COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Fall 2020

Announcements/Reminders

Last day to submit PR1

- Submit archive file on Canvas

HW3 due on Oct 20

Previously on COS 433…

Constructing MACs

Use a PRF

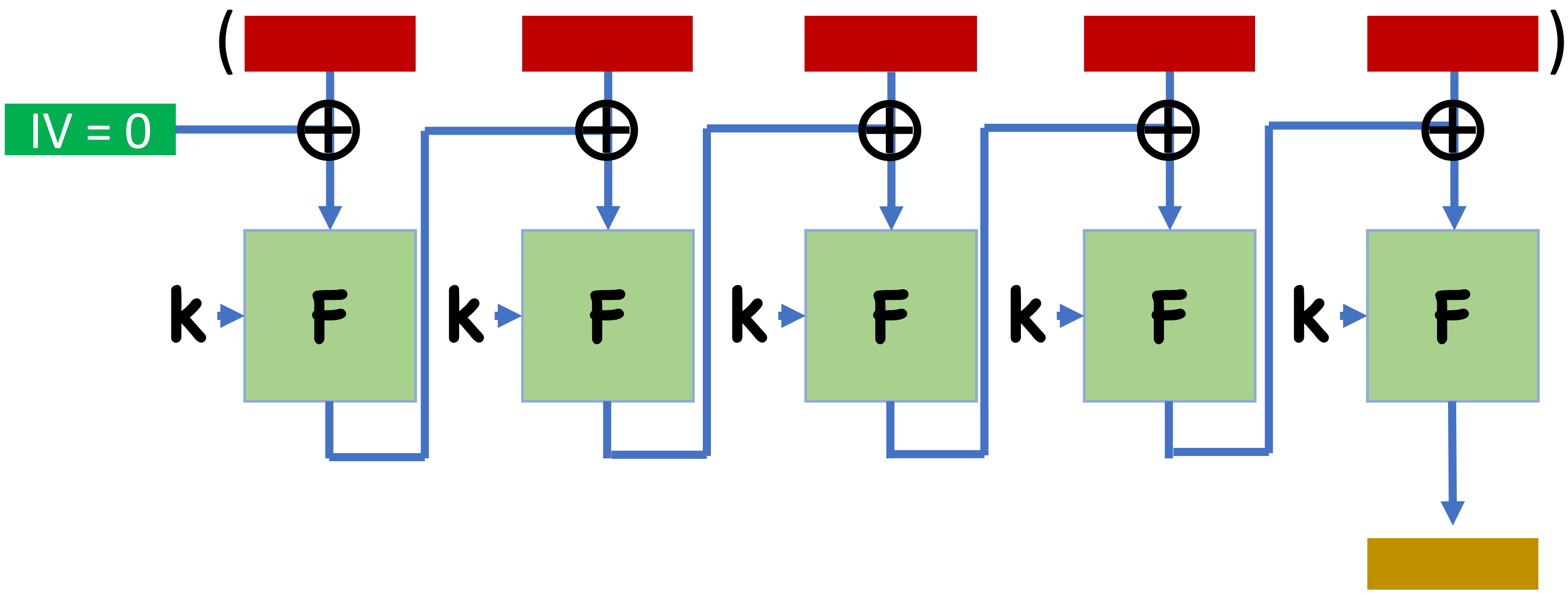
$F:K_\lambda \times M_\lambda \rightarrow T_\lambda$

$MAC(k,m) = F(k,m)$

$Ver(k,m,\sigma) = (F(k,m) == \sigma)$

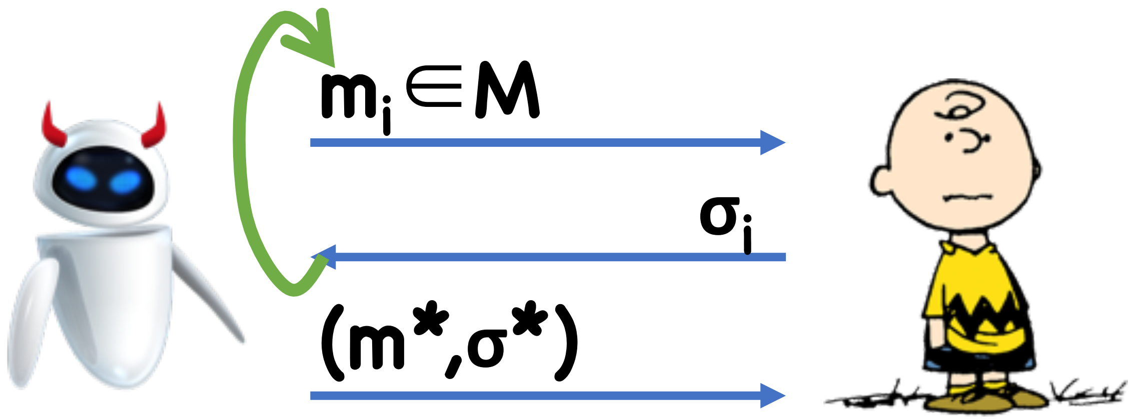
Theorem: If F is a secure PRF and $|\mathsf{T}_\lambda|$ is super-polynomial, then $(\mathsf{MAC},\mathsf{Ver})$ is CMA secure

CBC-MAC



Theorem: CBC-MAC is a secure PRF for **fixed-length** messages

Strongly Secure MACs



$k \leftarrow K$

$\sigma \leftarrow MAC(k,m_i)$

Output 1 iff:

- $(m^*,\sigma^*) \notin \{(m_1,\sigma_1),\ldots\}$
- $Ver(k,m^*,\sigma^*) = 1$

SCMA-Adv() = Pr[ outputs 1]

Carter Wegman MAC

$k' = (k,h)$ where:

- **k** is a PRF key for **$F:K \times R \rightarrow Y$**
- **h** is sampled from a pairwise independent function family

MAC(k',m):

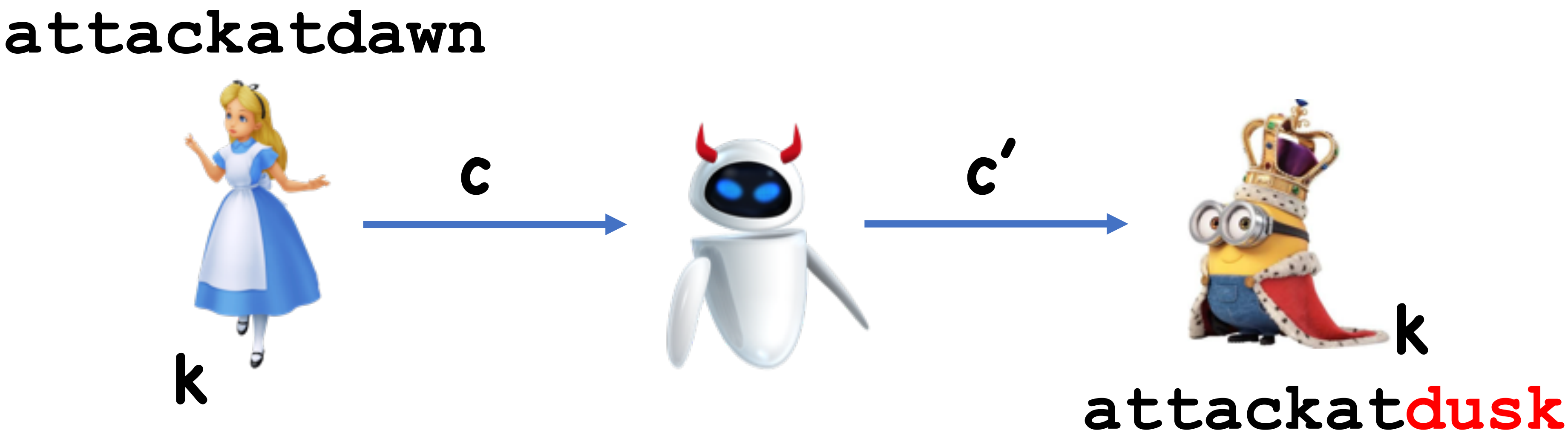
- Choose a random **$r \leftarrow R$**
- Set **$\sigma = (r, F(k,r) \oplus h(m))$**

Theorem: If $\mathbf{F}$ is secure and $\mathbf{|T|,|R|}$ are super-polynomial, then the Carter Wegman MAC is strongly CMA secure

Today:

- Authenticated Encryption, CCA security
- Hash functions

Authenticated Encryption



Goal: Eve cannot learn nor change plaintext

- Authenticated Encryption will satisfy two security properties

Syntax

Syntax:

