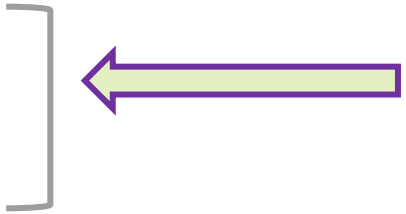


a construction that keys an underlying collision-resistant cryptographic hash function

Message Authentication Codes



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



combines a block cipher with a secret key



Message Authentication and Confidentiality

Message Authentication and Confidentiality



1. Encrypt-and-Authenticate

2. Authenticate-then-Encrypt

3. Encrypt-then-Authenticate

1. Encrypt and Authenticate



Alice





Bob



msg

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

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

(c,t)

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

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

Note: Alice and Bob share two symmetric keys in these next few slides.

Not a public/private key pair. These are two separate shared keys

1. Encrypt and Authenticate



Alice





Bob



msg

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

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

(c,t)



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

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

Not secure, doesn't provide (generally) msg authentication and confidentiality

tag is sent in clear. MACs not necessarily providing confidentiality

2. Authenticate-then-Encrypt



Alice





Bob



msg

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

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

(c)

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

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

Not secure in the general case

Ciphertext is not guaranteed to provide integrity of the message. It may be **malleable**.

3. Encrypt-then-Authenticate



Alice





Bob



msg

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

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

(c,t)



?

t = $MAC_{k2}(c)$

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

Provably secure scheme

Tag is checked first, then decryption is done **only** if tag verifies

Q1 Lec 26 WS

Alice



Bob

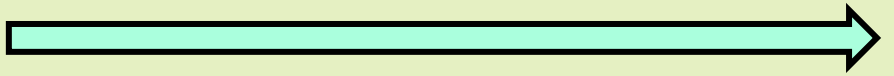


msg

H:= h(msg)



(msg, H)



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

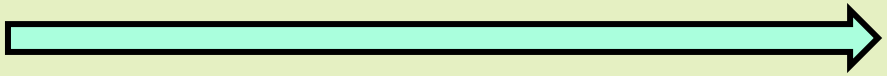
---

msg

tag:= MAC_k(msg)



(msg, tag)

