LectureLet 17

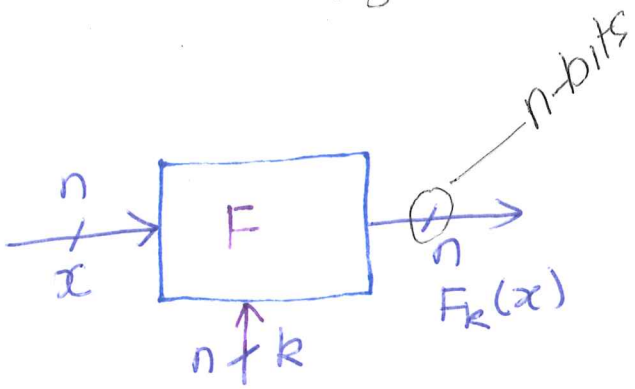
Date: Oct 13, 2020

Last time

- Mac-forge game
- Replay attacks
- Timing attacks

→ application layer

F PRF



$\Pi = (Gen, Mac, Vrfy)$

Gen:

$k \leftarrow Gen(1^n)$

↑ uniform n-bit string

Mac:

$t := F_k(m)$ ← n-bits

t: tag

$F_k(m) \rightarrow$ n-bits, $l(n) = n$ fixed length mac

Vrfy:

$$Vrfy_k(m, t) = \begin{cases} 1 \\ 0 \end{cases}$$

$t \stackrel{?}{=} F_k(m)$ ← comparison operator ($|m| = |k| = n$)

otherwise

↑ canonical verification

$(F \text{ is a PRF}) \Rightarrow (\Pi \text{ is secure})$ using Mac-forge.

$\neg(\Pi \text{ is secure}) \Rightarrow \neg(F \text{ is a PRF})$ contrapositive.

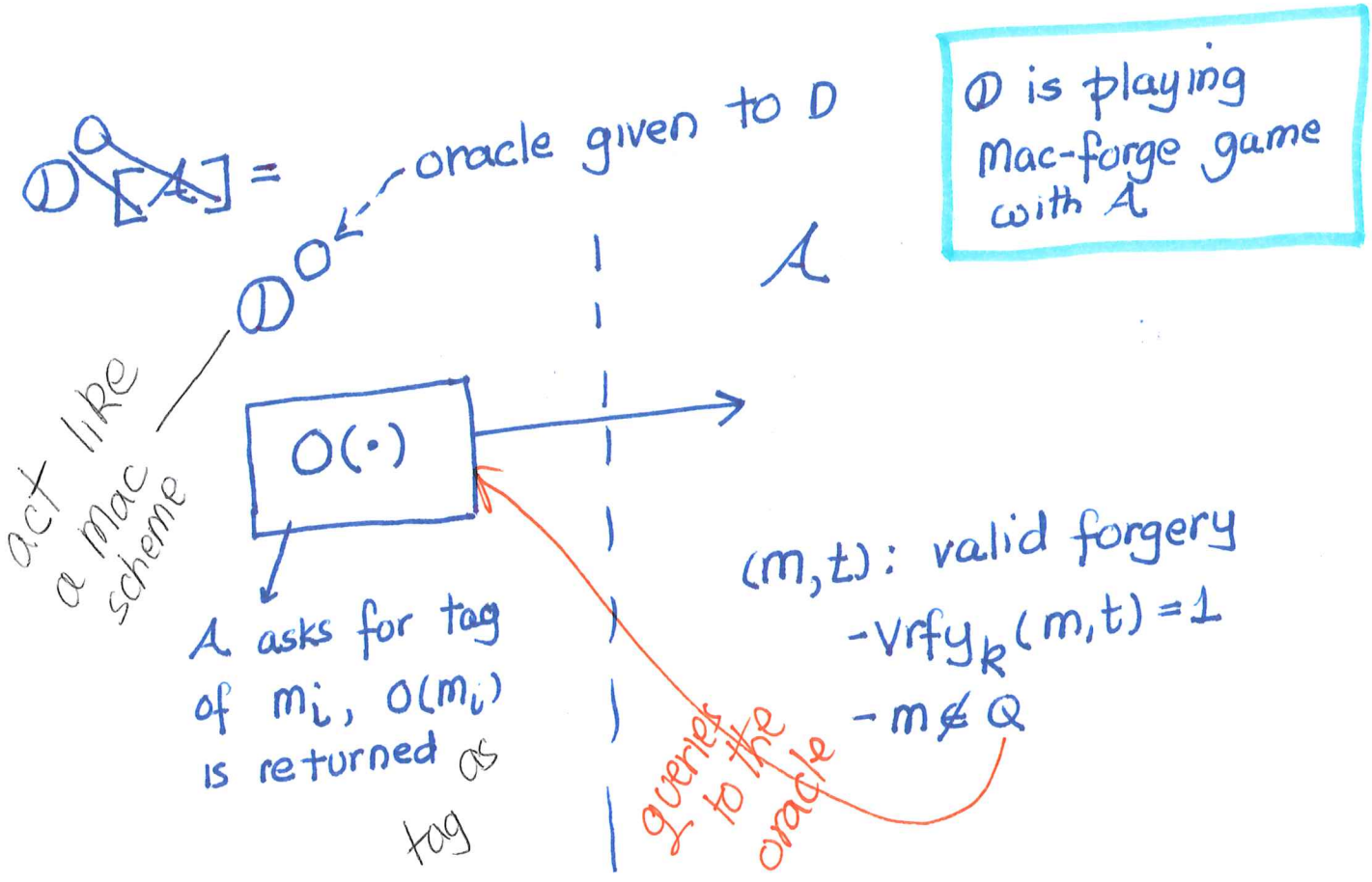
not ⋮ PPT

$\mathcal{A}$ adv. that can win the mac-forge game with Π with non-negl probability

$$Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] = f(n) \neq \text{negl}.$$

Using $\mathcal{A}$, $\mathcal{D}$ will create a distinguisher D

for $F \Longrightarrow F \neq PRF$



$D^0(A) = \begin{cases} 1 & \text{if } A \text{ produces a valid forgery } (m,t) \\ 0 & \text{otherwise} \end{cases}$

0 → world 0, 1 → world 1

World 0:

PRF

$O = F_k(\cdot)$

Exactly the mac-forge game

A given oracle access to $mac_k(\cdot)$

$Pr[D^{F_k(\cdot)}(1^n) = 1] = f(n) \neq \text{negl.}$

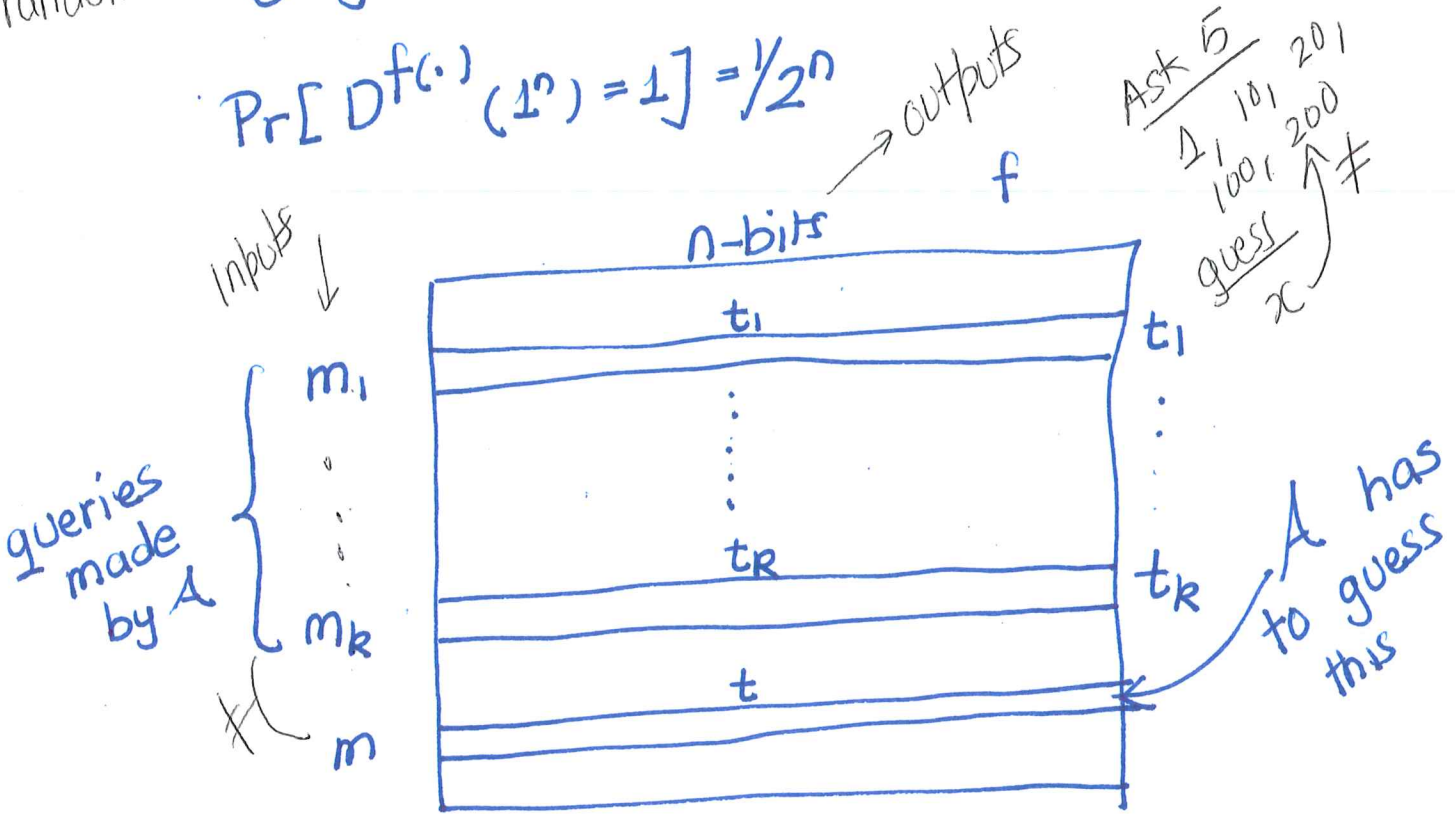
$= Pr[\text{Mac-forge}_{A,\pi}(n) = 1]$

World 1

random

$O = f(\cdot)$ <-- True random fn

$Pr[D^{f(\cdot)}(1^n) = 1] = 1/2^n$



In world 1 A has to guess a n-bit number uniformly distributed

$1/2^n$

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right|$$

world 0 — world 1

$$= f(n) - 1/2^n$$

$f(n)$ ≠ negl, $1/2^n$ = negl

$f(n) \neq \text{negl}$

$$g(n) = f(n) - \frac{1}{2^n}$$

$$g(n) + \frac{1}{2^n} = f(n)$$

proof by contradiction

If $g(n)$ was negl then

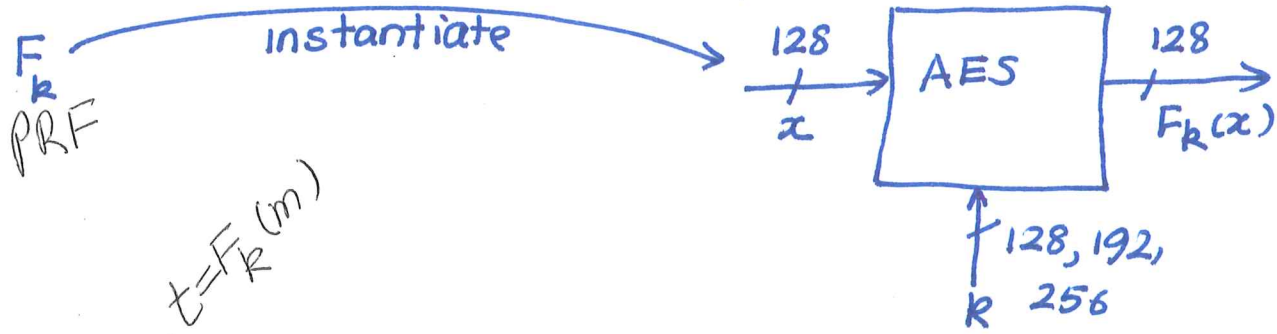
$f(n)$ is negl

(contradiction)