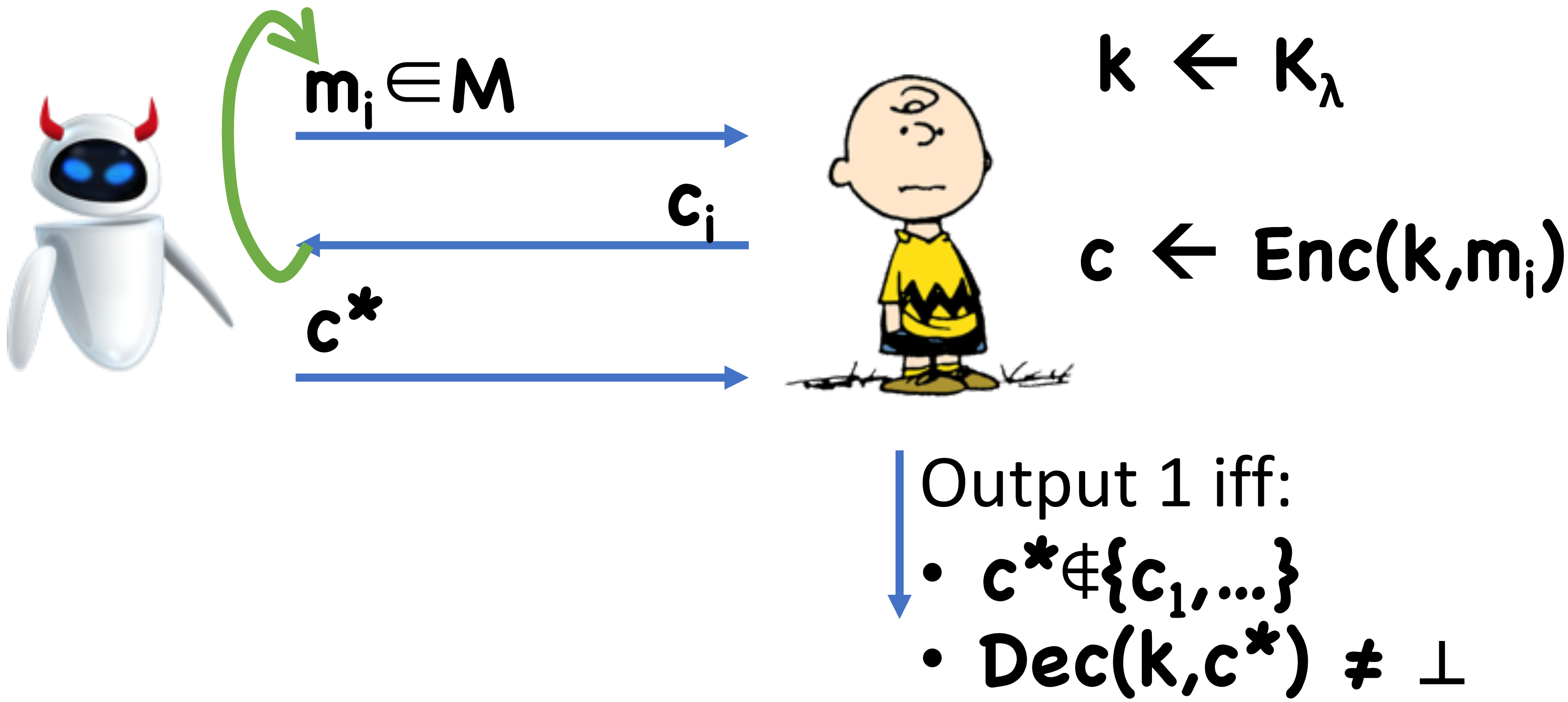
- $\mathbf{Enc}: K \times M \rightarrow C$
- $\mathbf{Dec}: K \times C \rightarrow M \cup \{\perp\}$

Correctness:

- For all $k \in K$, $m \in M$,

$$\Pr[\mathrm{Dec}(k, \mathrm{Enc}(k,m)) = m] = 1$$

Unforgeability

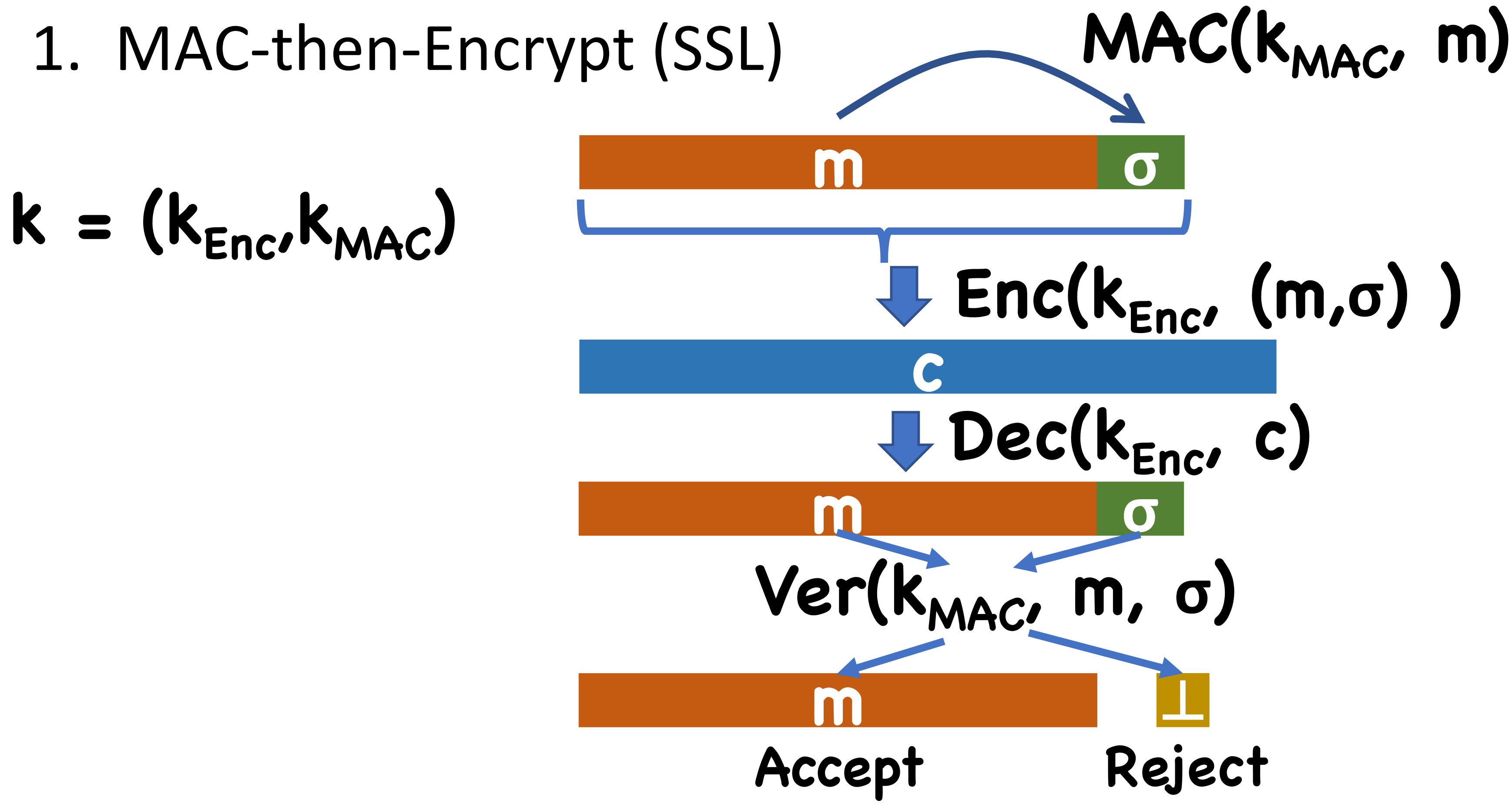


Definition: An encryption scheme **(Enc,Dec)** is an **authenticated encryption scheme** if it is unforgeable and CPA secure

Constructing Authenticated Encryption

Three possible generic constructions:

1. MAC-then-Encrypt (SSL)

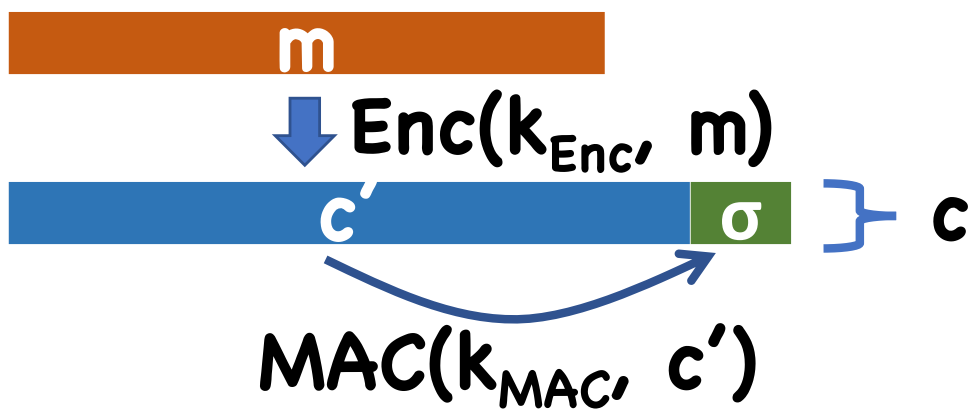


Constructing Authenticated Encryption

Three possible generic constructions:

2. Encrypt-then-MAC (IPsec)

$$k = (k_{Enc}, k_{MAC})$$

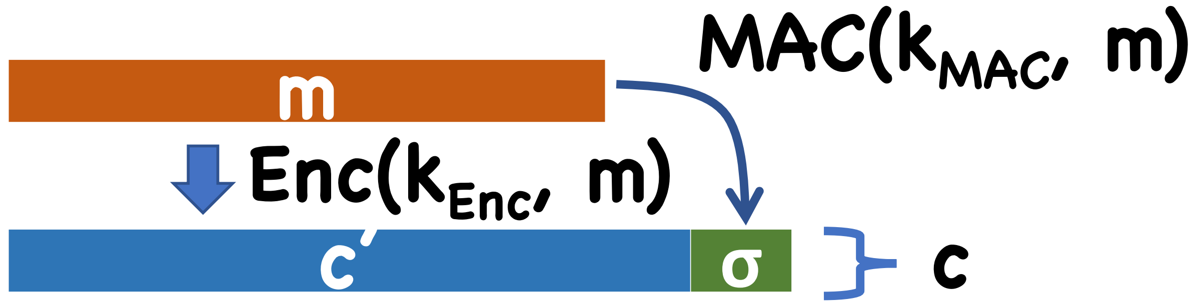


Constructing Authenticated Encryption

Three possible generic constructions:

3. Encrypt-and-MAC (SSH)

$k = (k_{Enc}, k_{MAC})$



Constructing Authenticated Encryption

1. MAC-then-Encrypt

2. Encrypt-then-MAC

3. Encrypt-and-MAC

Which one(s) **always** provides authenticated encryption (assuming strongly secure MAC)?