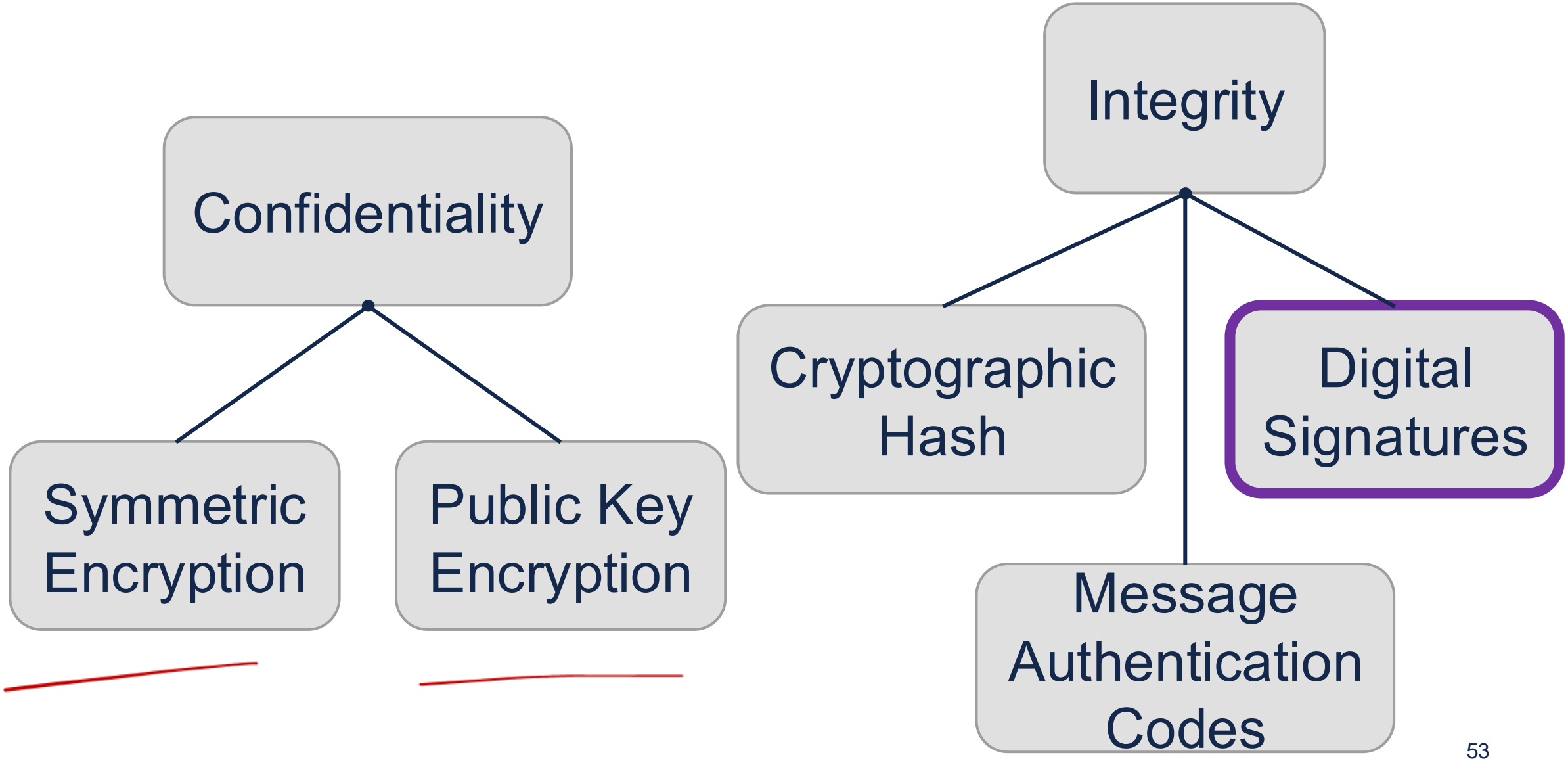


tag $\stackrel{?}{=}$ MAC_k(msg)



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**
- Integrity
- Accountability