Last time
- MAC using PRF
- Proof it is secure

LectureLet 18

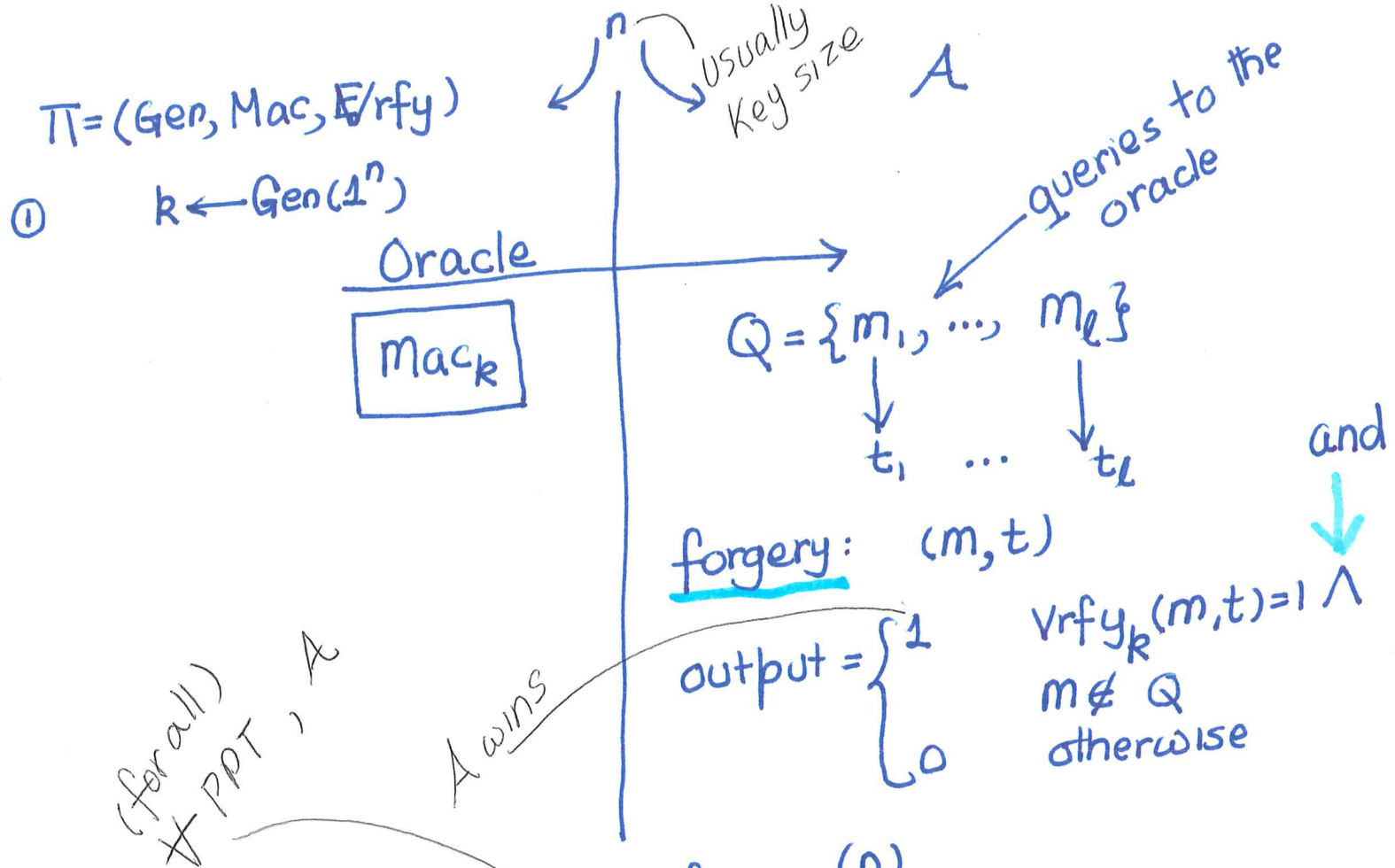
Date: Oct 17, 2020



$t = F_k(m)$

Can compute MAC of 128-bit msgs?

- what about larger msgs.



$$Pr[Mac\text{-}forge_{A,\Pi}(n) = 1] \leq negl(n)$$

Instructive

DOMAIN EXTENSION

MAC works on n-bits (e.g. AES works on 128-bits) → MAC on larger messages (1 MB msg)

ECB, CBC, OFB | CTR

Similar to modes of operation (MOO)

IDEA 1 ✗

Blocks of messages

$m = m_1 \dots m_d$

MAC → $t_1 \dots t_d$

very similar ECB (secure) MOO

oracle $MAC_K(\cdot)$ ask tag for m

Reordering attack

$m = m_1 \; m_2$

$t_1 \; t_2 \leftarrow$ tag

$(m_2 m_1, t_2 t_1)$

$t_2 \; t_1$ is tag for $m_2 m_1$ — $m' \notin Q$ ← valid forgery

≠ secure

IDEA 2 ✗

$m = m_1 \dots m_d$

$MAC'_K(i \| m_i)$ <add index before MAC>

2. verify → $t_1 \dots t_d$

concat: $t_1 = MAC_K(1 \| m_1)$, $t_2 = MAC_K(2 \| m_2)$ ⋮

The previous attack does not work? → $t_2 t_1$

Truncation attack

A asks oracle → $m = m_1 \; m_2$, $t_1 \; t_2$

t_1 is tag for $m_1 \neq m$ — valid forgery

IDEA 3 (X)

Review the MAC-forge game

$m = m_1 \ldots m_d$

$t_1 = MAC_K(\ell \| 1 \| m_1)$

$t_2 = MAC_K(\ell \| 2 \| m_2)$

$\vdots$

$|m|$

$t_1 \ldots t_d$

$MAC'_K(\ell \| i \| m_i)$ ← depend on length of the msg.

added length of the msg in bits

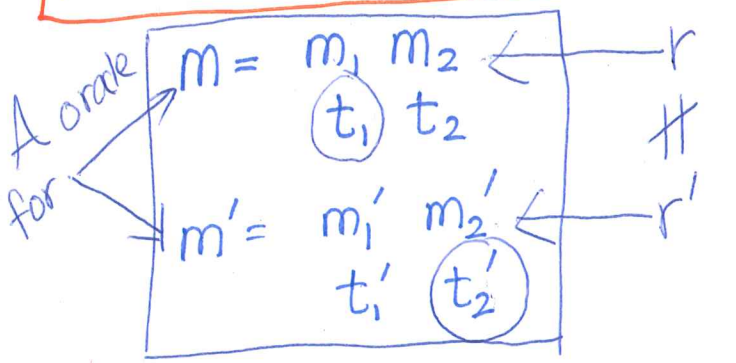
$t_1 = MAC(1000 \| 1 \| m_1)$ ← 1000

Previous attack does not work

to do

len of m

Mix-and-match attack



$t_1\ t_2'$ tag for $m_1\ m_2' \neq m, m'$ ← valid forgery

IDEA ~~3~~ 4 [FINAL]

Pick a random # r per message

$m = m_1 \ldots m_d$

$t_1 = MAC_K(r \| \ell \| 1 \| m_1)$

$t_1 \ldots t_d$

$MAC_K(r \| \ell \| i \| m_i)$ ← random #, len, index

added random #r before MACing

Previous attack does not work

Last time

Lecture Let 19

Domain extension for MACs

Logistics

- HW 3 ~~to go~~ out ~~soon~~

~~Sept~~ Oct 2020 18

$m = m_1 \ldots m_d$ (blocks)

$t_1 \ldots t_d$

$MAC_k(r \| l \| i \| m_i)$ → 32 (128)

