COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2018

Randomized Encryption

Syntax:

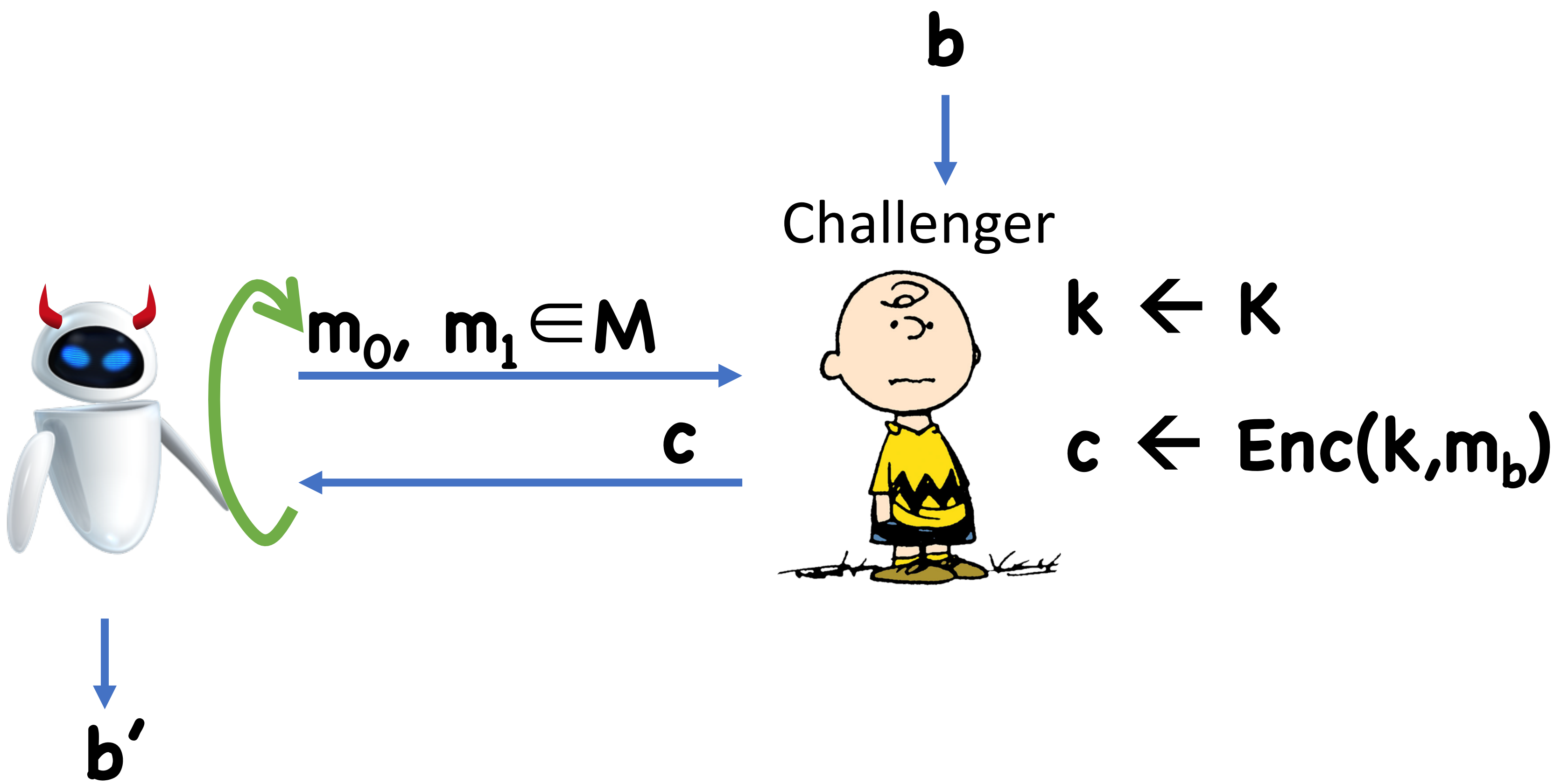
- Key space **K** (usually $\{0,1\}^{\lambda}$)
- Message space **M** (usually $\{0,1\}^{n}$)
- Ciphertext space **C** (usually $\{0,1\}^{m}$)
- **Enc: K×M → C** (potentially probabilistic)
- **Dec: K×C → M∪{⊥}** (usually deterministic)

Correctness:

- For all **k∈K, m∈M,**

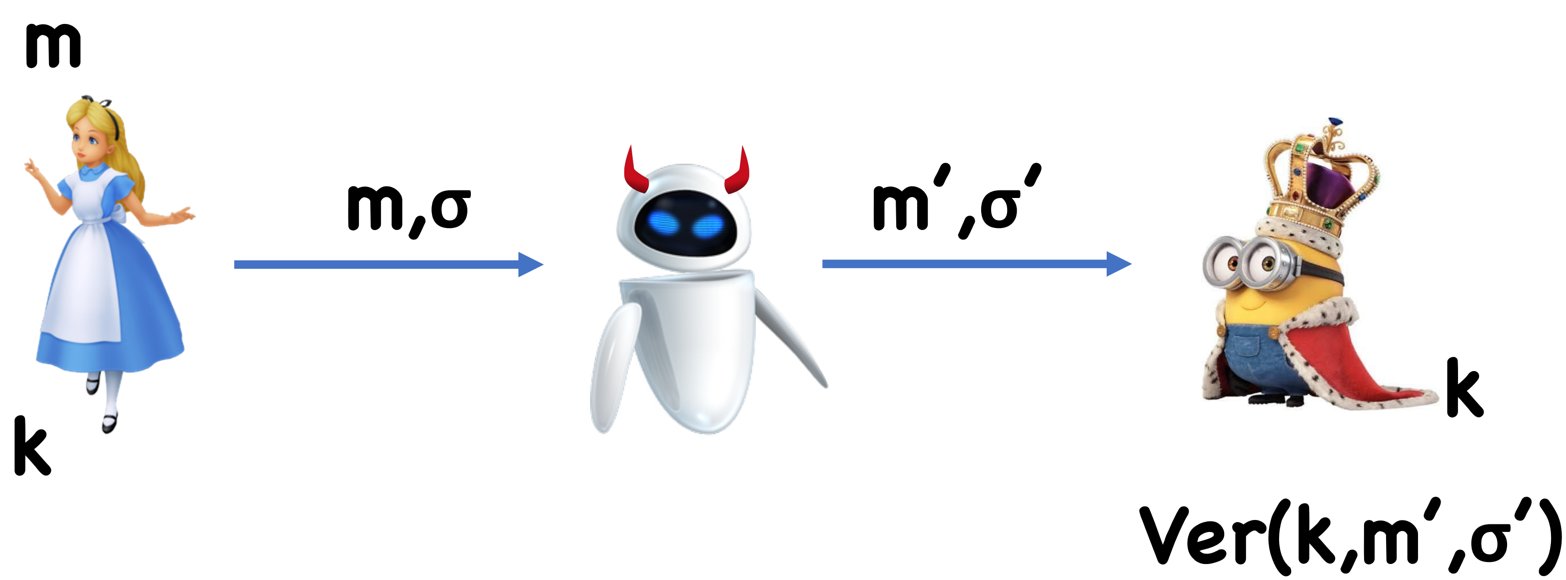
$$\Pr[\ \mathrm{Dec}(k, \mathrm{Enc}(k,m)\ ) = m\ ] \quad = \quad 1$$