Authentication of data origin

Properties of a Digital Signature



- Authentication
- **Integrity**
- Accountability



Integrity of data

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

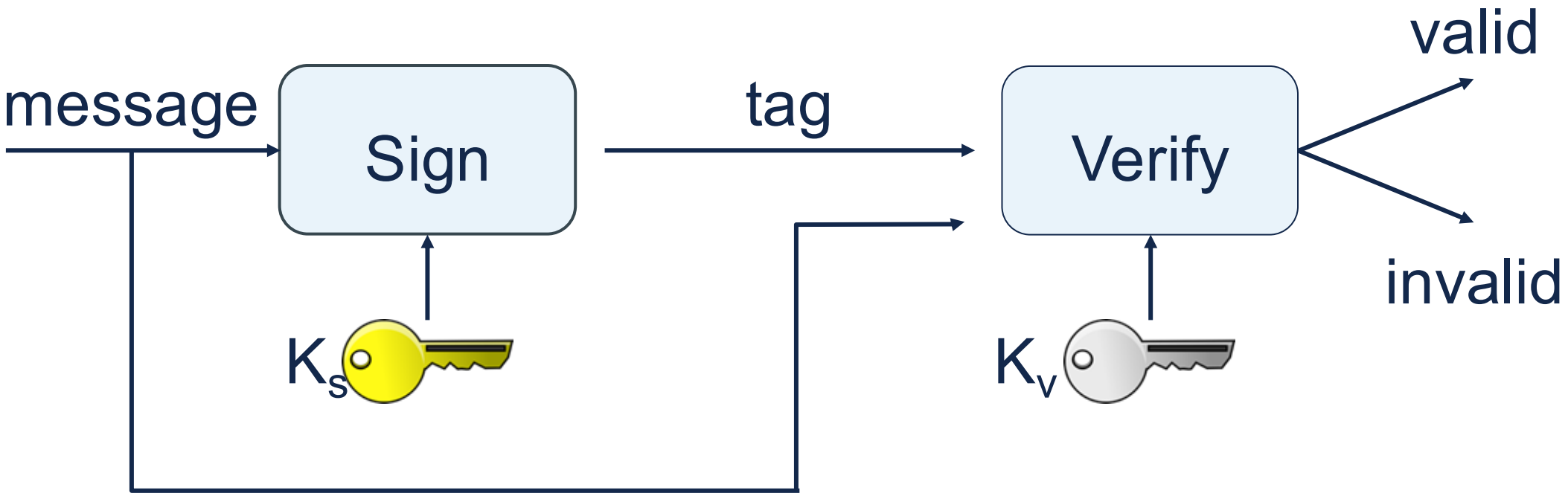
Digital Signature Requirements

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

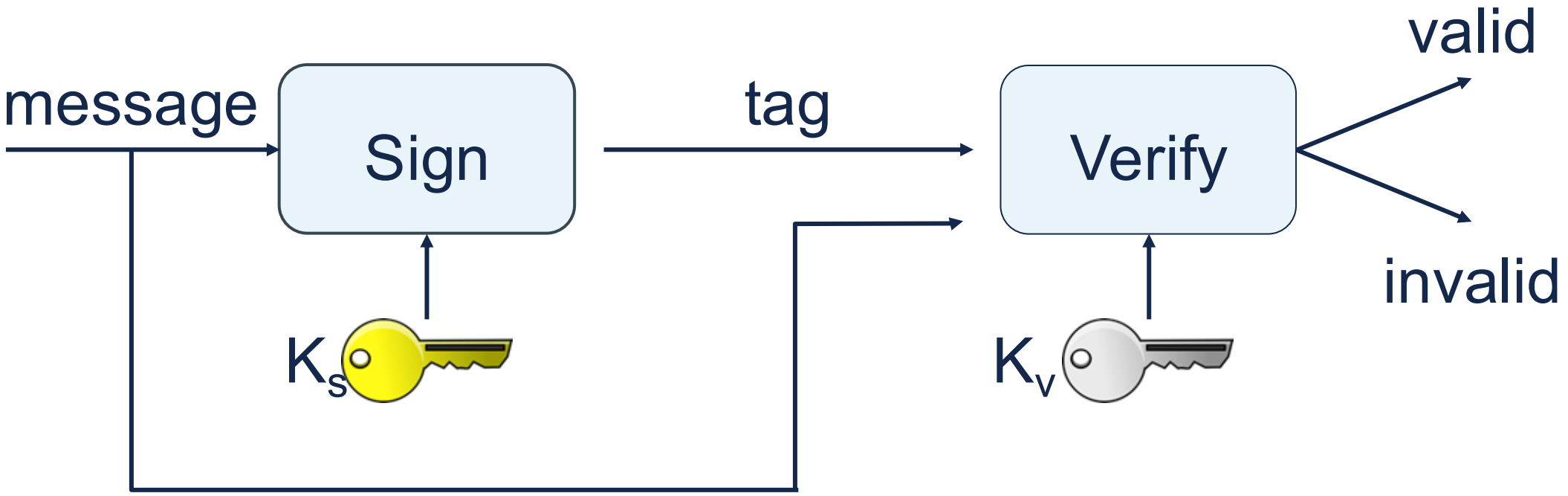


A signature is tied to the signed message

Public Key Signatures



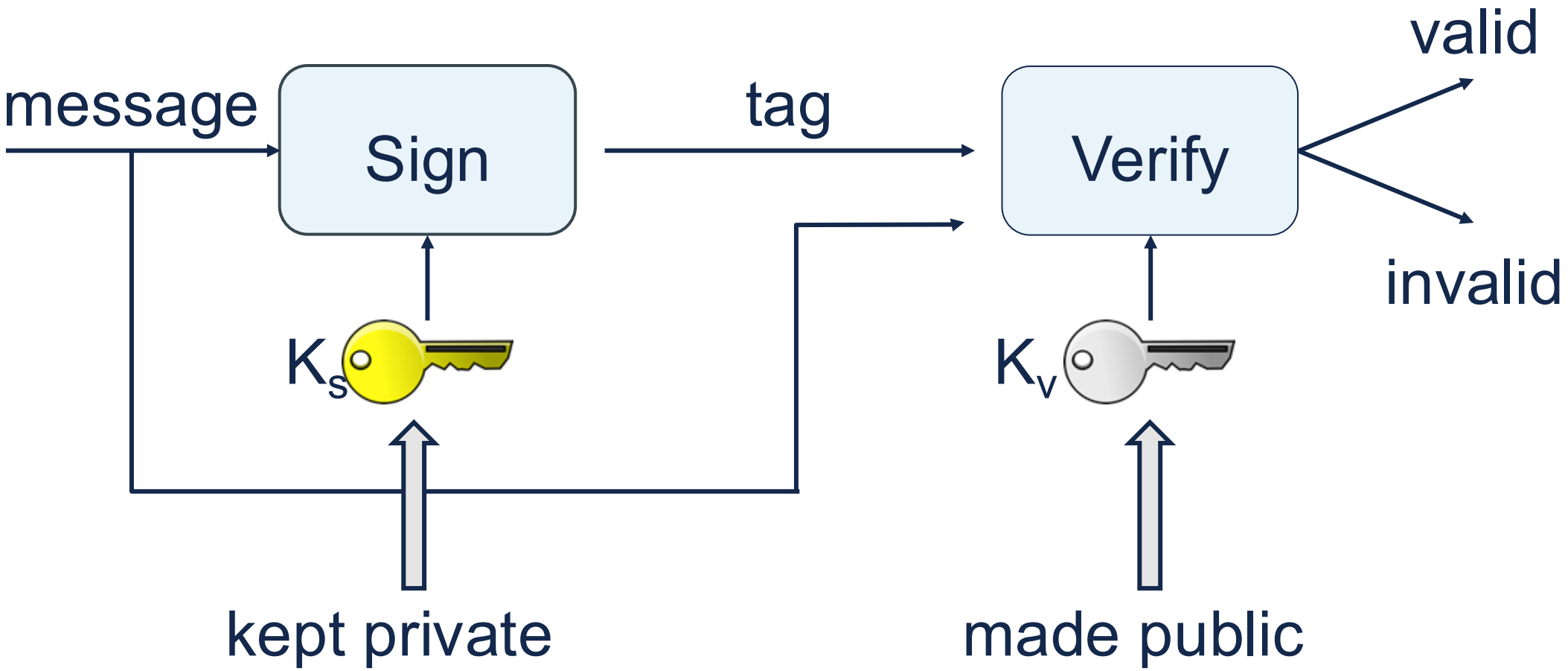
Public Key Signatures



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

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

Public Key Signatures



msg

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

K_v

K_s

Alice



Hello, World.
This is my
public key.

K_v

Bob



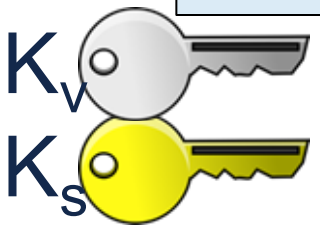


msg

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

Alice

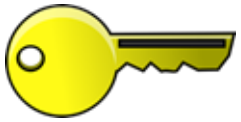




Bob



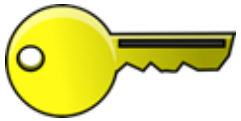
$tag := Sign_{K_S}(msg)$







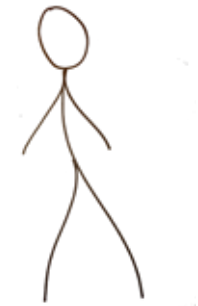
$tag := Sign_{Ks}(msg)$



(tag, msg)

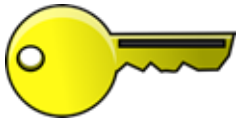






Bob

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



(tag, msg)