r: random #, i: index

wastes bandwidth/space

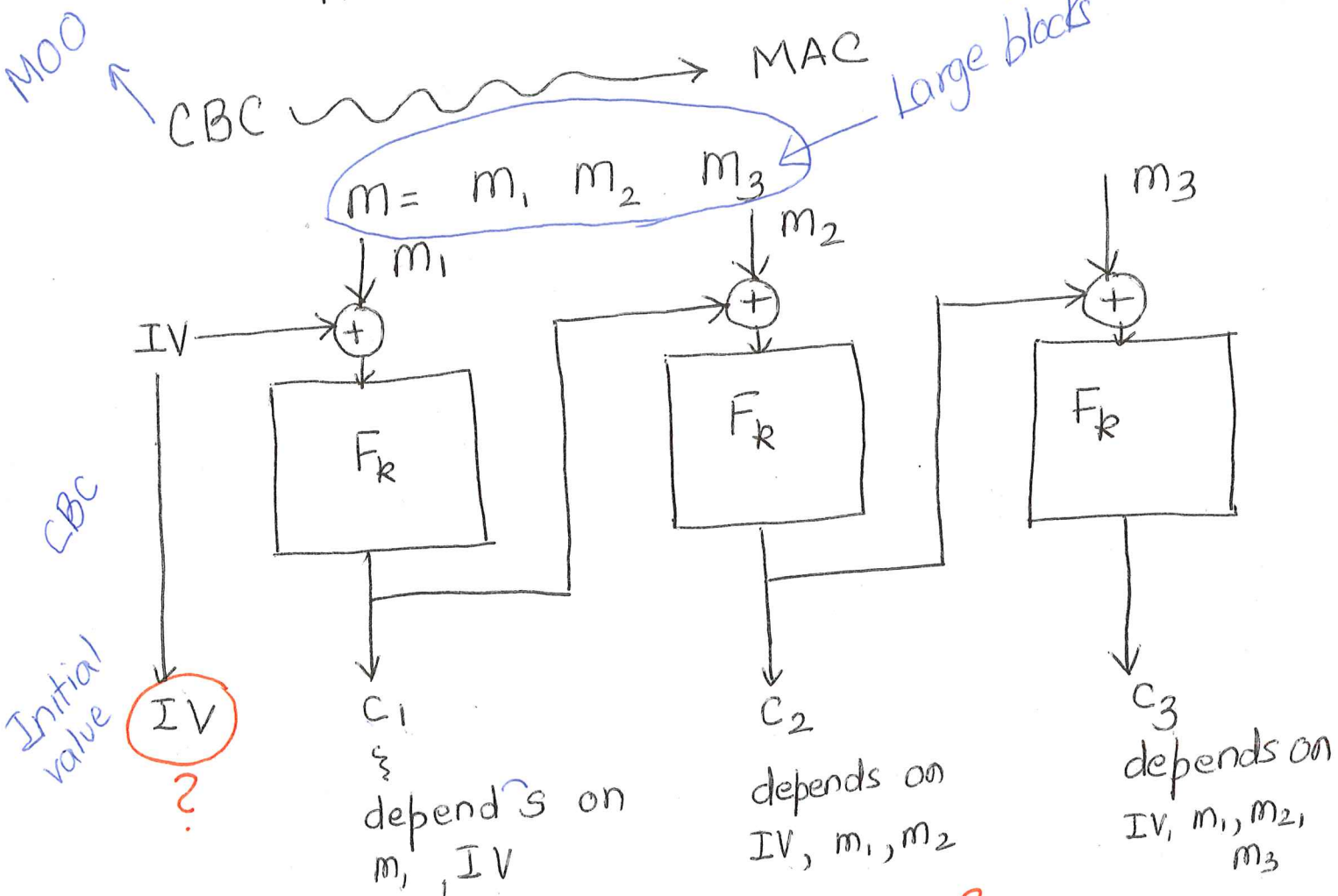
len of $|m|$ 32/128

MOO

CBC → MAC

Large blocks

$m = m_1\ m_2\ m_3$



c_1 depends on m_1, IV

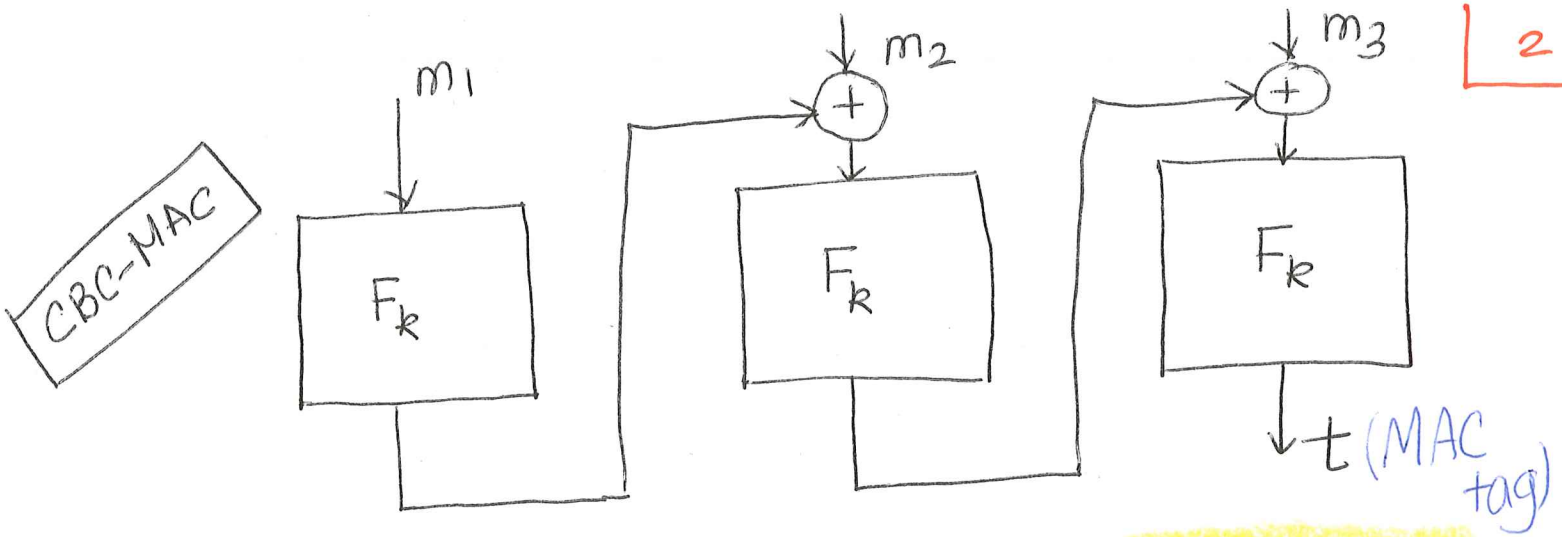
c_2 depends on IV, m_1, m_2

c_3 depends on IV, m_1, m_2, m_3

why not use c_3 as a MAC?

$IV = 0$

Recall: $0 \oplus m_1 = m_1$



-Fix #3 blocks

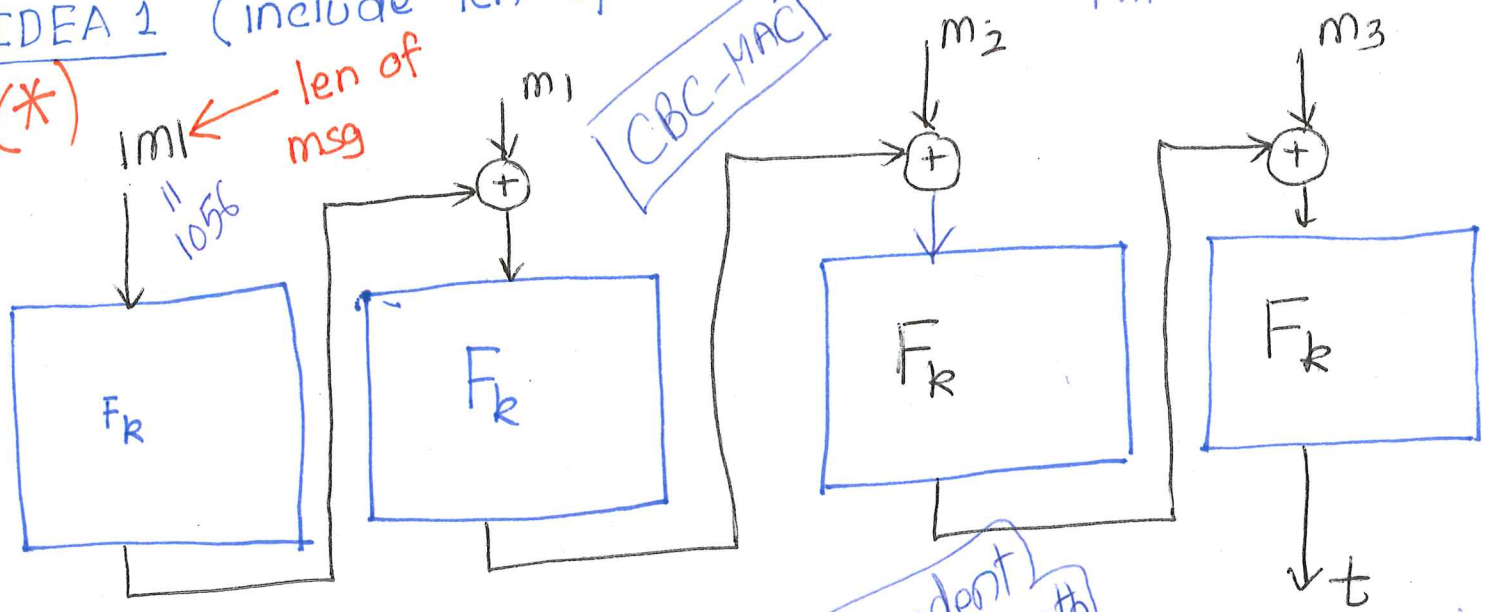
Problem: only works for fixed length messages

How to extend to arbitrary length msgs?