Constructing Authenticated Encryption

MAC-then-Encrypt?

- Encryption not guaranteed to provide authentication
- May be able to modify ciphertext to create a new ciphertext
- Toy example: **Enc(k,m) = (0,Enc'(k,m))**
  Dec(k, (b,c)) = Dec'(k,c)

Constructing Authenticated Encryption

Encrypt-then-MAC?

- Inner encryption scheme guarantees secrecy, regardless of what MAC does
- (strongly secure) MAC provides integrity, regardless of what encryption scheme does

Theorem: Encrypt-then-MAC is an authenticated encryption scheme for any CPA-secure encryption scheme and *strongly* CMA-secure MAC

✓

Constructing Authenticated Encryption

Encrypt-and-MAC?

- MAC not guaranteed to provide secrecy
- Even though message is encrypted, MAC may reveal info about message
- Toy example: **MAC(k,m) = (m,MAC'(k,m))**

Constructing Authenticated Encryption

1. MAC-then-Encrypt ✘
2. Encrypt-then-MAC ✓
3. Encrypt-and-MAC ✘

Which one(s) **always** provides authenticated encryption (assuming strongly secure MAC)?

Constructing Authenticated Encryption

Just because MAC-then-Encrypt and Encrypt-and-MAC are insecure for *some* MACs/encryption schemes, they may be secure in some settings

Ex: MAC-then-Encrypt with CTR or CBC encryption
- For CTR, any one-time MAC is actually sufficient

Theorem: MAC-then-Encrypt with any one-time MAC and CTR-mode encryption is an authenticated encryption scheme

Chosen Ciphertext Attacks

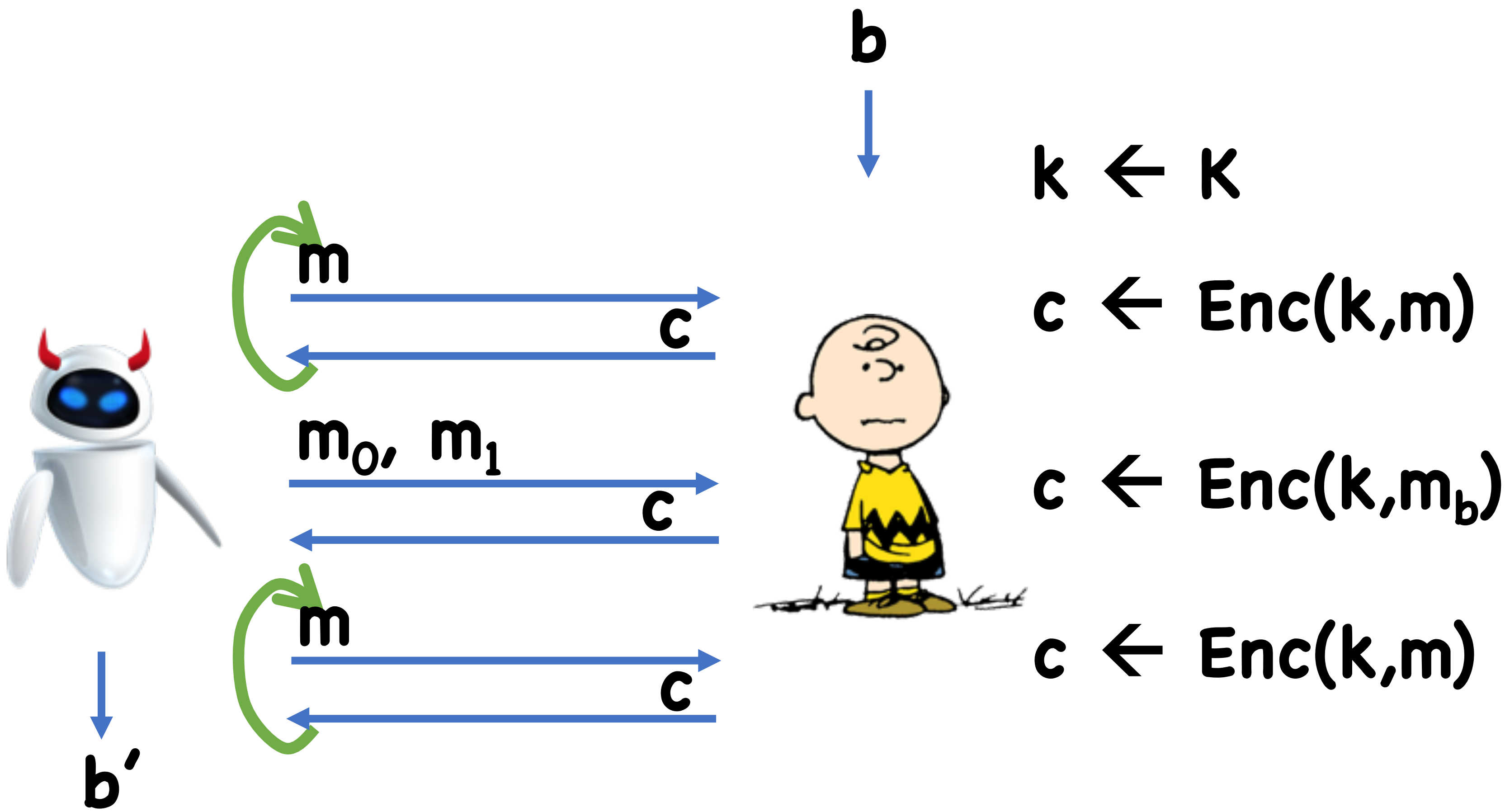
Chosen Ciphertext Attacks

Often, adversary can fool server into decrypting certain ciphertexts

Even if adversary only learns partial information (e.g. whether ciphertext decrypted successfully), can use info to decrypt entire message

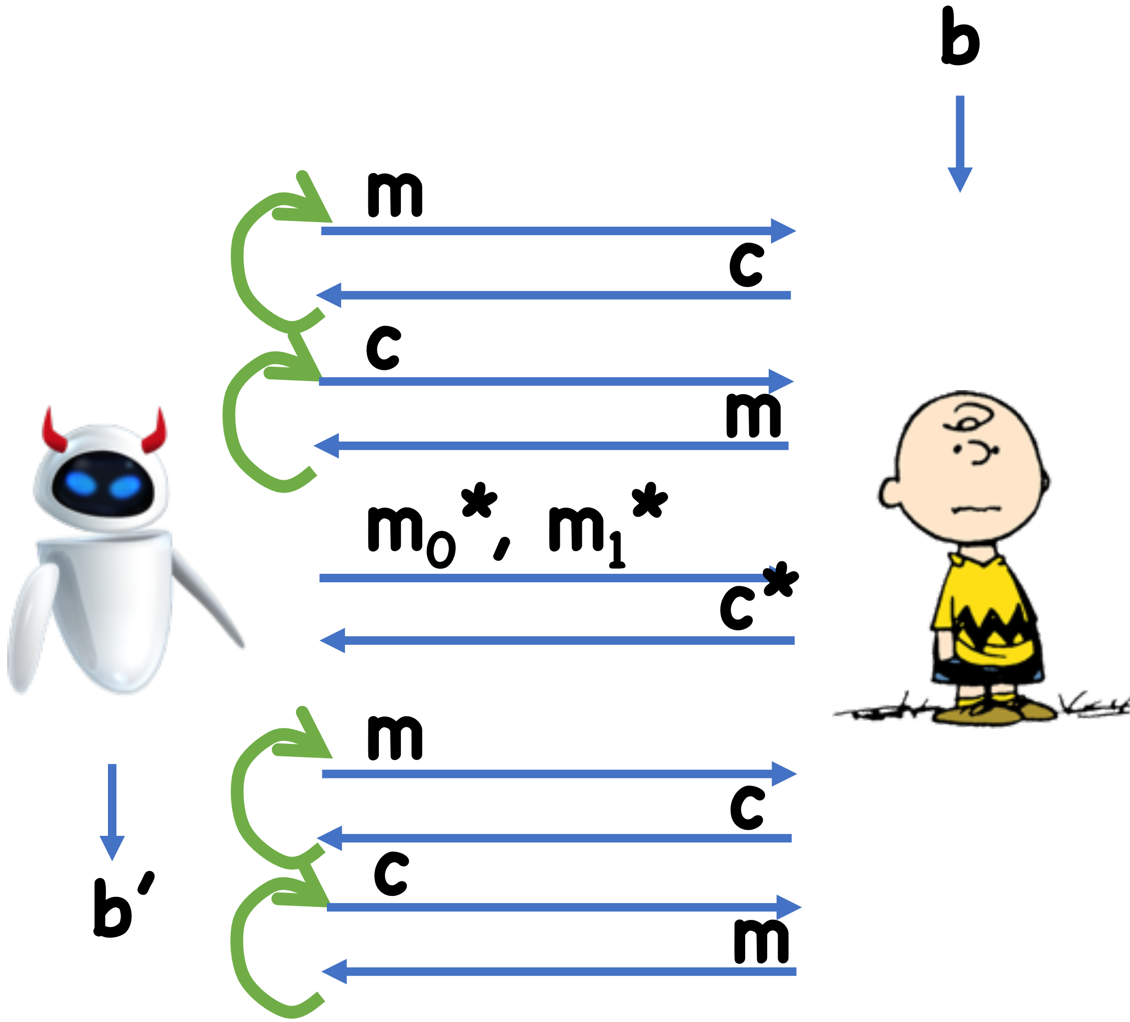
Therefore, want security even if adversary can mount decryption queries

Chosen Plaintext Security



CPA-Exp$_b$( **, λ)**

Chosen Ciphertext Security?