Left-or-Right Experiment



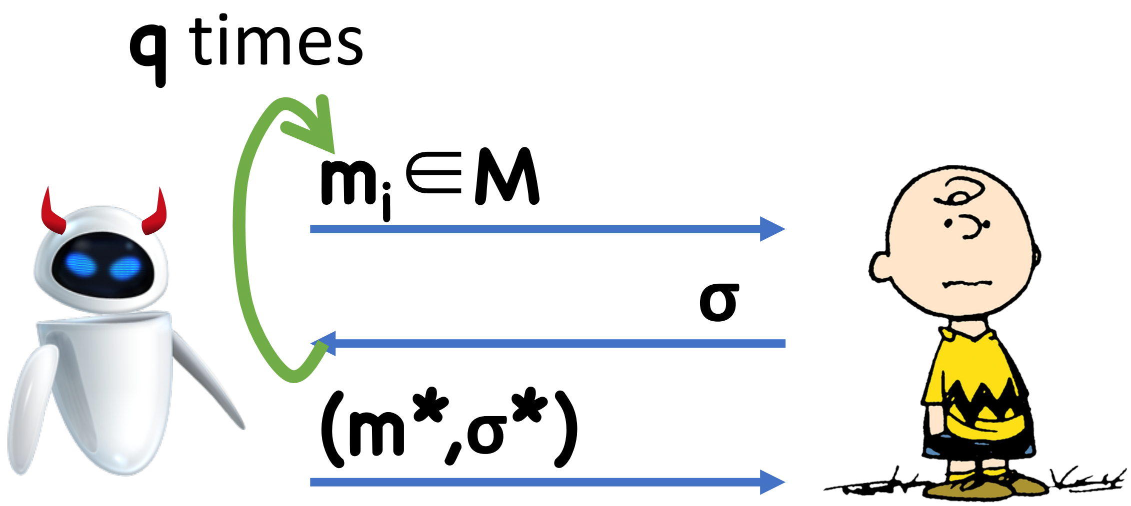
LoR-Exp$_b$( **)**

Message Authentication



Goal: If Eve changed **m**, Bob should reject

q-Time MACs



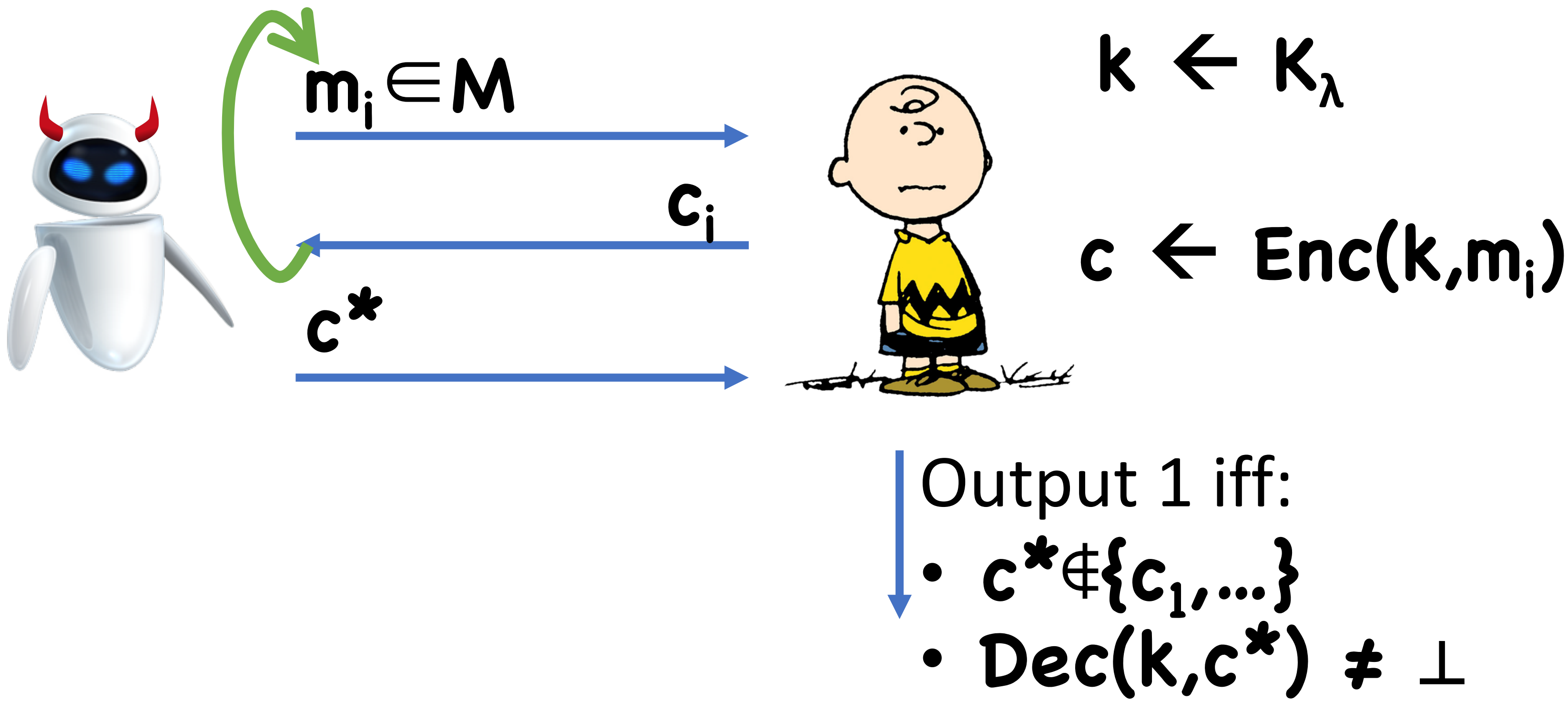
$k \leftarrow K$

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

Output 1 iff:
- $m^* \notin \{m_1,\ldots,m_q\}$
- $Ver(k,m^*,\sigma^*) = 1$

qCMA-Adv() = Pr[ outputs 1]

Unforgeability



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

Pseudorandom Permutations

(also known as block ciphers)