IDEA 1 (include len of the msg)

(*) |m| ← len of msg

1056 bits = |m|



dependent on length

IDEA 2 (two keys)

k_1 k_2

without adding original |m|

$t = \text{CBC-MAC}_{k_1}(m)$

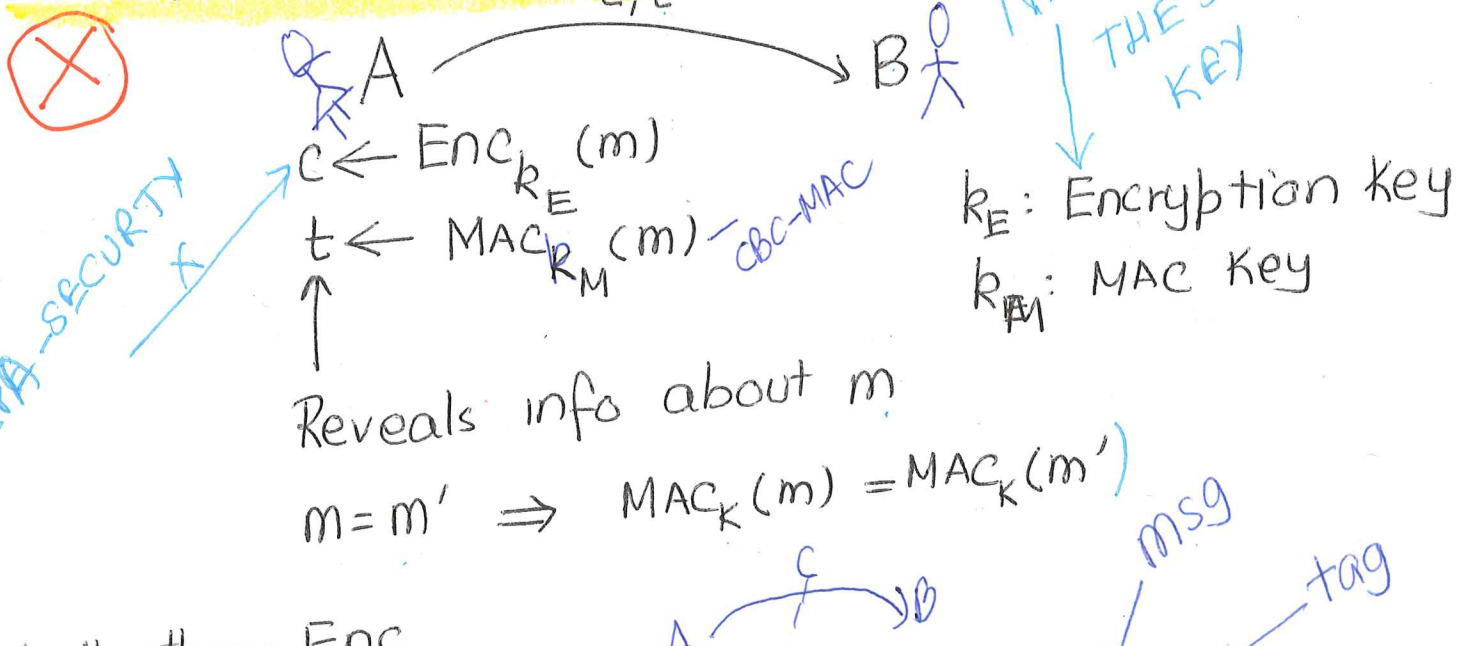
2 Keys

$\hat{t} = F_{k_2}(t)$

Authenticated Encryption

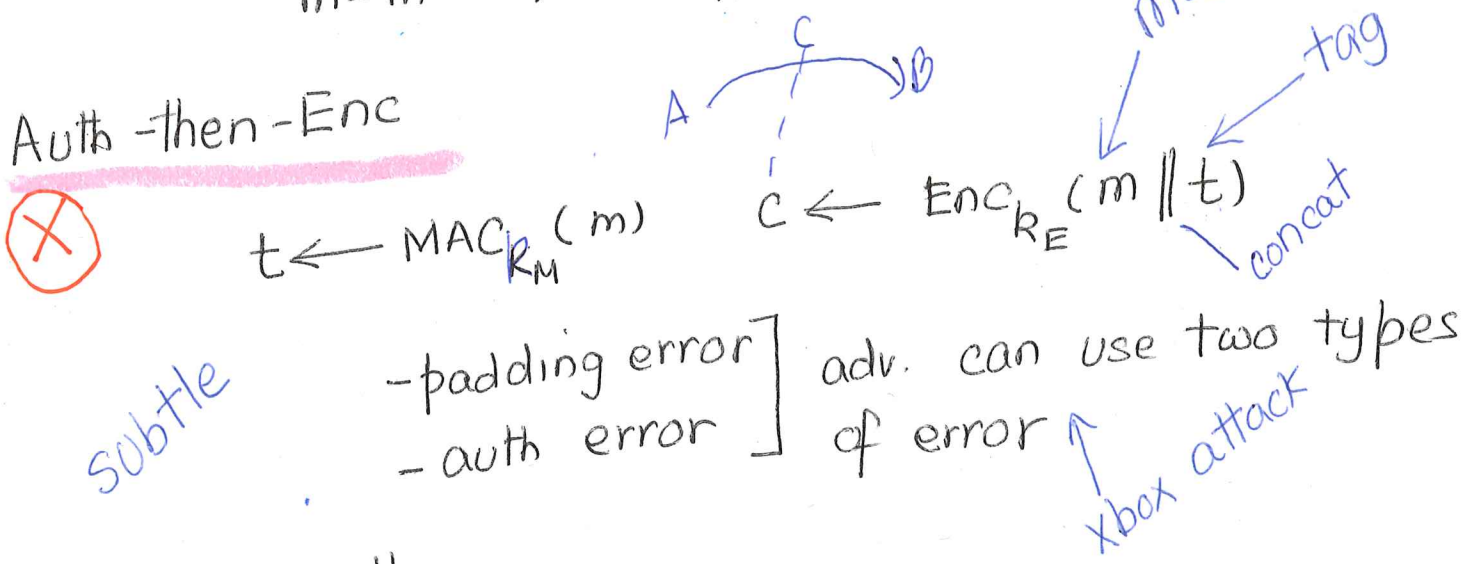
Secrecy + Authentication ← both

- Encrypt-and-authenticate

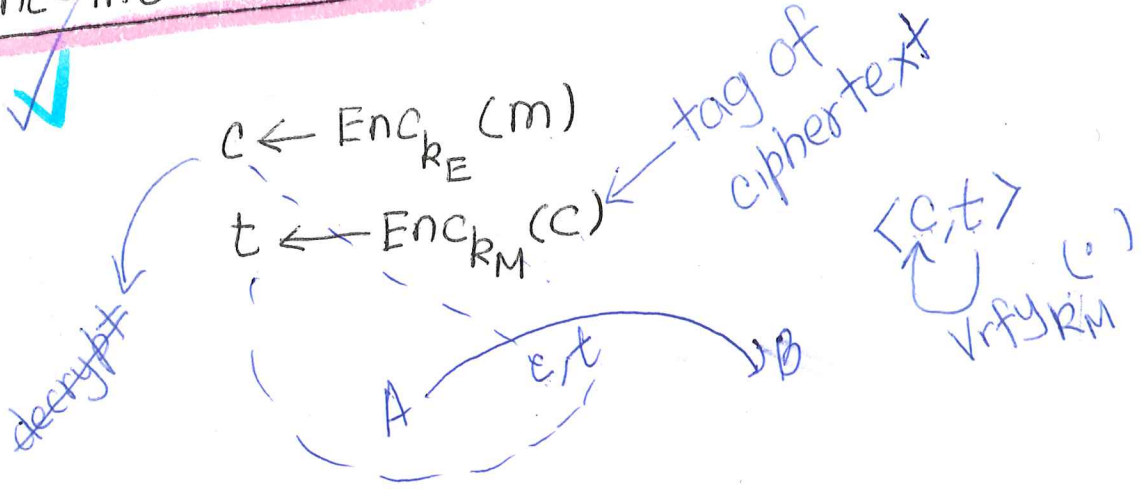


$m = m' \Rightarrow MAC_K(m) = MAC_K(m')$

- Auth-then-Enc



- Enc-then-auth



Lecture Let 20

Date: Oct 20 2020

Last time

- Domain extension for MACs (CBC-MAC)
- Authentication + secrecy
  - Enc-then-Auth ✓
  - Enc-and-Auth
  - Auth-then-Enc
  - ✗ check canvas