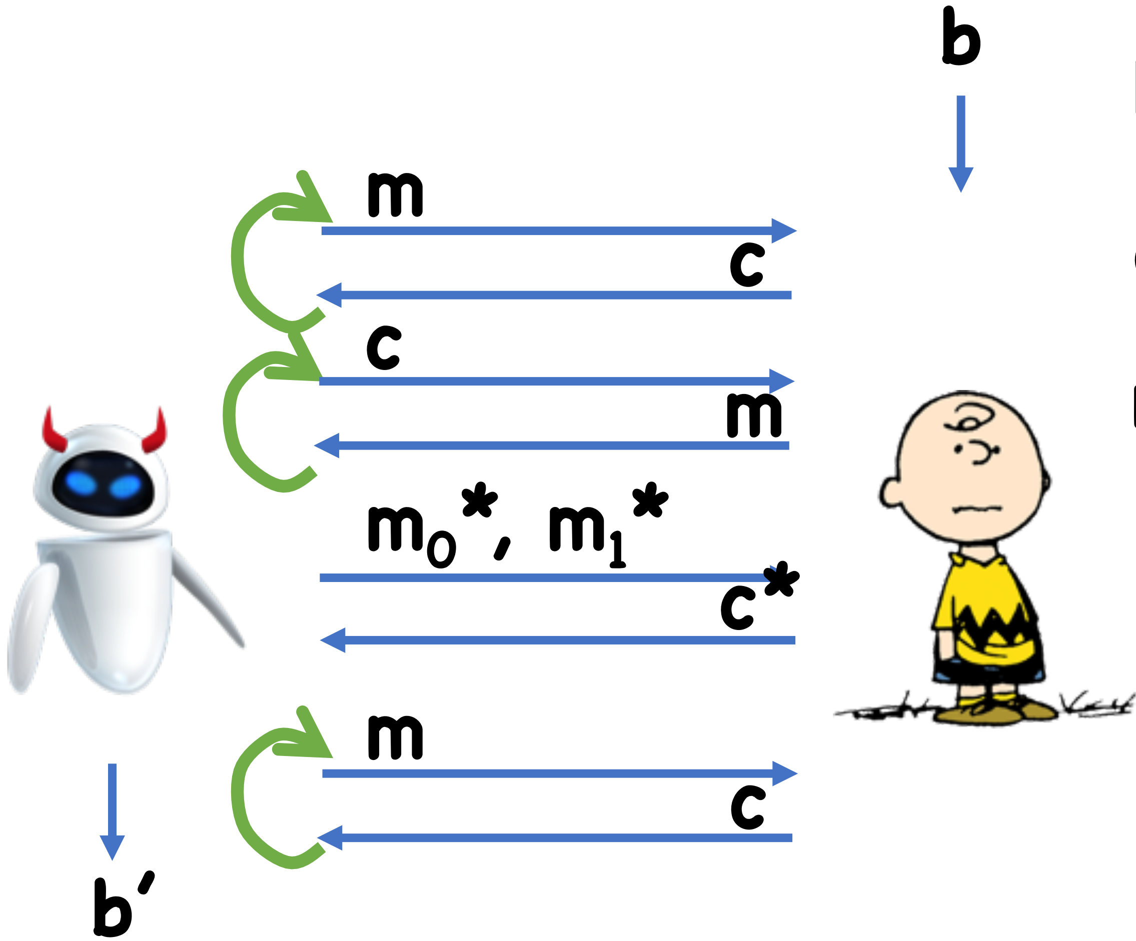
$k \leftarrow K$

$c \leftarrow \mathrm{Enc}(k,m)$

$m \leftarrow \mathrm{Dec}(k,c)$

$c^* \leftarrow \mathrm{Enc}(k,m_b^*)$

Lunch-time CCA (CCA1)



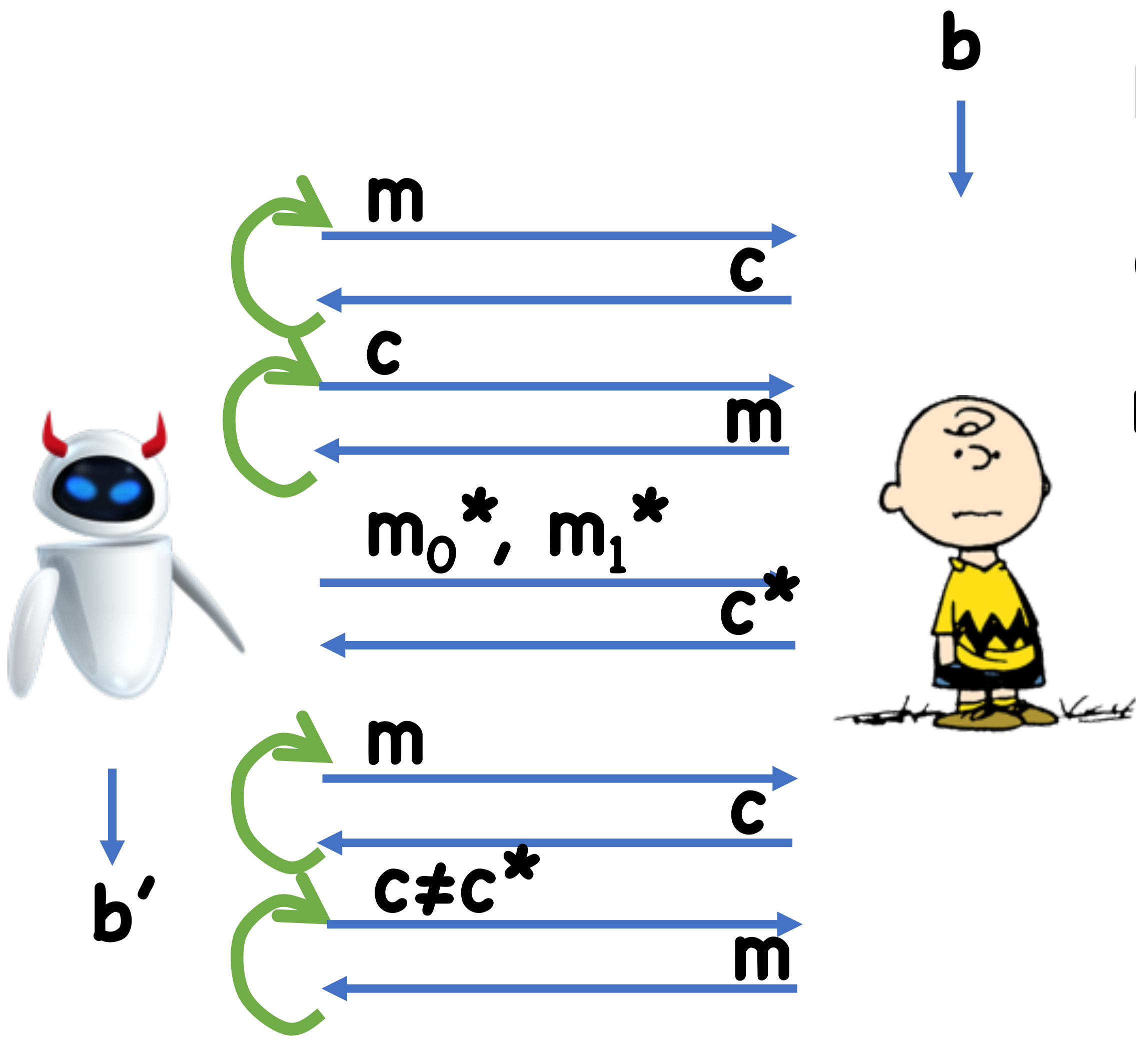
$k \leftarrow K$

$c \leftarrow \mathrm{Enc}(k,m)$

$m \leftarrow \mathrm{Dec}(k,c)$

$c^* \leftarrow \mathrm{Enc}(k,m_b^*)$

Full CCA (CCA2)



$k \leftarrow K$

$c \leftarrow Enc(k,m)$

$m \leftarrow Dec(k,c)$

$c^* \leftarrow Enc(k,m_b^*)$

Theorem: If **(Enc,Dec)** is an authenticated encryption scheme, then it is also CCA secure

Proof Sketch

For any decryption query, two cases

1. Was the result of a CPA query
- In this case, we know the answer already!

2. Was not the result of an encryption query
- In this case, we have a ciphertext forgery

Collision Resistant Hashing

Expanding Message Length for MACs

Suppose we have a MAC **(MAC,Ver)** that works for small messages (e.g. 256 bits)

How can I build a MAC that works for large messages?