Functions that "look like" random **permutations**

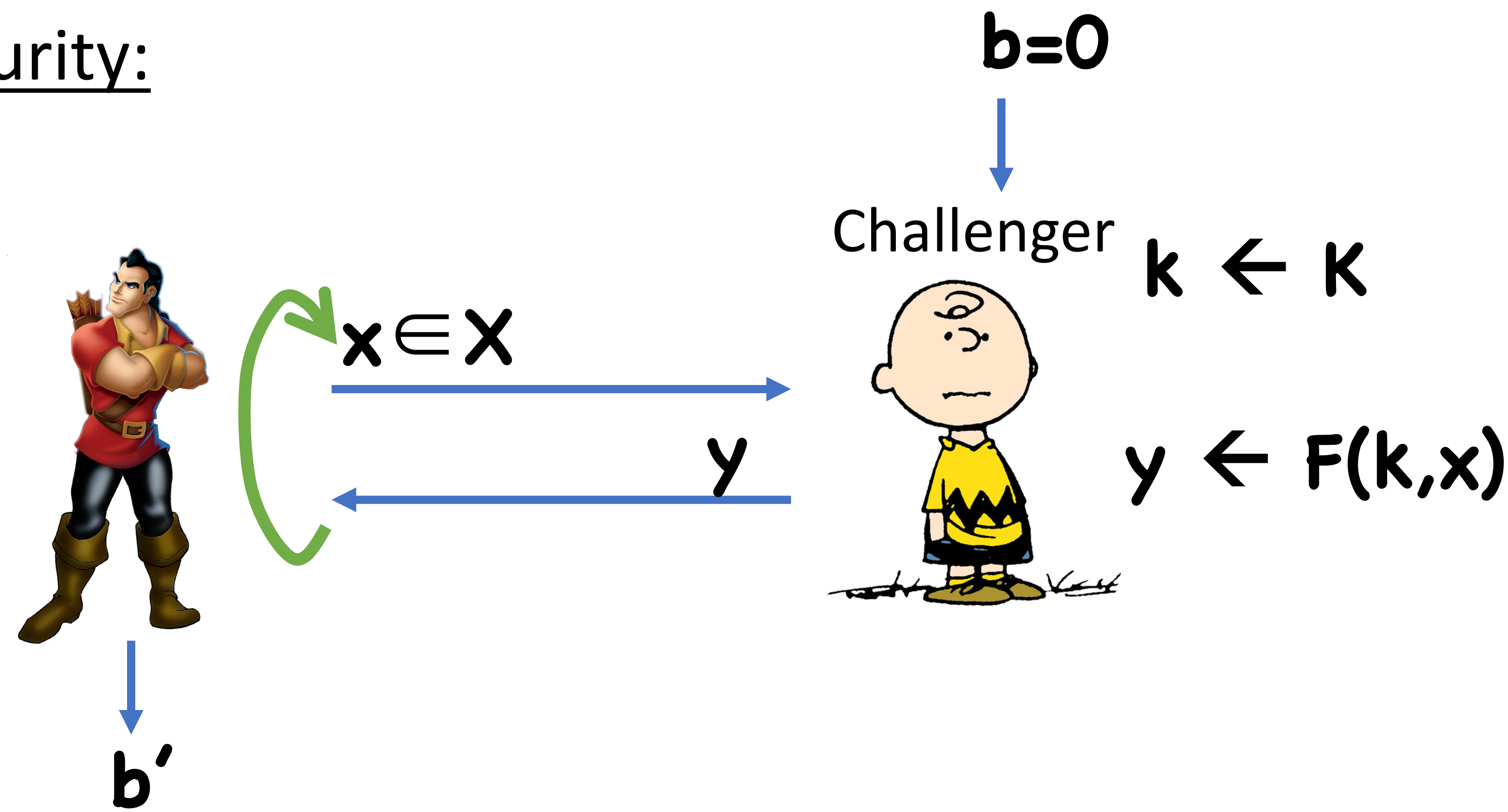
Syntax:

- Key space **K** (usually $\{0,1\}^\lambda$)
- Domain=Range= **X** (usually $\{0,1\}^n$)
- Function $F: K \times X \rightarrow X$
- Function $F^{-1}: K \times X \rightarrow X$

Correctness: $\forall k,x,\ F^{-1}(k, F(k, x)) = x$

Pseudorandom Permutations

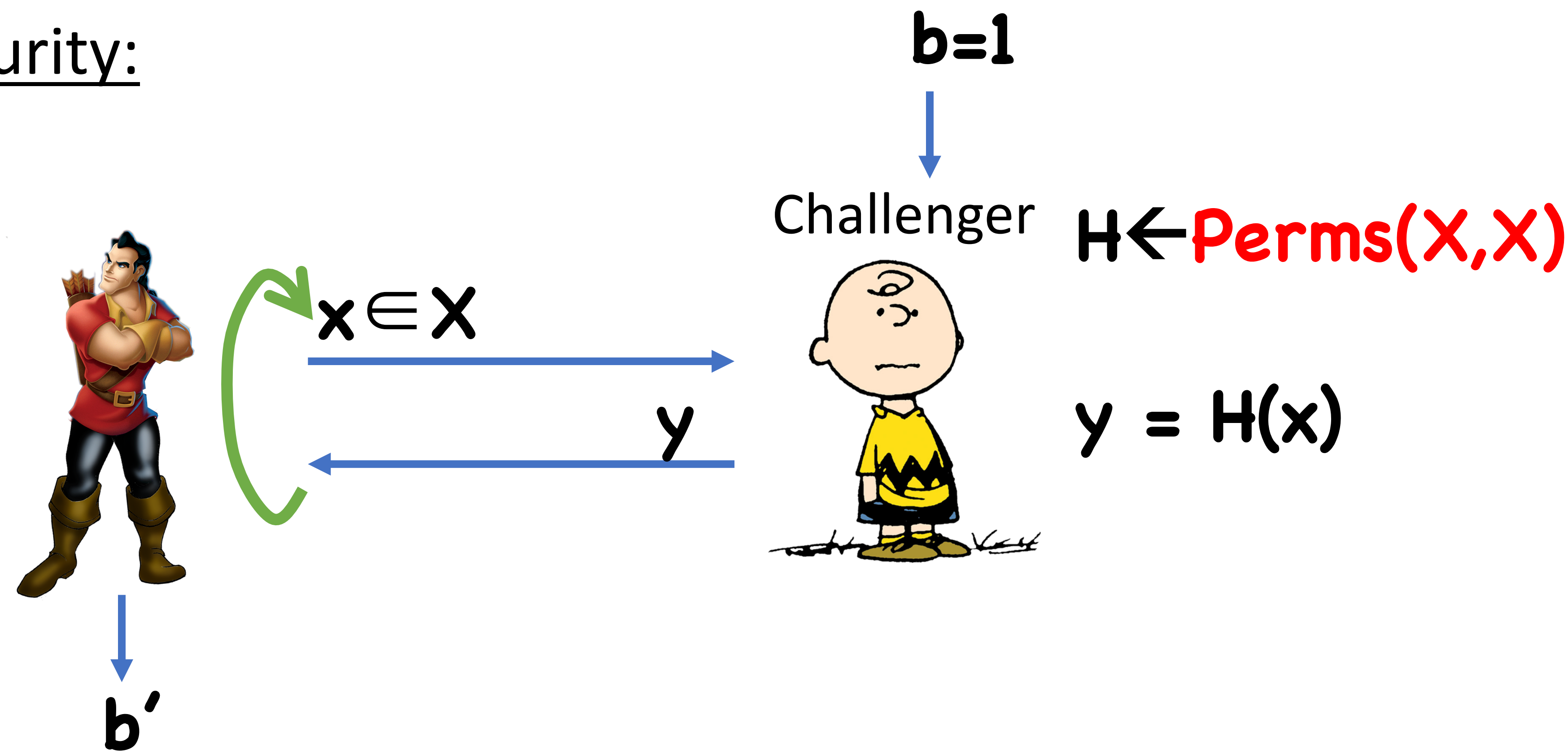
Security:



$\text{PRF-Exp}_0($$)$

Pseudorandom Permutations

Security:



PRF-Exp$_1$()

PRF Security Definition

Definition: F is a **(t,q,ε)–**secure PRP if, for all  running in time at most **t** and making at most **q** queries,

$\big|\ \Pr[1 \leftarrow \text{PRF-Exp}_0($  $)\ ]$

$\quad - \Pr[1 \leftarrow \text{PRF-Exp}_1($  $)\ ]\ \big| \ \leq\ \varepsilon$

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

Correctness: $n \ll l$

Security?