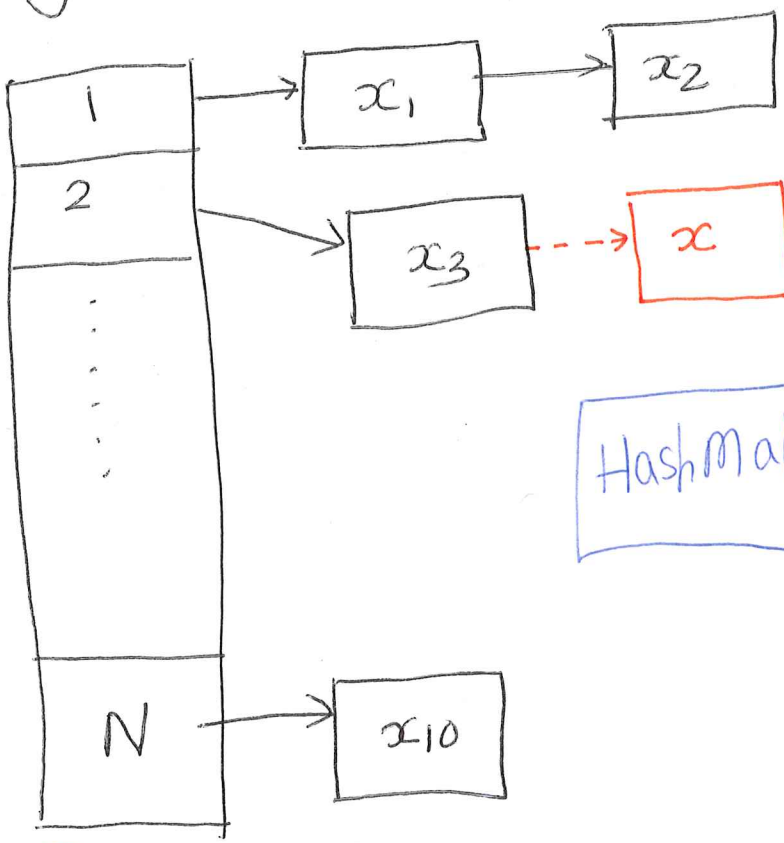
Hash functions

- Probably seen ~~as~~ in data structures

$H: O \longrightarrow \{1, \ldots, N\}$

$N = 10{,}000$

O: objects/images



Look up x

- compute $H(x)$
- say $H(x) = 2$

$x_3 = x$ (Yay) ✓

$x_3 \neq x$

collisions are bad

$H(x) = H(x')$ means they are in the same bucket

$x \neq x'$

if there are too many $o \in O$, then ....

such that $H(o) = 2$

H should map ○ uniformly to [1...N]

↑ images

But no adversary

Cryptographic hash functions are harder

- Adversary present

-Def:

hash function with output len $\ell(\cdot)$

-Gen:

$s \leftarrow Gen(1^n)$

↑ key of len n

Ex: $n^3 = 100^3$ or $n = 100$

- H^s:

$H^s: \{0,1\}^* \rightarrow \{0,1\}^{\ell(n)}$

arbitrary msg

$\ell(n)$-bit output

public

H^s

$n^5 = \ell'(n)$

$H^s: \{0,1\}^{\ell'(n)} \rightarrow \{0,1\}^{\ell(n)}$

$(\ell'(n) > \ell(n))$

compression fn

$H^s: \{0,1\}^{n^3} \rightarrow \{0,1\}^n$

$\Pi = (Gen, H)$

$Hash\text{-}coll_{A,\Pi}(n)$

1^n participants

A: adversary

$s \leftarrow Gen(1^n)$

s

Key is given A

$H^s(\cdot)$

generates x, x'

Adversary wins · he/she collision

1	$H^s(x) = H^s(x')$
0	otherwise

coll-resis

For all PPT A

$$Pr[Hash\text{-}coll_{A,\Pi}(n) = 1] \leq negl(n)$$

Real hash function

SHA-1, SHA-2, SHA-3] Look up

No keys

- creates technical issues.

Collisions are hard to find

H^s H

$SHA\text{-}1(x) = SHA\text{-}1(x')$

x, x'

- 2^{nd}-preimage resistance (2^{nd}-preimage) ← short

A: given: s, x (*)

output: x'

$H^s(x) = H^s(x')$ ← game

- preimage resistance (preimage). ← short

A: given: s, y (**)

output: x'

$H^s(x') = y$

$1010\ldots111 = H^s(x')$

NOT ↓

$\neg(2^{nd}\text{-preimage}) \Rightarrow \neg(\text{coll-resis})$

- A solves (*)
- sample x, use A to find x' s.t $H^s(x) = H^s(x')$ collision !

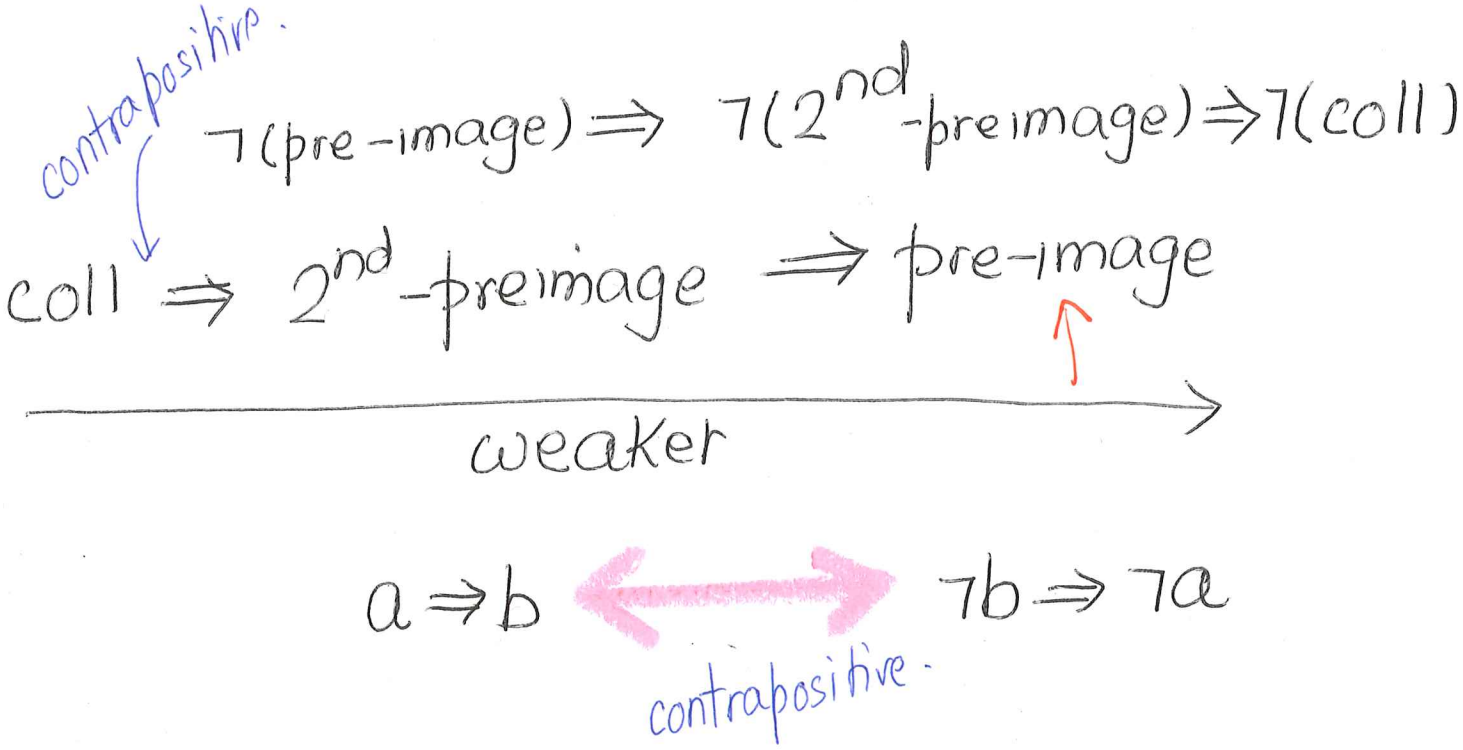
$\neg(\text{preimage}) \Rightarrow \neg(2^{nd}\text{-preimage})$