One approach:
- MAC blockwise + extra steps to insure integrity
- Problem: extremely long tags

Hash Functions

Let $h:\{0,1\}^l \rightarrow \{0,1\}^n$ be a function, $n << l$

$MAC'(k,m) = MAC(k,\ h(m))$
$Ver'(k,m,\sigma) = Ver(k,\ h(m),\ \sigma)$

Correctness is straightforward

Security?

- Pigeonhole principle: $\exists\, m_0 \neq m_1$ s.t. $h(m_0)=h(m_1)$
- But, hopefully such collisions are hard to find

Collision Resistant Hashing?

Syntax:

- Domain $\mathbf{D}$ (typically $\{0,1\}^l$ or $\{0,1\}^*$)
- Range $\mathbf{R}$ (typically $\{0,1\}^n$)
- Function $\mathbf{H}: \mathbf{D} \rightarrow \mathbf{R}$

Correctness: $\mathbf{n} \ll \mathbf{l}$

Security?

Definition: H is collision resistant if, for all  running in polynomial time, $\exists$ negligible **ε** such that:

$\Pr[H(x_0) = H(x_1) \wedge x_0 \neq x_1 : (x_0, x_1) \leftarrow$  $()] < \varepsilon(\lambda)$

Problem?

Theory vs Practice

In practice, the existence of an algorithm with a built in collision isn't much of a concern

- Collisions are hard to find, after all

However, it presents a problem with our definitions

- So theorists change the definition
- Alternate def. will also be useful later

Collision Resistant Hashing

Syntax:

- Key space **K** (typically $\{0,1\}^\lambda$)
- Domain **D** (typically $\{0,1\}^l$ or $\{0,1\}^*$)
- Range **R** (typically $\{0,1\}^n$)
- Function $H: K \times D \rightarrow R$

Correctness: $n \ll l$

Security

Definition: H is collision resistant if, for all  running in polynomial time, $\exists$ negligible **ε** such that:

$$\Pr[H(k,x_0) = H(k,x_1) \wedge x_0 \neq x_1 :$$
$(x_0,x_1) \leftarrow$  $(k), k \leftarrow K] < \varepsilon(\lambda)$

Collision Resistance and MACs

Let $\mathbf{h(m) = H(k,m)}$ for a random choice of $\mathbf{k}$

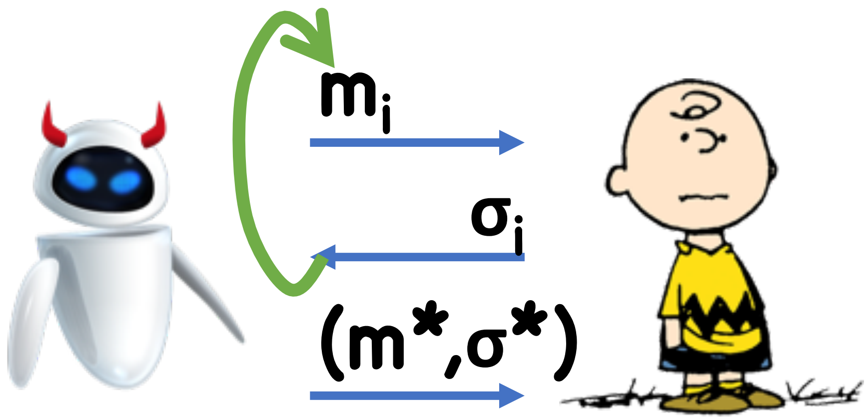
$$\mathrm{MAC}'(k_{MAC},m) = \mathrm{MAC}(k_{MAC},\ h(m))$$
$$\mathrm{Ver}'(k_{MAC},m,\sigma) = \mathrm{Ver}(k_{MAC},\ h(m),\ \sigma)$$

Think of $\mathbf{k}$ as part of key for $\mathbf{MAC'}$

Theorem: If **(MAC,Ver)** is CMA-secure and **H** is collision resistant, then **(MAC',Ver')** is CMA secure

Proof

<u>Hybrid 0</u>



$k_H \leftarrow K_H$

$k_{MAC} \leftarrow K_{MAC}$

$t_i \leftarrow H(k_H,m_i)$

$\sigma \leftarrow MAC(k_{MAC}, t_i)$

Output 1 iff:

- $m^* \notin \{m_1,\ldots\}$
- $Ver(k,t^*,\sigma^*)$ where

  $t^* \leftarrow H(k_H,m^*)$

Proof

<u>Hybrid 1</u>

$k_H \leftarrow K_H$

$k_{MAC} \leftarrow K_{MAC}$

m_i

σ_i

(m^*,σ^*)

$t_i \leftarrow H(k_H,m_i)$

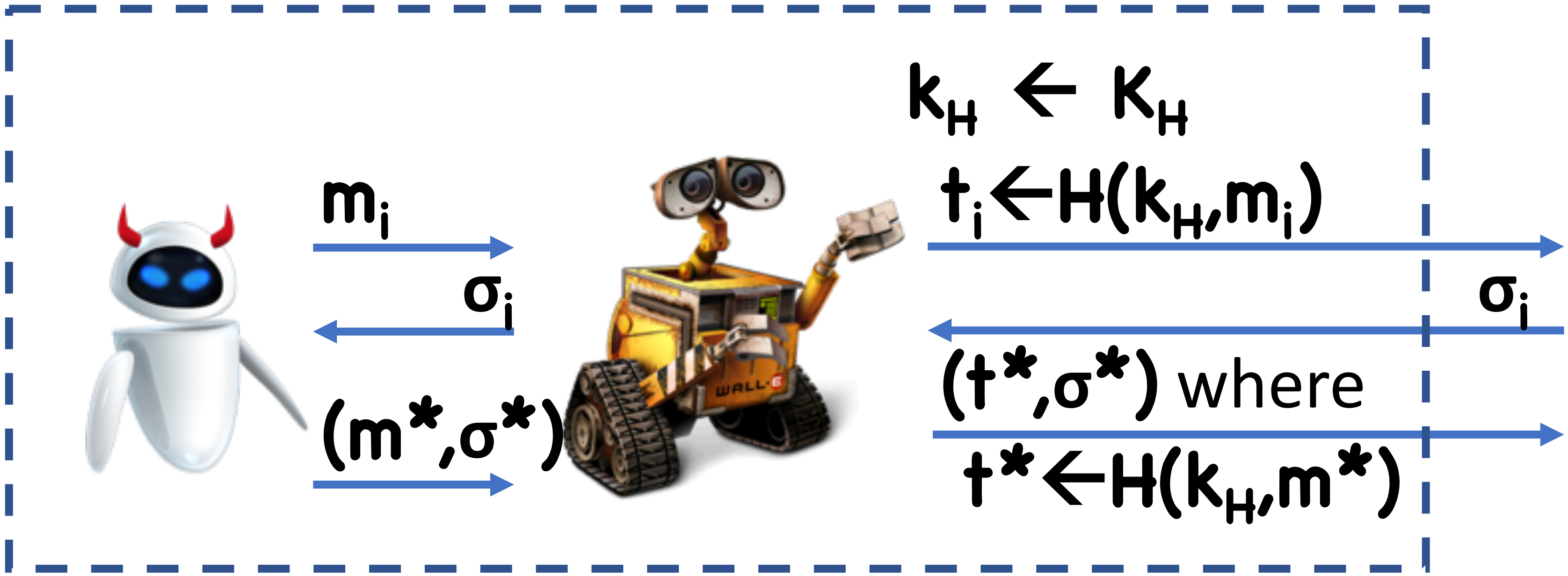
$\sigma \leftarrow MAC(k_{MAC}, t_i)$

Output 1 iff:

- $t^* \notin \{t_1,\ldots\}$
- $Ver(k,t^*,\sigma^*)$ where
  $t^* \leftarrow H(k_H,m^*)$

Proof

In Hybrid 1, negligible advantage using MAC security



If  forges with $t^* \notin \{t_1,\ldots\}$, then  also forges

Proof

If  succeeds in Hybrid 0 but not Hybrid 1, then

- $\mathbf{m^*} \notin \{\mathbf{m_1,\ldots}\}$
- But, $\mathbf{t^*} \in \{\mathbf{t_1,\ldots}\}$