Definition: H is **(t,ε)–**collision resistant if, for all running  in time at most **t**,

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

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

Correctness: $n \ll l$

Security

Definition: H is **(t,ε)-**collision resistant if, for all running  in time at most **t**,

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

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 **(t,q,ε_0)**-CMA-secure and **H** is **(t,ε_1)**-collision resistant, then **(MAC',Ver')** is

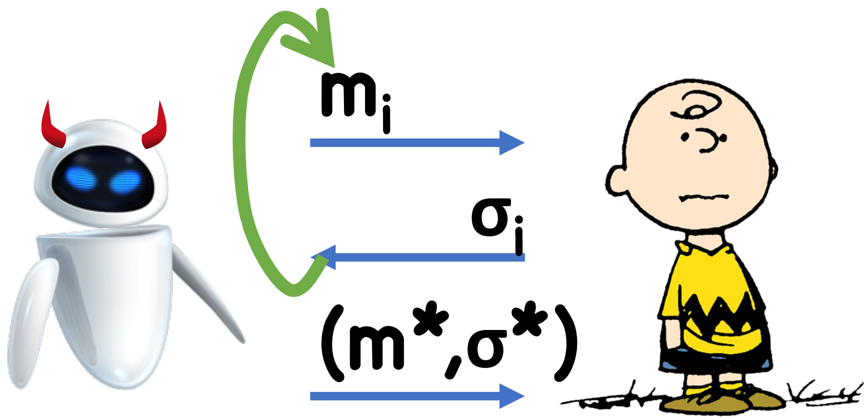
$(t-t', q, \varepsilon_0+\varepsilon_1)$-CMA secure

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:

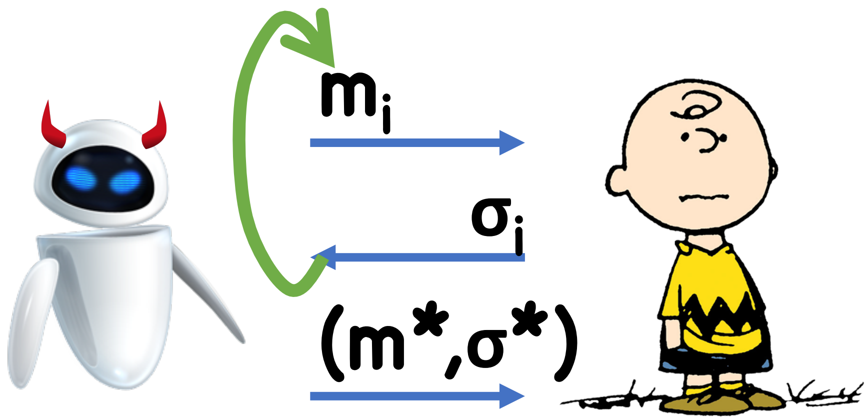
- $m^* \notin \{m_1,\ldots\}$
- $Ver(k,t^*,\sigma^*)$ where $t^* \leftarrow H(k_H,m^*)$

Proof

<u>Hybrid 1</u>

$k_H \leftarrow K_H$

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



$t_i \leftarrow H(k_H,m_i)$

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

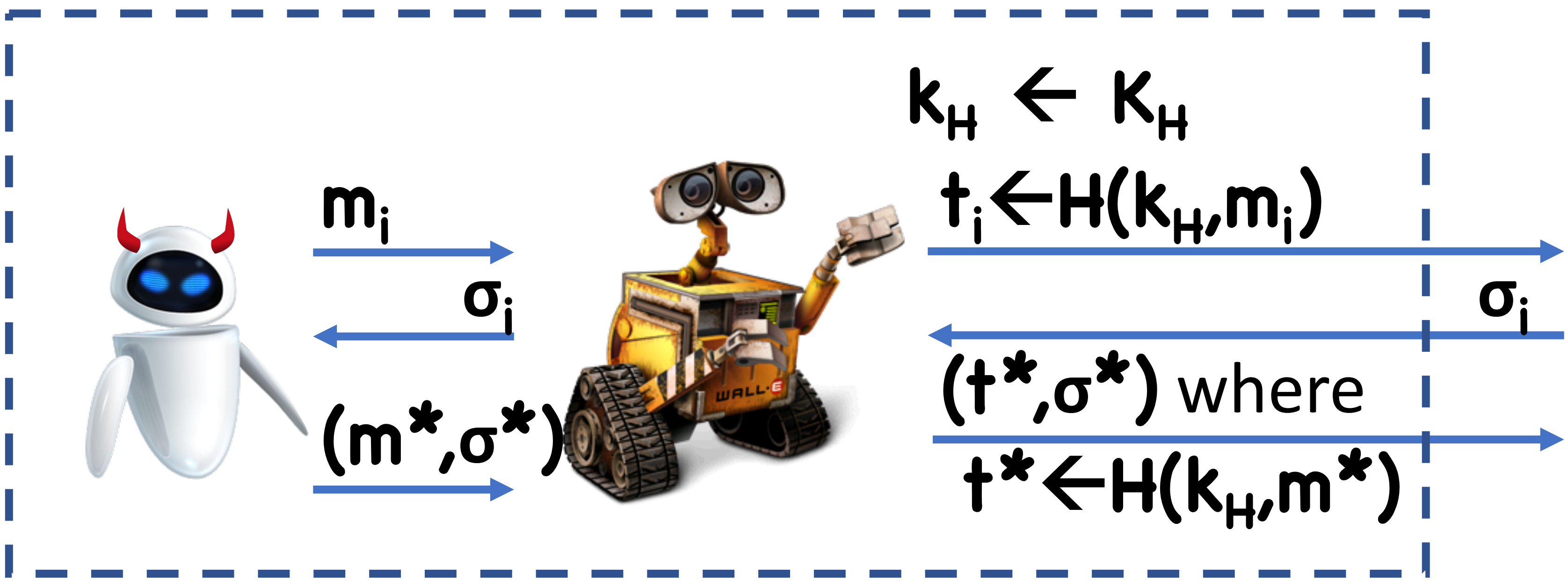
Output 1 iff:

- $t^* \notin \{t_1,\ldots\}$
- $Ver(k,t^*,\sigma^*)$ 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

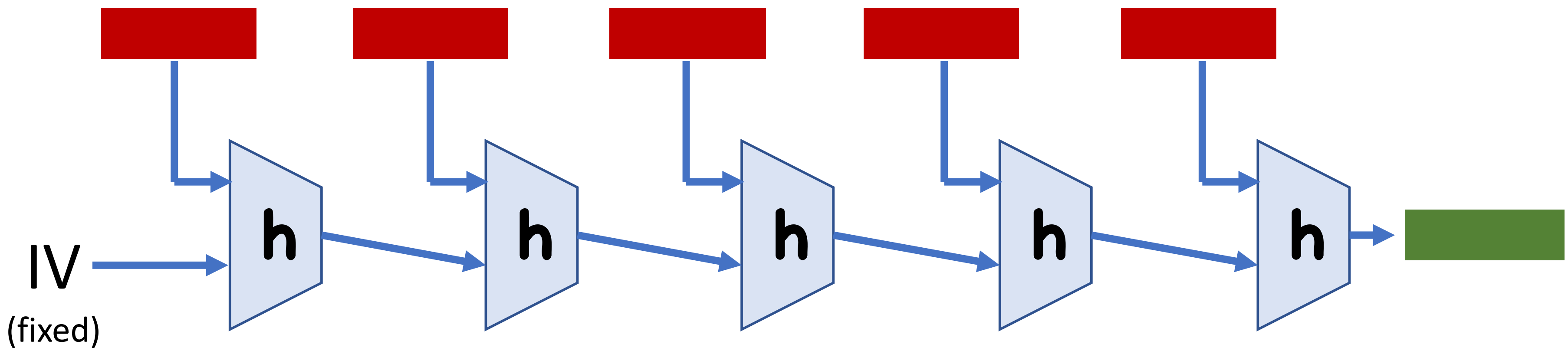
Goal: given **h** that compresses small inputs, construct **H** that compresses large inputs

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

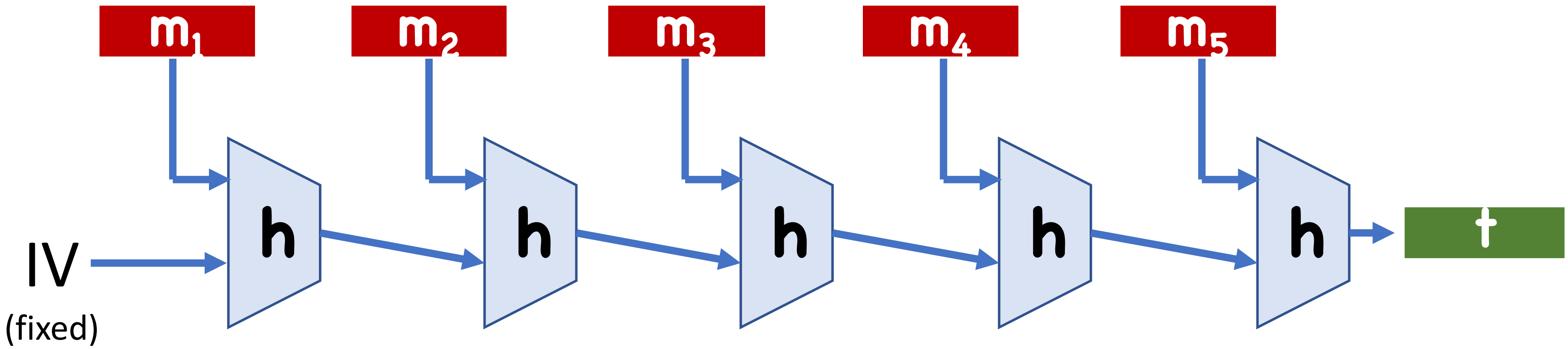
- Called compression functions
- Easier to design