- A solves (**)
- s, x use A with $H^s(x)$ to find x' s.t $H^s(x') = y \equiv H^s(x)$

$y = H^s(x)$

$H^s(x) = H^s(x')$

~~$\neg(\text{pre-image}) \Rightarrow \neg$~~



$a \Rightarrow b \iff \neg b \Rightarrow \neg a$

contrapositive.

Oct 22, 2020

~~2021~~ 2020

1

LectureLet 21

Last time

- Hash functions
  - definition
  - property

weaker ↓

- coll-resistance
- 2nd-preimage resistance
- preimage resistance

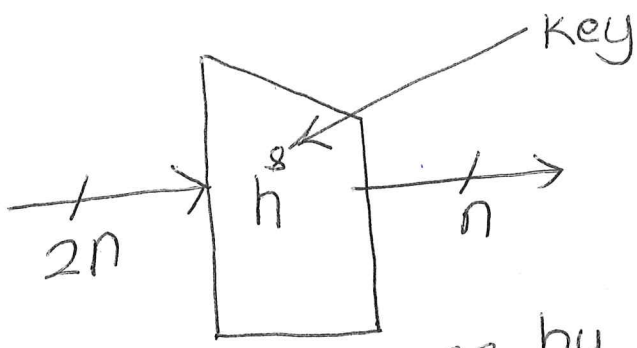
Logistics

- HW ~~4~~ 3 due soon (

Domain Extension for hash functions

compresses "little" → compresses "a lot"

Merkle-Damgard construction



compresses by 1/2

each block is n-bits

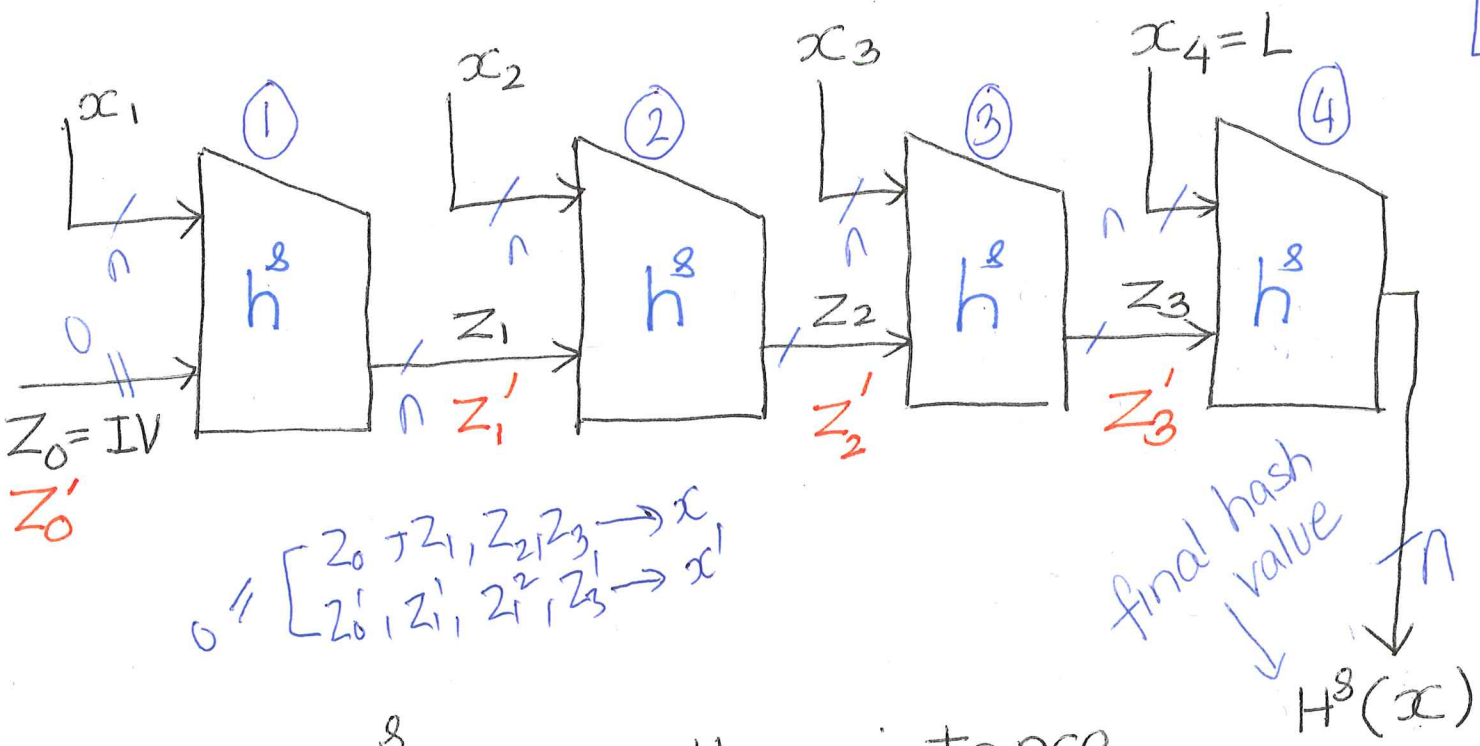
use

$x = \underbrace{x_1 \; x_2 \; x_3}_{3n\text{-bits}}$

L = len. of x (Last block x_4)

$3n \longrightarrow n$

compress by 1/3



Assume: h^s is coll-resistance

To prove: $H^s(x)$ is coll-resistance

Proof by contradiction

$H^s(x) = H^s(x')$ $x \neq x'$

$x_1\ x_2\ x_3\ x_4$ (= L (len of x))

$x_1'\ x_2'\ x_3'\ x_4'$ (= L' (len of x'))

$L \neq L'$

$h^s(L \| Z_3) = h^s(L' \| Z_3')$ (concat)

$\neq$ collision in h^s ✓

$L = L'$, ~~$x_3 \neq x_3'$~~
$Z_3 \neq Z_3'$

$$h^s(L \| Z_3) = h^s(L' \| Z_3')$$

($h^s(L \| Z_3)$ = $H^s(x)$, $h^s(L' \| Z_3')$ = $H^s(x')$)

$\neq$ collision in h^s

$L = L'$, $Z_3 = Z_3'$ → $x_3 \neq x_3'$ – 3rd block ③

$$Z_3 = h^s(x_3 \| Z_2) = h^s(x_3' \| Z_2') = Z_3'$$

$\neq$ collision in h^s

$x_3 = x_3'$, $Z_2 \neq Z_2'$

$$h^s(x_3 \| Z_2) = h^s(x_3' \| Z_2')$$

(Z_2 = Z_2')

$\neq$ collision in h^s

$L = L'$, $Z_3 = Z_3'$, $x_3 = x_3'$, $Z_2 = Z_2'$

Box ②

collision

$x_2 \neq x_2'$

$x_2 = x_2'$, $Z_1 \neq Z_1'$

$L = L'$, $Z_3 = Z_3'$, $x_3 = x_3'$, $Z_2 = Z_2'$, $x_2 = x_2'$, $Z_1 = Z'$

NOT POSSBLE.