Suppose $\mathbf{t^* = t_i}$

Then $\mathbf{(m_i,m^*)}$ is a collision for $\mathbf{H(k,\ \cdot\ )}$

- Straightforward to construct collision finder

Constructing Hash Functions

Domain Extension

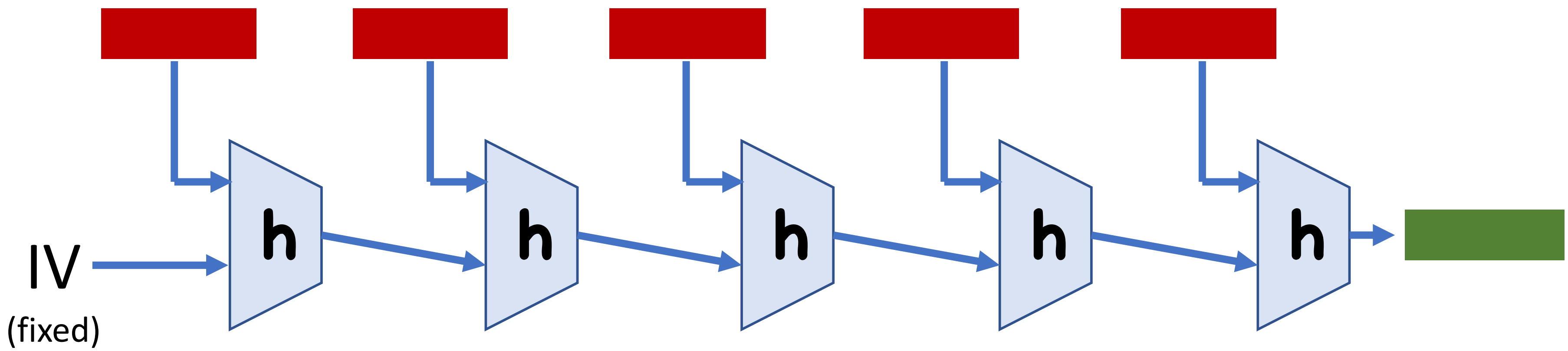
Goal: given **h** that compresses small inputs, construct **H** that compresses large inputs

Shows that even compressing by a single bit is enough to compress by arbitrarily many bits

Useful in practice: build hash functions for arbitrary inputs from hash functions with fixed input lengths

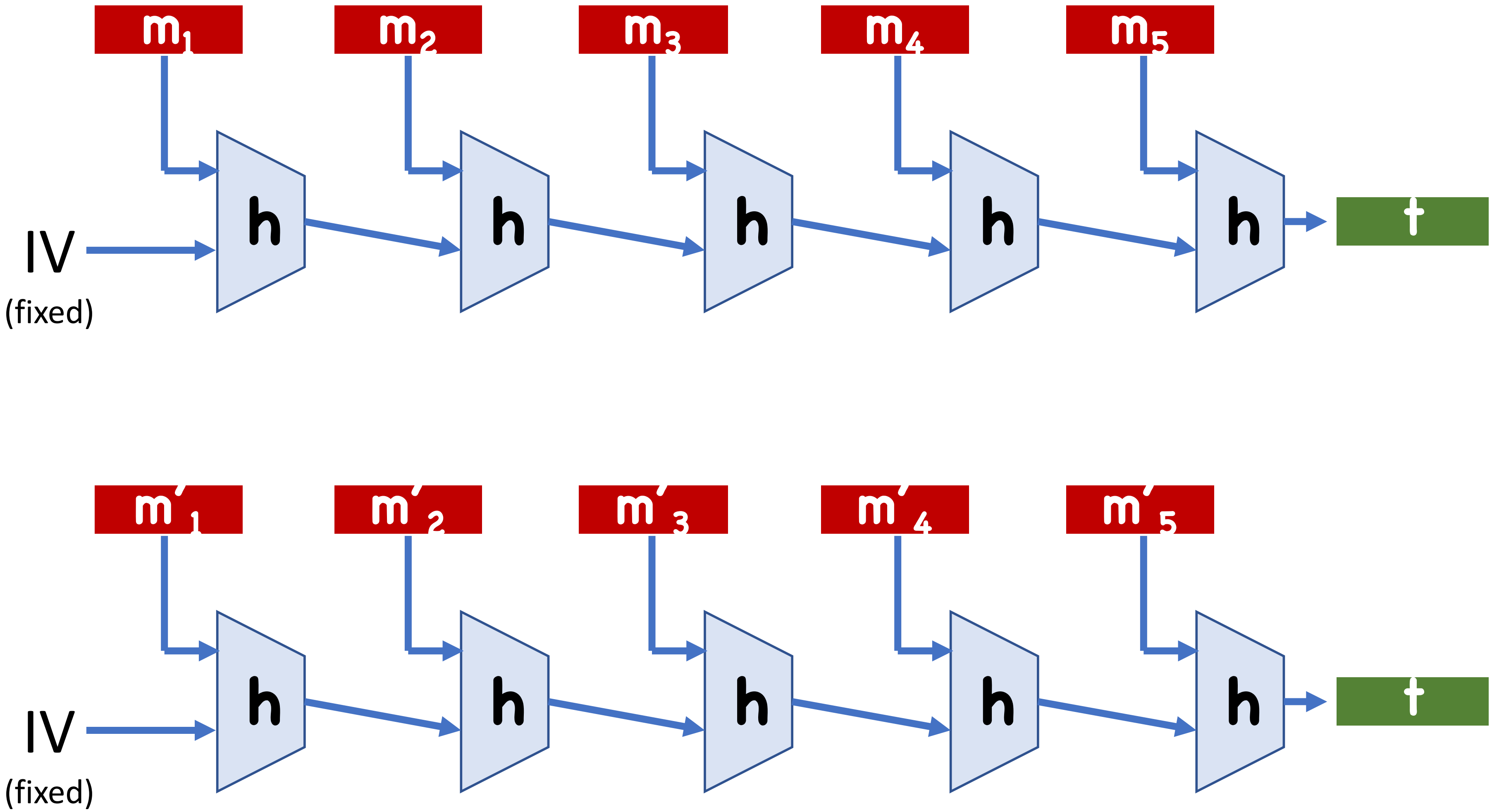
- Called compression functions
- Easier to design

Merkle-Damgard

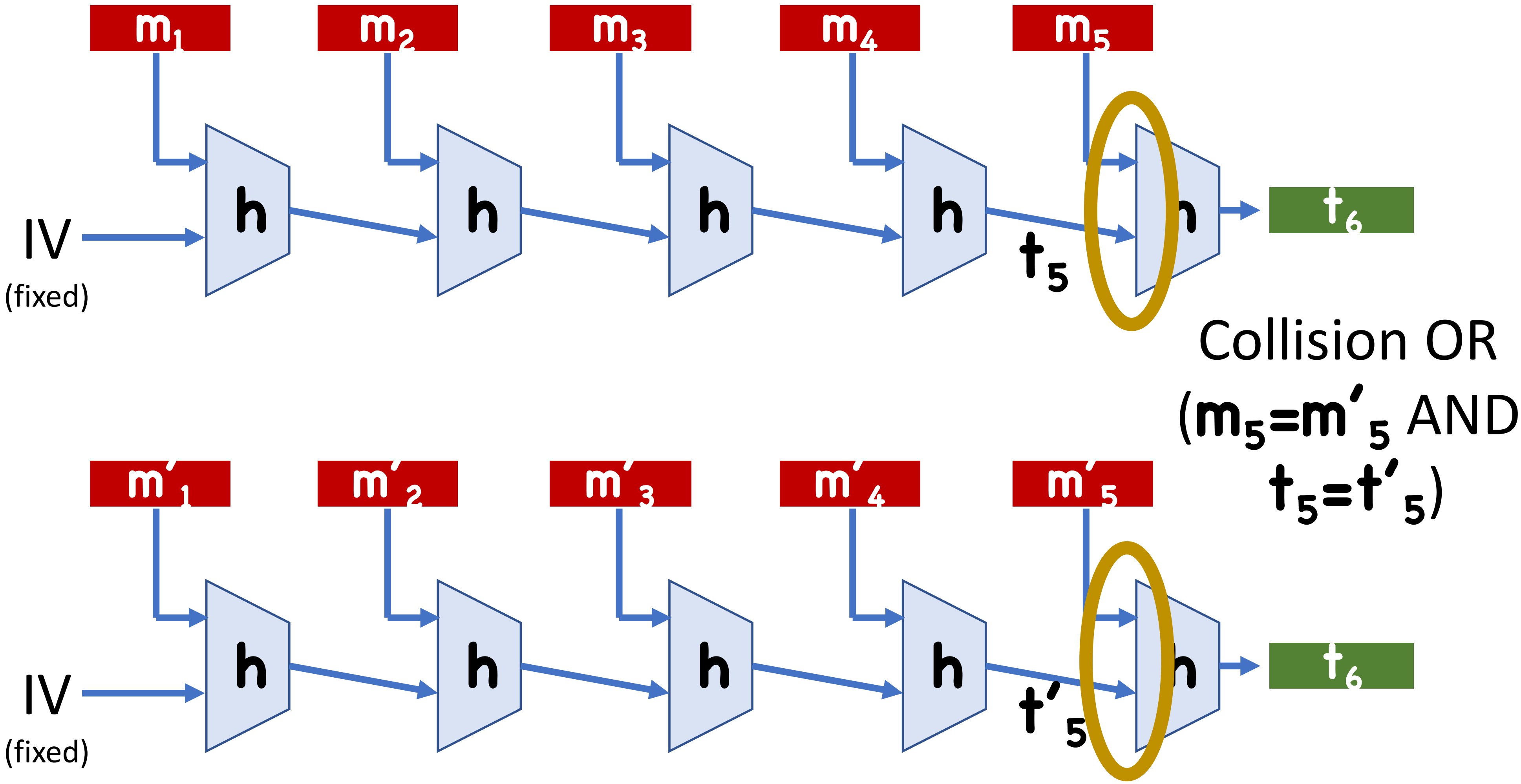


Theorem: If an adversary knows a collision for fixed-length Merkle-Damgard, it can also compute a collision for **h**