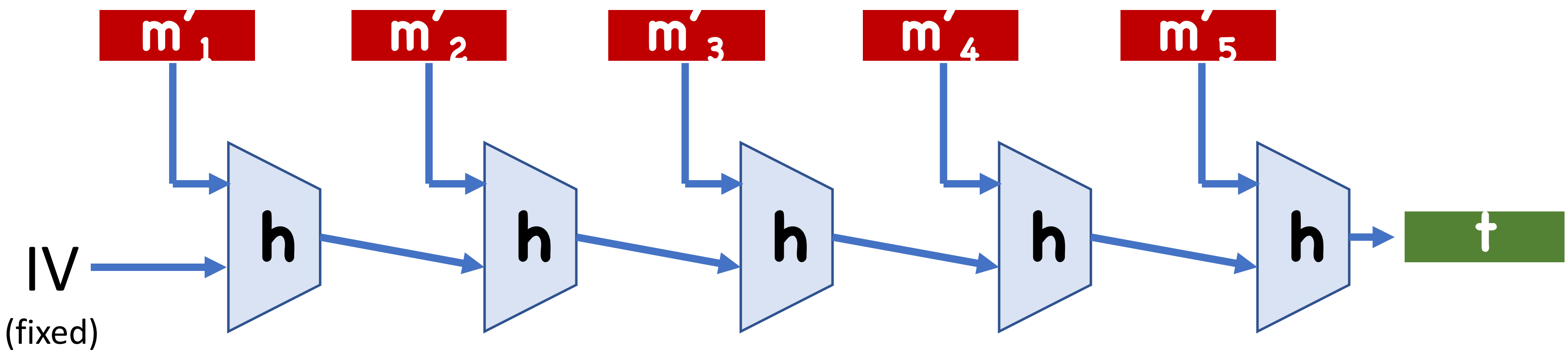
Merkle-Damgard



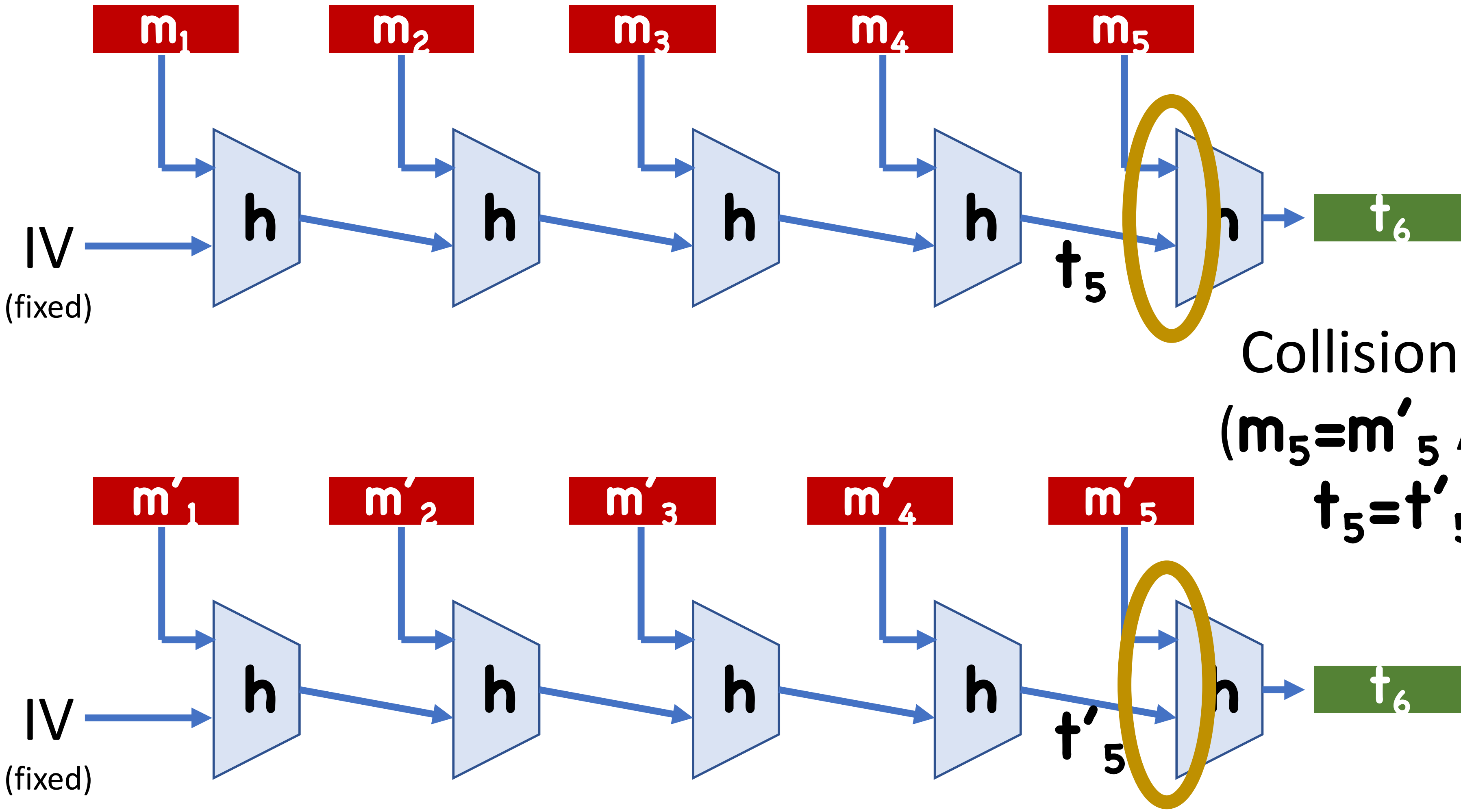
Theorem: If an adversary knows a collision for fixed-length Merkle-Damgard, it can also compute a collision for **h**