$x_1 \neq x_1'$

$x_1 = x_1'$, $Z_0 \neq Z_0'$

if no collision found,

$\underbrace{x_1 = x_1',\ x_2 = x_2',\ x_3 = x_3'}_{\text{msgs were same } x=x'},\ L = L'$

contradiction ✗

if we use ℓ-block/stages

$\ell = 7$ (1/6)

$6n \longrightarrow n$

$\ell = 5$ construction repeat proof

$(\ell - 1)n \longrightarrow n$

$\ell = 4$ blocks/stages in example

Hash-and-Mac

MAC: $(Mac, Vrfy)$ — Gen

works for messages of length $\ell(n)$ — size of the key — $8n$

(ex: $\ell(n) = n^3$)

$n = 100$

$800 = 8n$

Hash: (Gen_H, H)

$H^s: \{0,1\}^* \longrightarrow \ell(n)$

arbitrary

(ex: $\ell(n) = 8n$)

$m \in \{0,1\}^*$ $\quad |m| > \ell(n)$

len $l(n)$

$MAC_K(H^s(m)) = t$ (tag)

$H^s(m)$: $\ell(n)$ (ex: $8n$)

Hash-then-Mac

$k \leftarrow Gen(1^n)$ mac-forge

Hash-and-Mac $= MAC'_k(\ )$ — oracle

$Q = \{m_1, \ldots, m_k\}$

$t_1 \ldots t_k$

$mac_k(H^s(m_1))$

$mac_k(H(m^*)) = mac_k(H(m_2))$

$H(m^*) = H(m_2)$

(m^*, t_2)

collision-resistant

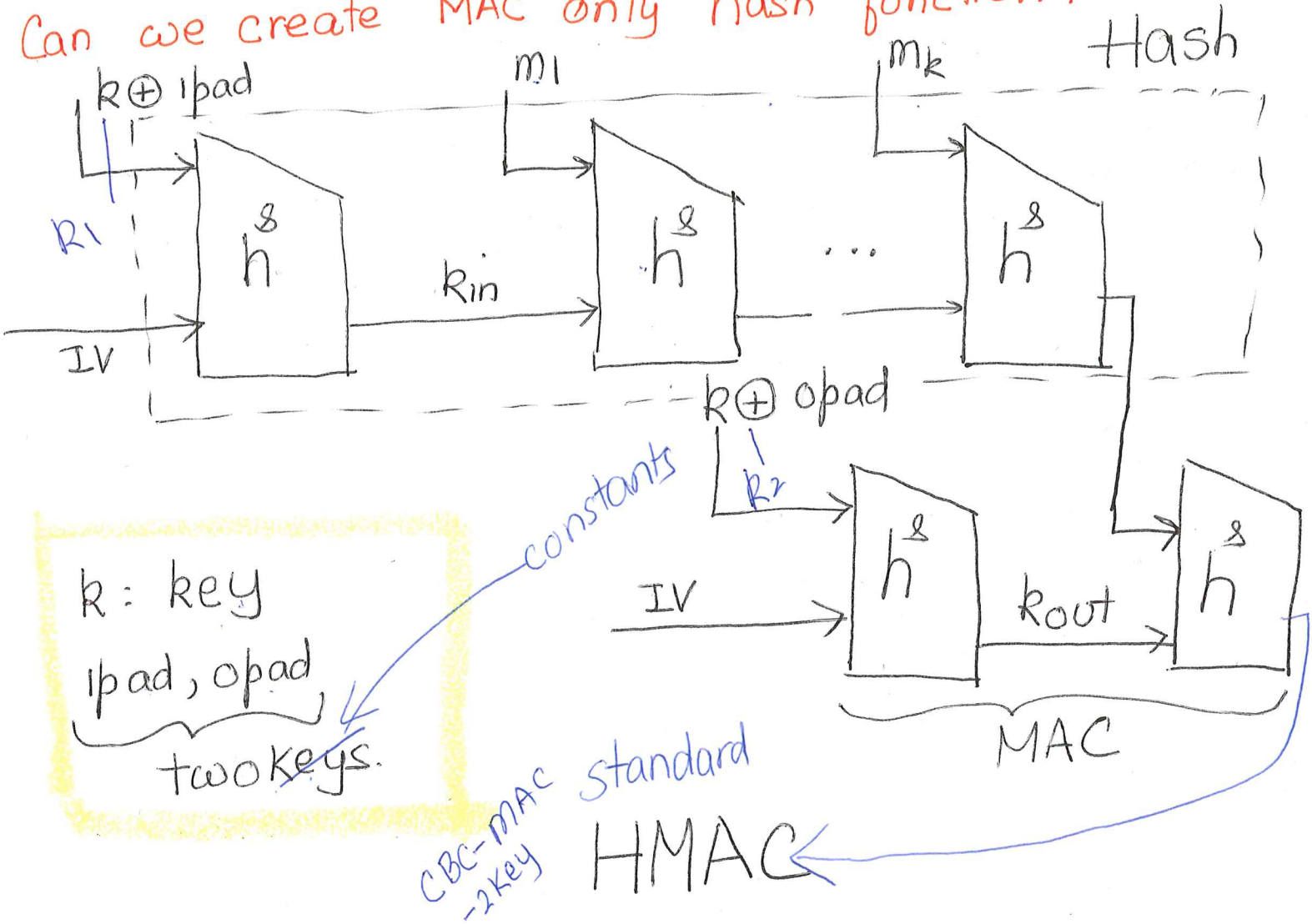
(m^*, t_i)

$(m^*, t) \leftarrow$ forgery

- $H(m^*) = H(m_i)$ collision
- $H(m^*) \neq H(m_i)$ $1 \le i \le k$

valid forgery

Can we create MAC only hash function?



k : key

ipad, opad

two keys.

constants

CBC-MAC standard

-2key

HMAC

Attack on hash function

SHA-1

$H: \{0,1\}^* \longrightarrow \{0,1\}^n$ $H(x) \sim$ random

PRF

use RO-model

# of n-bit strings

n-bit string

2^n+1

$(x, x+1, \ldots, x+2^n)$

$H(x), H(x+1), \ldots H(x+2^n)$ ← 2^n+1

2^n-possible values

2 of them have same value

$H(x+i) = H(x+j)$

Collision!

$n = 256$

Uses time $O(2^n)$ $O(2^{256})$

Can we do better?

YES Birthday paradox!

30 people at a party. What is the prob. two people will have the same birthday

(I) $< 50\%$

(II) $\geq 50\%$