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





RSA for Digital Signatures

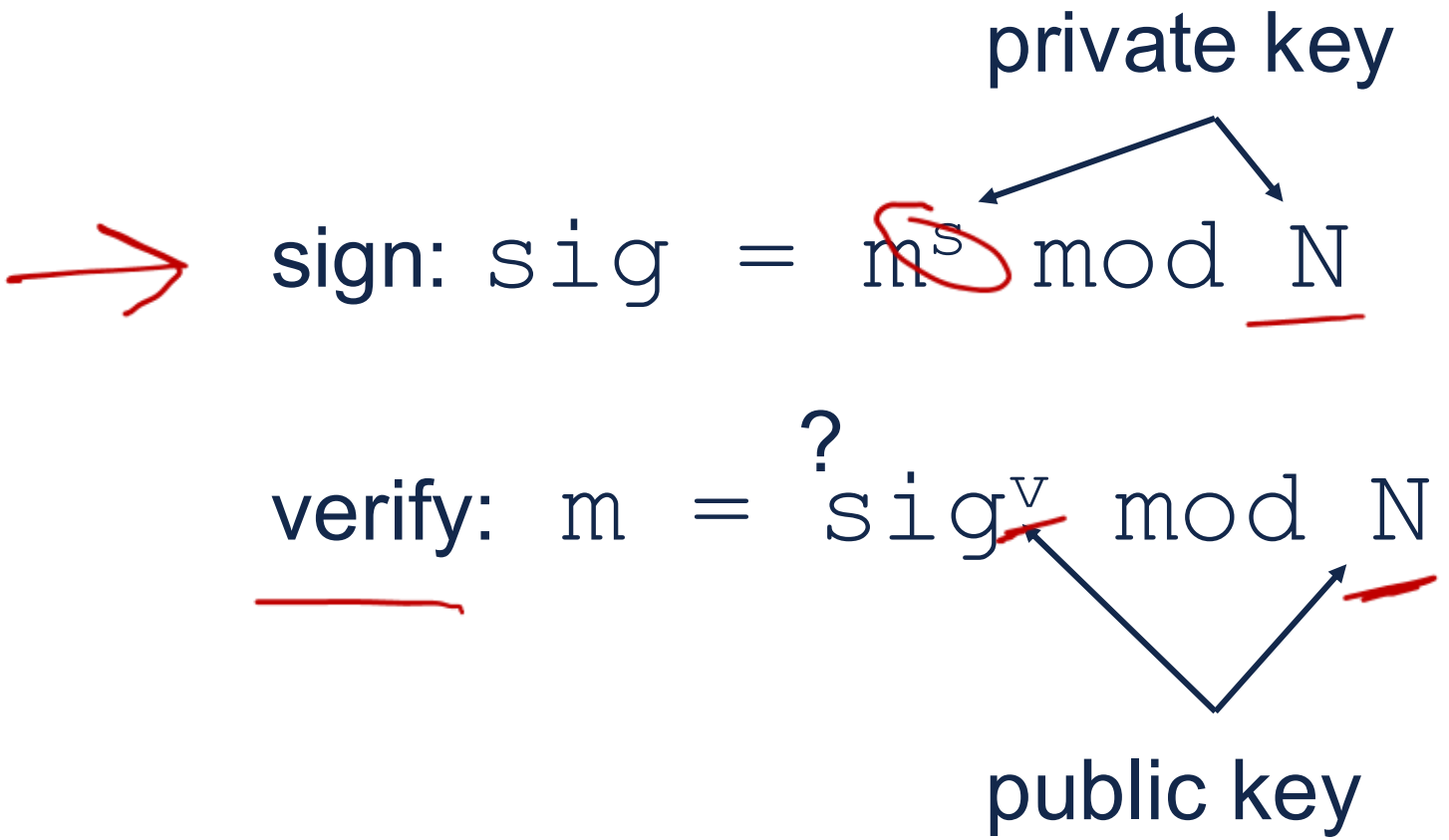
RSA for digital signatures



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

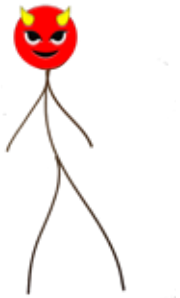
verify: $m \stackrel{?}{=} sig^v \bmod N$

RSA for digital signatures



(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

K_v

K_s

Alice



Hello, World.
This is my
public key.

K_v

Bob



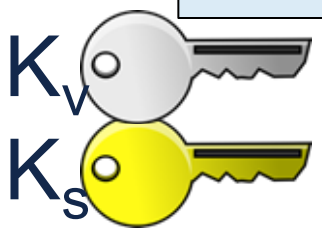


msg

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

Alice

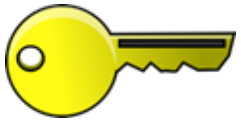




Bob



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

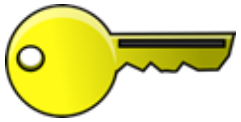








$tag := Sign_{Ks}(h(msg))$



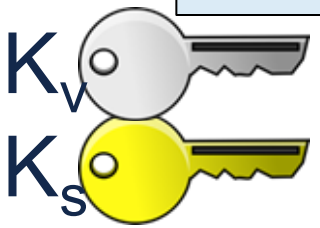
(tag, msg)



msg

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

Alice



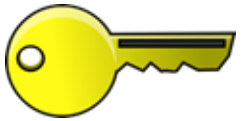


Bob





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



(tag, msg)