Proof





Proof

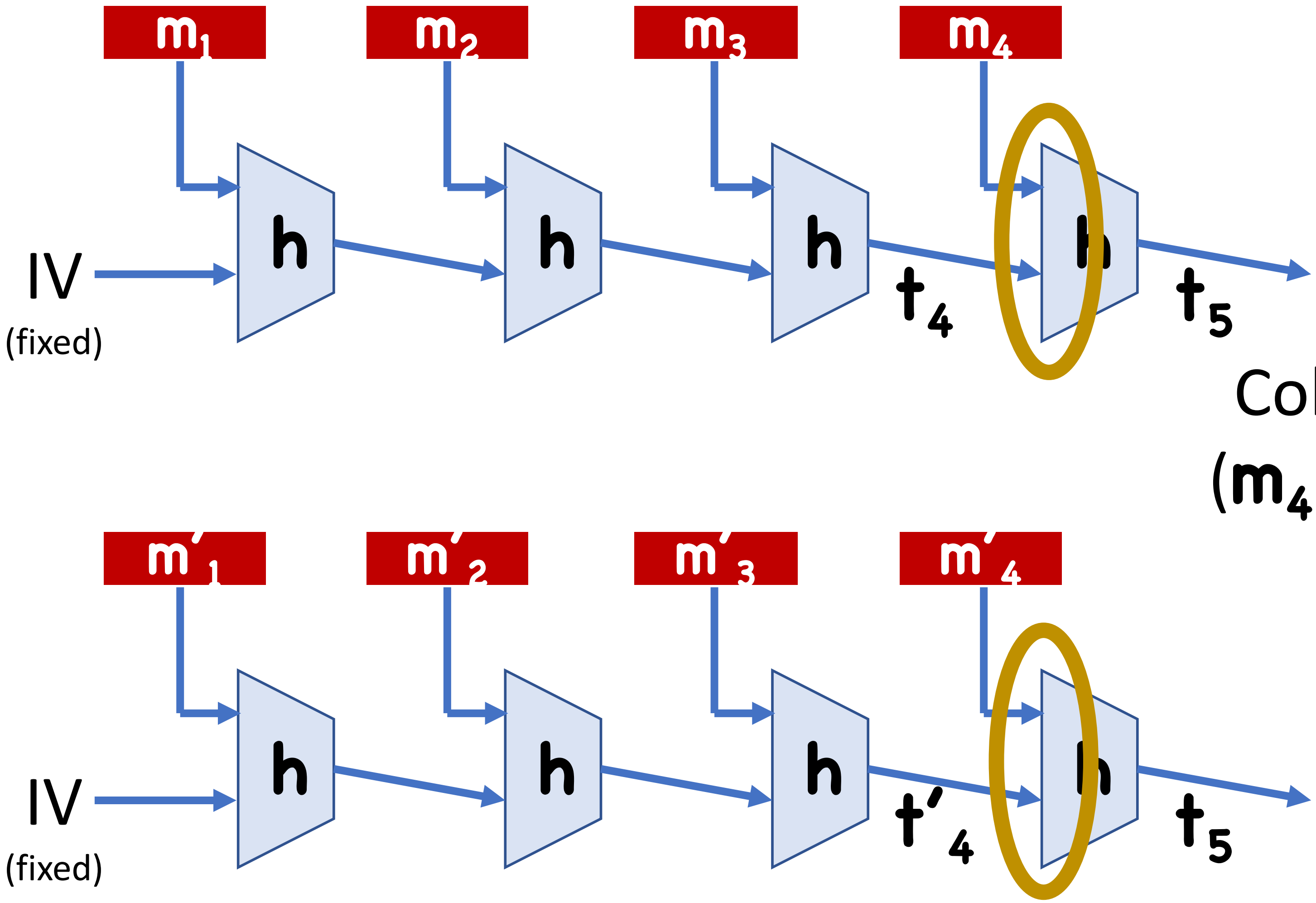


Collision OR

(m_5=m'_5 AND

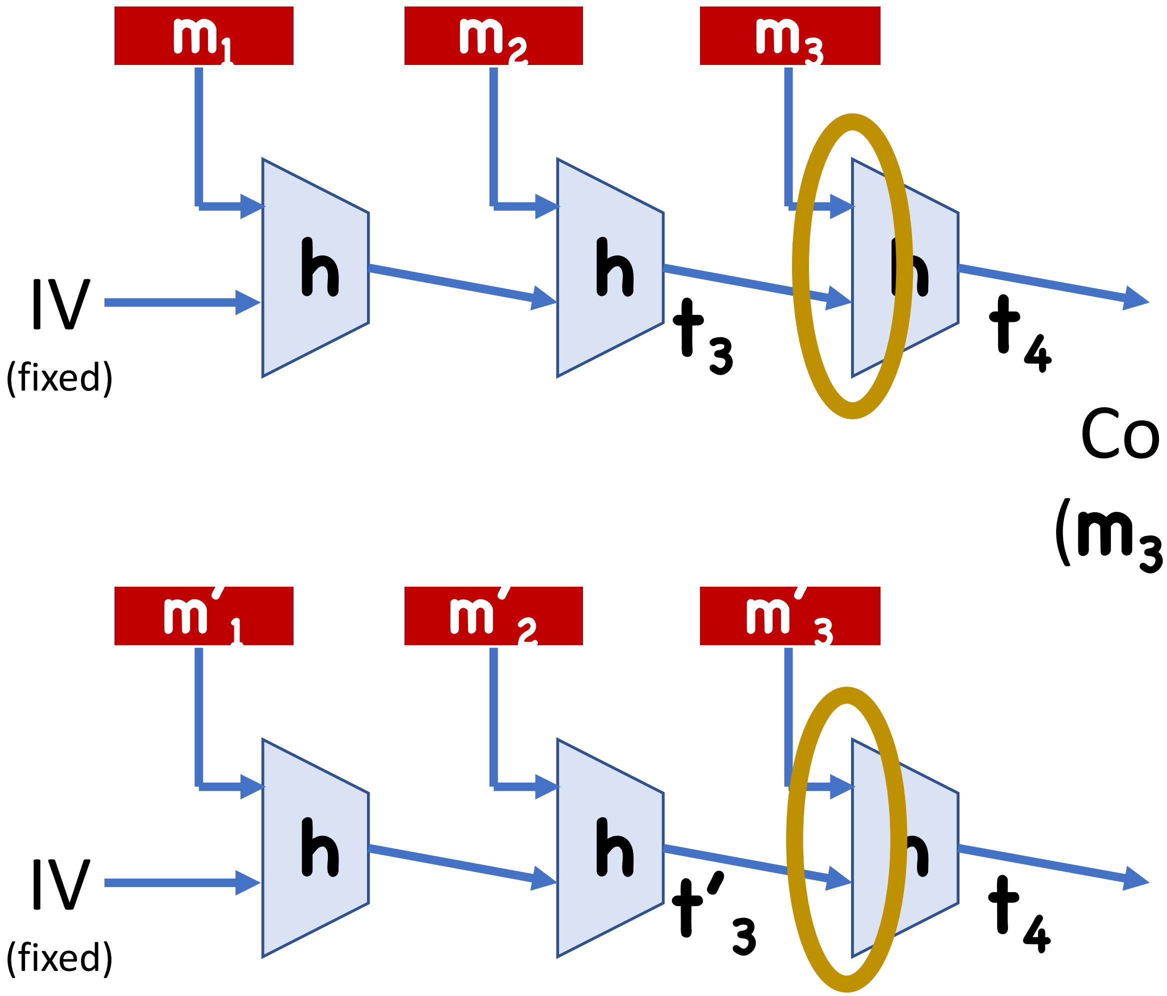
t_5=t'_5)

Proof



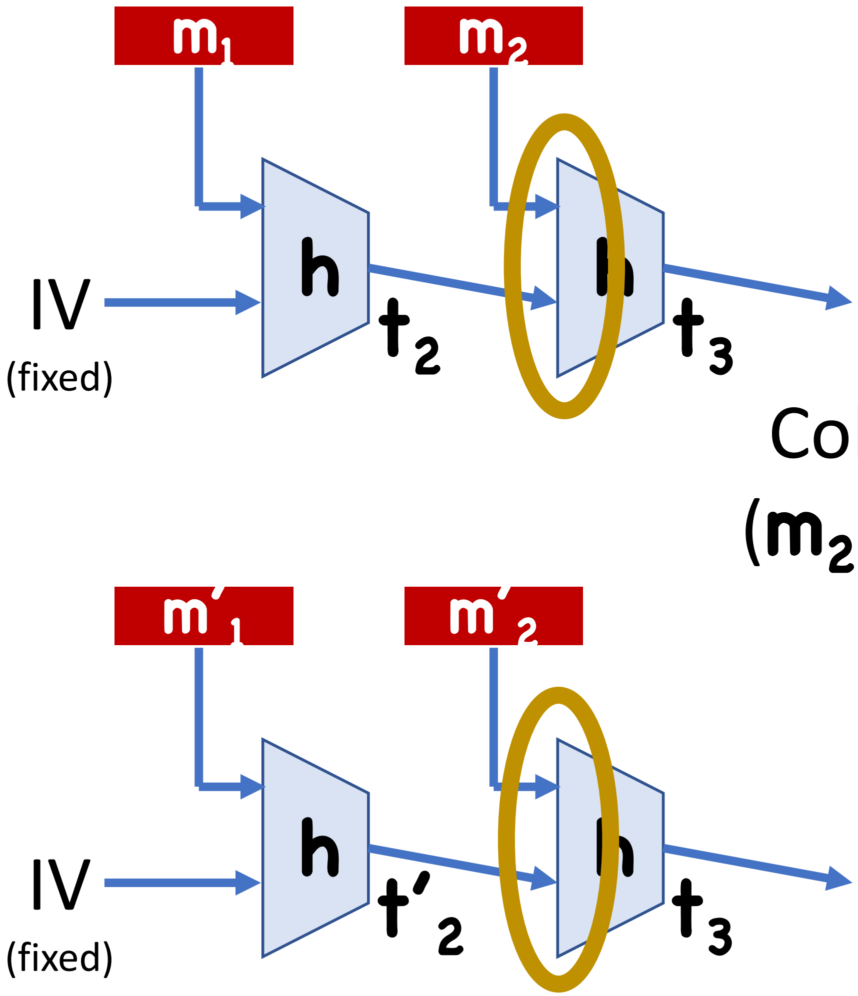
Collision OR
(m_4=m'_4 AND
t_4=t'_4)

Proof



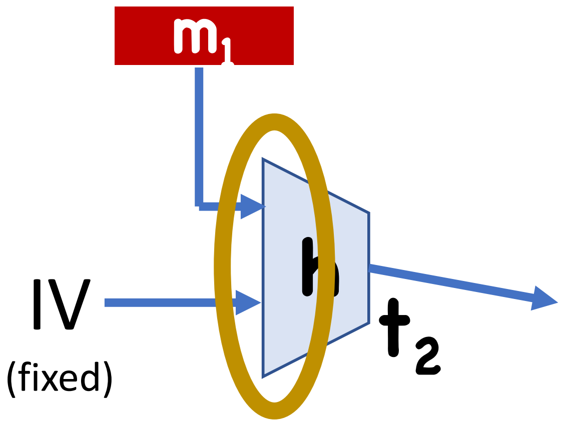
Collision OR ($m_3 = m'_3$ AND $t_3 = t'_3$)