Proof

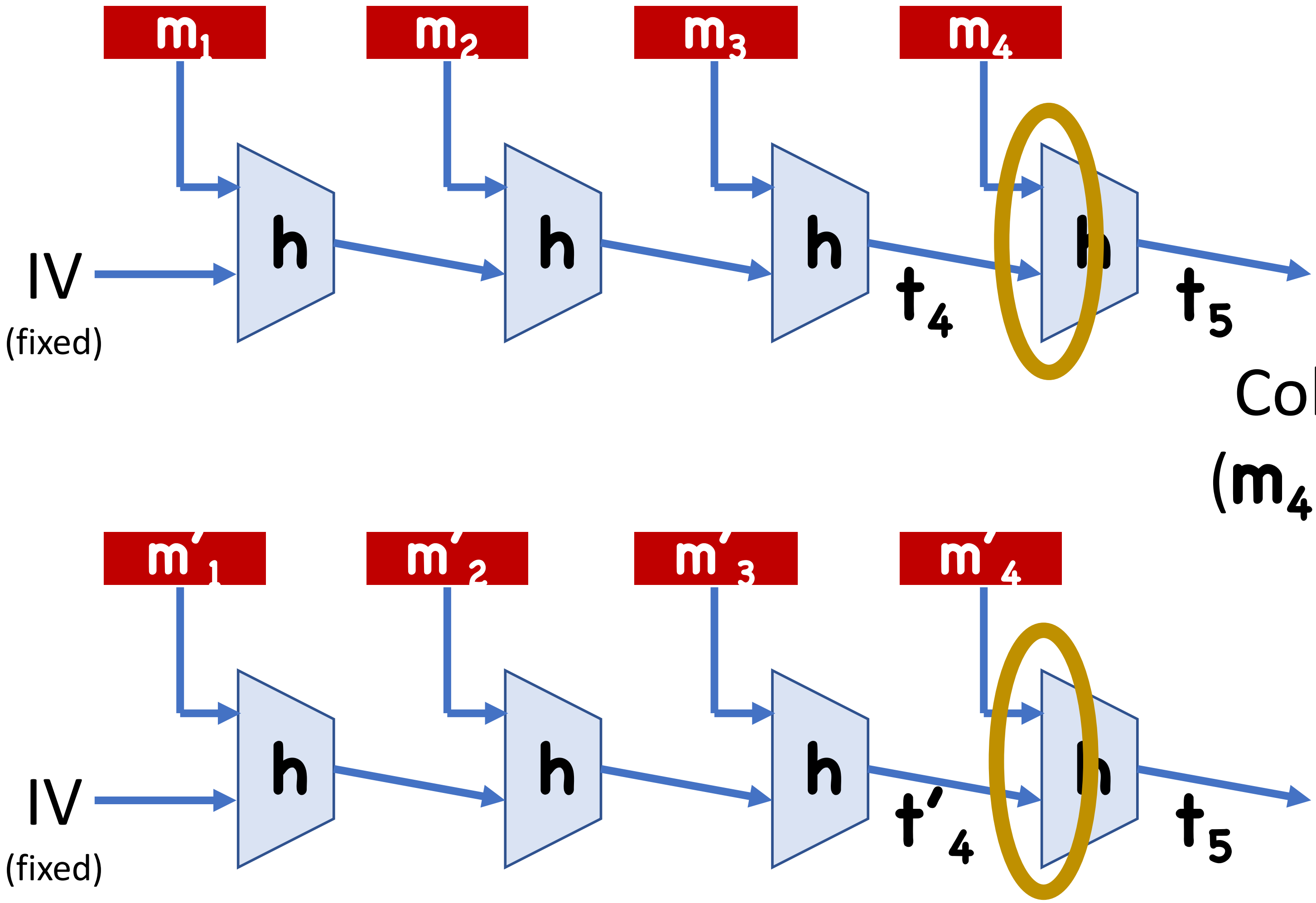


Proof



m_1
m_2
m_3
m_4
m_5
IV
(fixed)
h
h
h
h
h
t_5
t_6
Collision OR
(m_5=m'_5 AND
t_5=t'_5)
m'_1
m'_2
m'_3
m'_4
m'_5
IV
(fixed)
h
h
h
h
h
t'_5
t_6