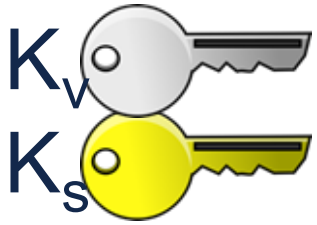
$isValid = Verify_{Kv}(h(msg), tag)$



WS Q2



Encrypt then Sign



Alice

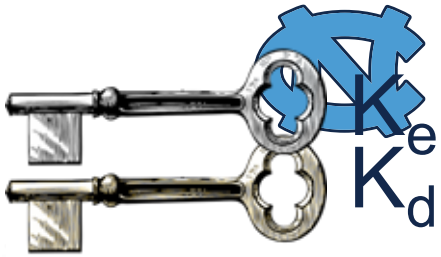


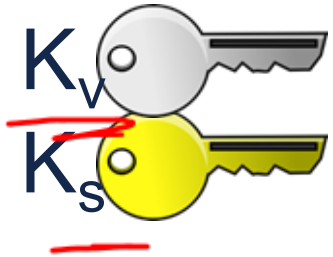
Hello, World. This is my public key for verifying my signature.



Bob



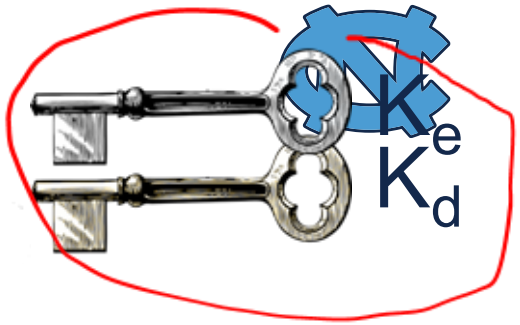




Alice



Bob

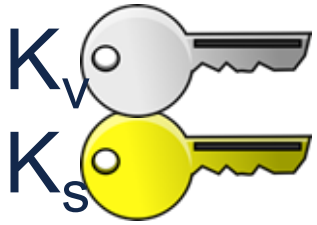




Hello, World. This is my public key for encrypting.



K_e



Alice

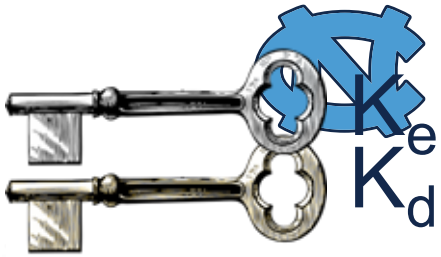


msg

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

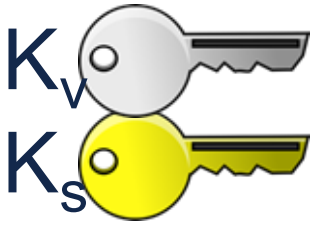
Bob





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

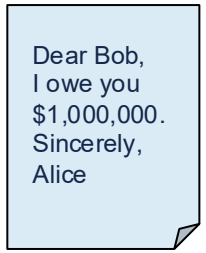
$tag := Sign_{ks}(c)$



Alice

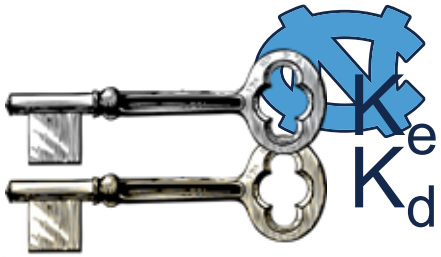


msg



Bob



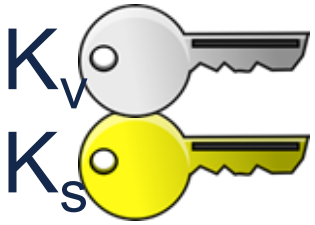


$c := Enc_{Ke}(msg)$

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

(c,tag)





Alice

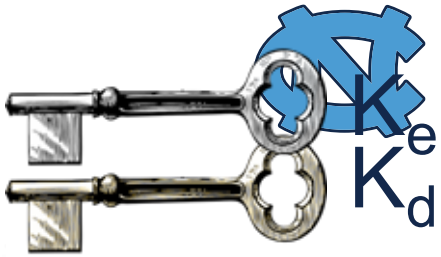


msg

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

Bob





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

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

(c,tag)



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

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

WS Q3

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