Proof

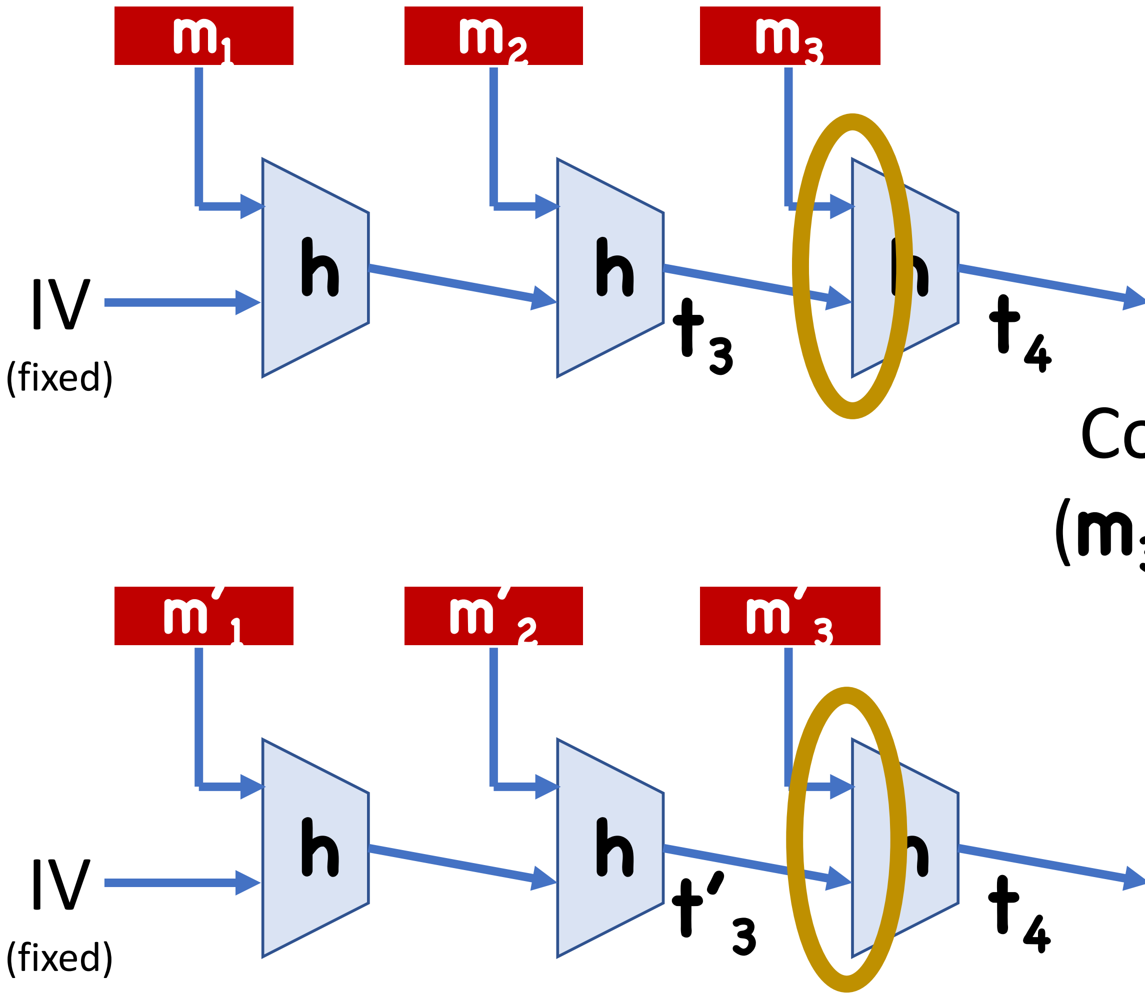


Collision OR

(m_4=m'_4 AND

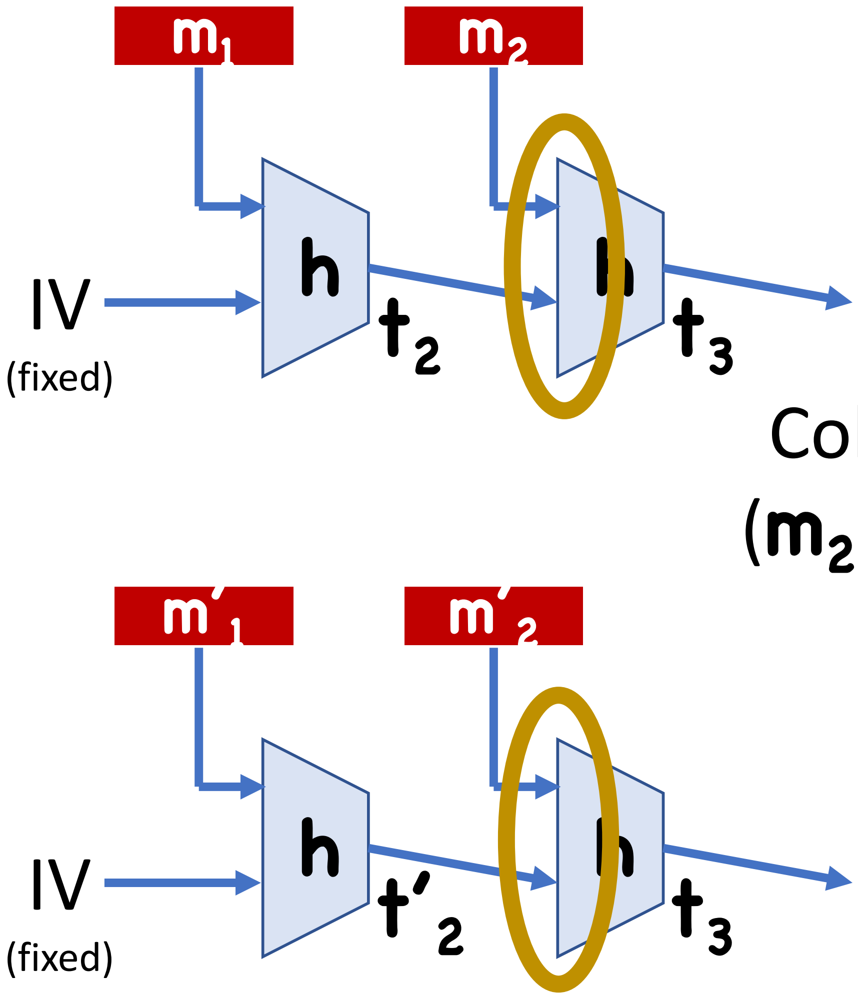
t_4=t'_4)

Proof



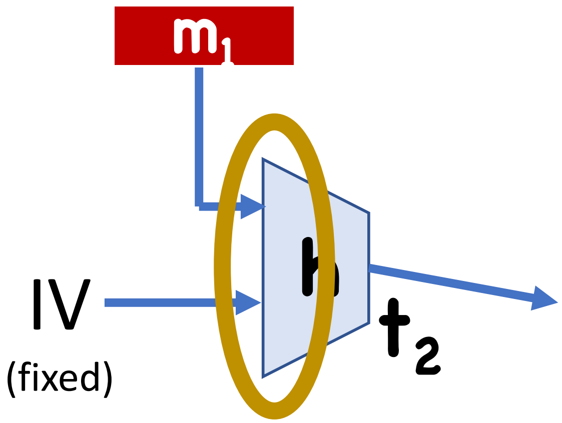
Collision OR
($m_3 = m'_3$ AND
$t_3 = t'_3$)

Proof

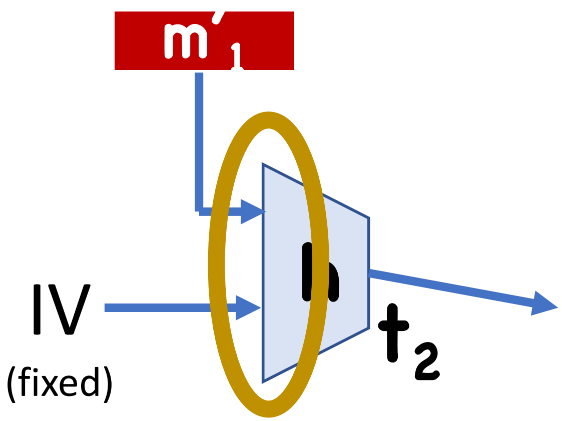


Collision OR
($m_2=m'_2$ AND
$t_2=t'_2$)

Proof



Collision OR
$\mathbf{m_1=m'_1}$



But, if $\mathbf{m_1=m'_1}$, then $\mathbf{m=m'}$

Merkle-Damgard

So far, assumed both inputs in collision has to have the same length

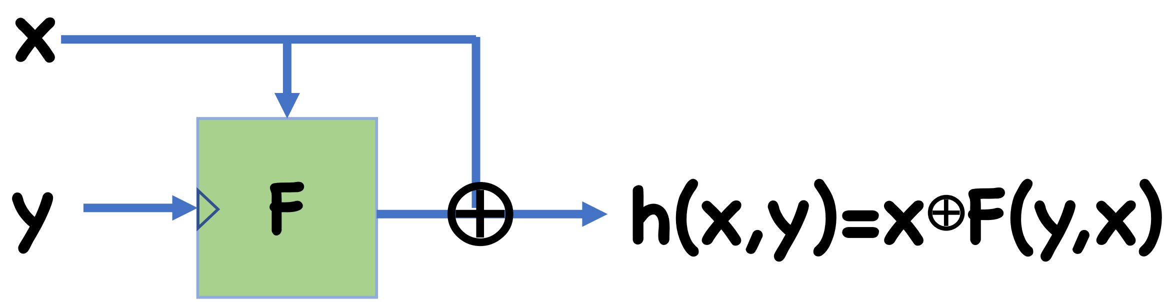
As described, cannot prove Merkle-Damgard is secure if inputs are allowed to have different length
- Recursion ends at different points

Need proper padding to enable security proof
- Ex: append message length to end of message

Constructing **h**

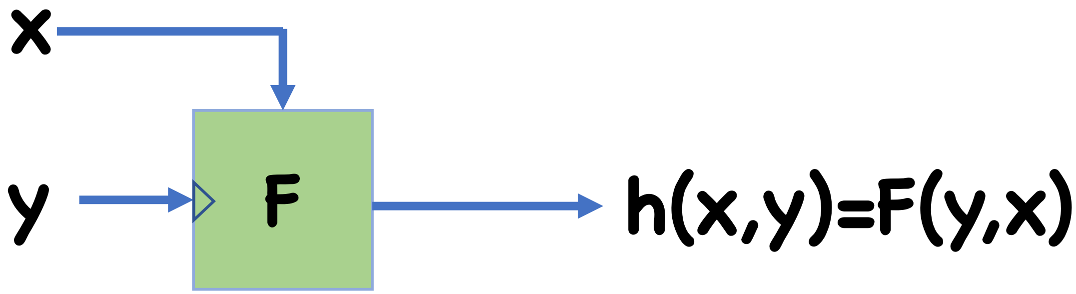
Common approach: use block cipher

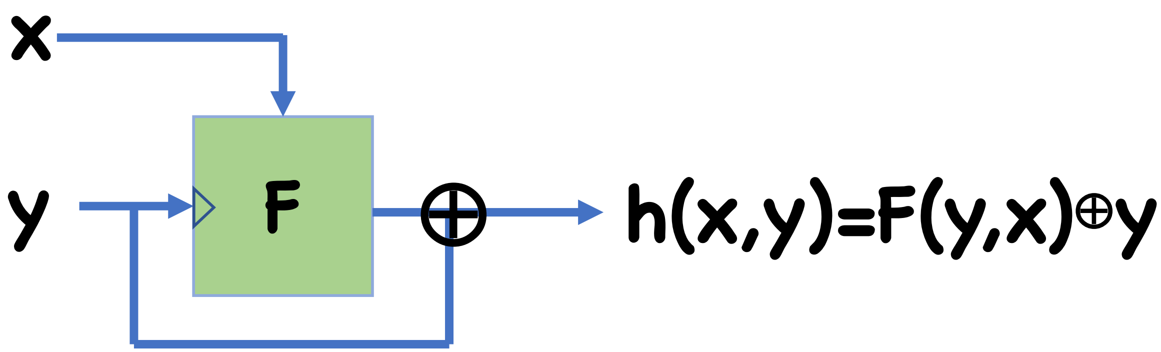
Davies-Meyer



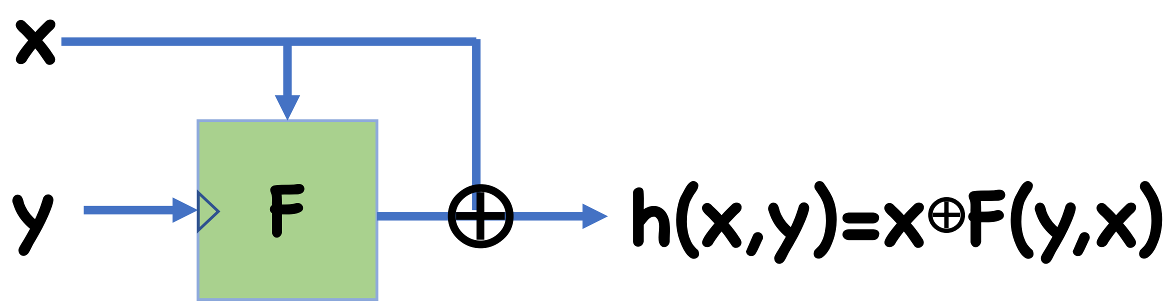
Constructing **h**

Some other possibilities are insecure





Constructing **h**



Why do we think Davies-Meyer is reasonable?

- Cannot prove collision resistance just based on **F** being a secure PRP

Instead, can argue security in "ideal cipher" model

- Pretend **F**, for each key **y**, is a uniform random permutation

Announcements/Reminders

Last day to submit PR1

- Submit archive file on Canvas

HW3 due on Oct 20