Proof



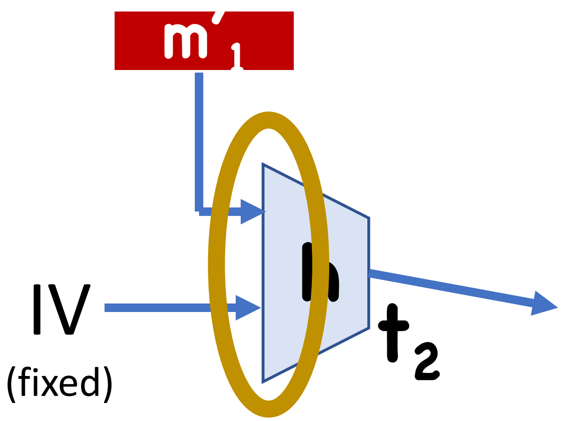
Collision OR (m_2=m'_2 AND t_2=t'_2)

Proof



Collision OR

$m_1 = m'_1$



But, if **$m_1 = m'_1$**, then **$m = m'$**

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

- What if adversary knows an input **x** such that **h(x||IV) = IV**?

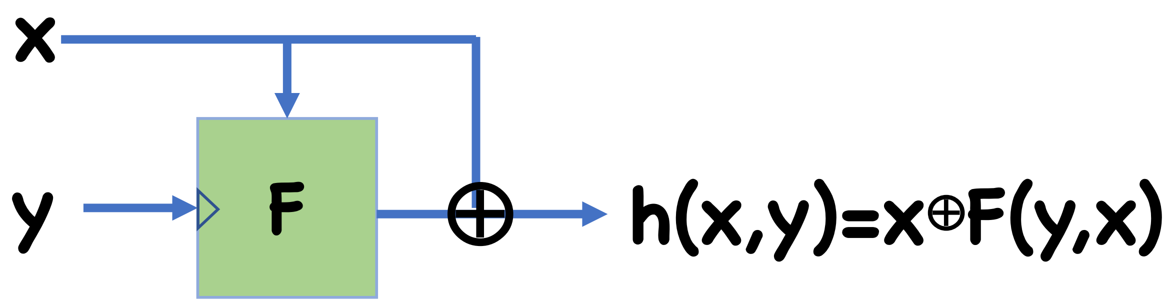
Need proper padding to enable security proof

- Ex: append message length to end of message

Constructing **h**

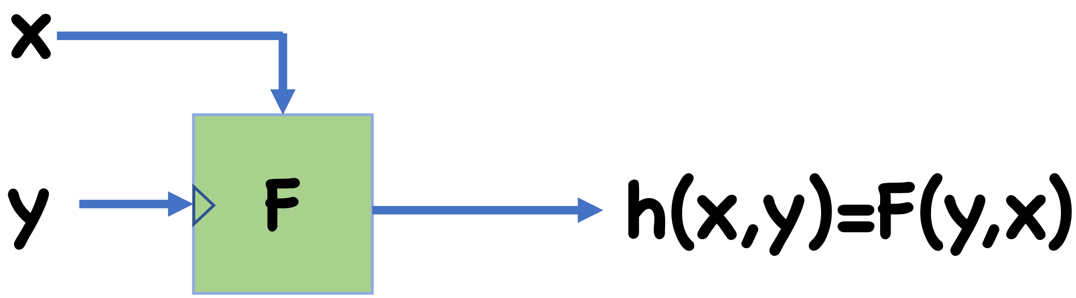
Common approach: use block cipher

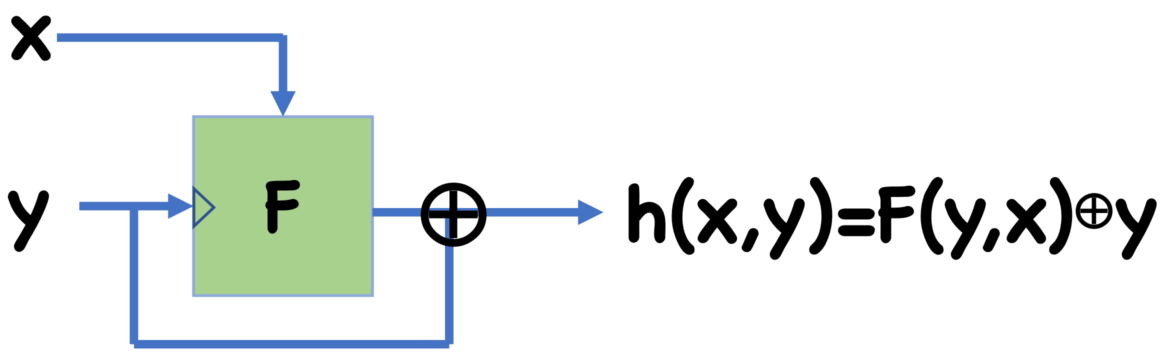
Davies-Meyer



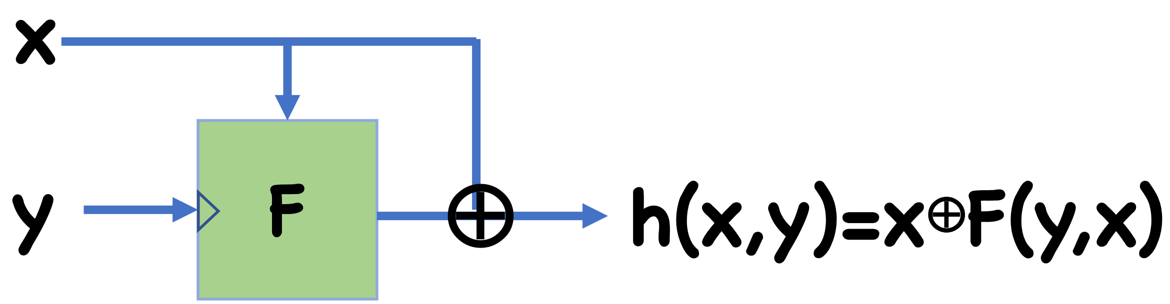
Constructing **h**

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

We said 128 bit security is usually enough

Why is a block cipher with 128-bit blocks insufficient?

Birthday Attack

If the range of a hash function is $\mathbf{R}$, a collision can be found in time $\mathbf{T=O(|R|^{1/2})}$

Attack:

- Given key $\mathbf{k}$ for $\mathbf{H}$
- For $\mathbf{i=1,\ldots,T}$,
  - Choose random $\mathbf{x_i}$ in $\mathbf{D}$
  - Let $\mathbf{t_i \leftarrow H(k,x_i)}$
  - Store pair $\mathbf{(x_i,\ t_i)}$
- Look for collision amongst stored pairs

Birthday Attack

Analysis:

Expected number of collisions

= Number of pairs × Prob each pair is collision

$\approx$ **(T choose 2) × 1/|R|**

By setting **$T=O(|R|^{1/2})$,** expectend number of collisions found is at least **1**

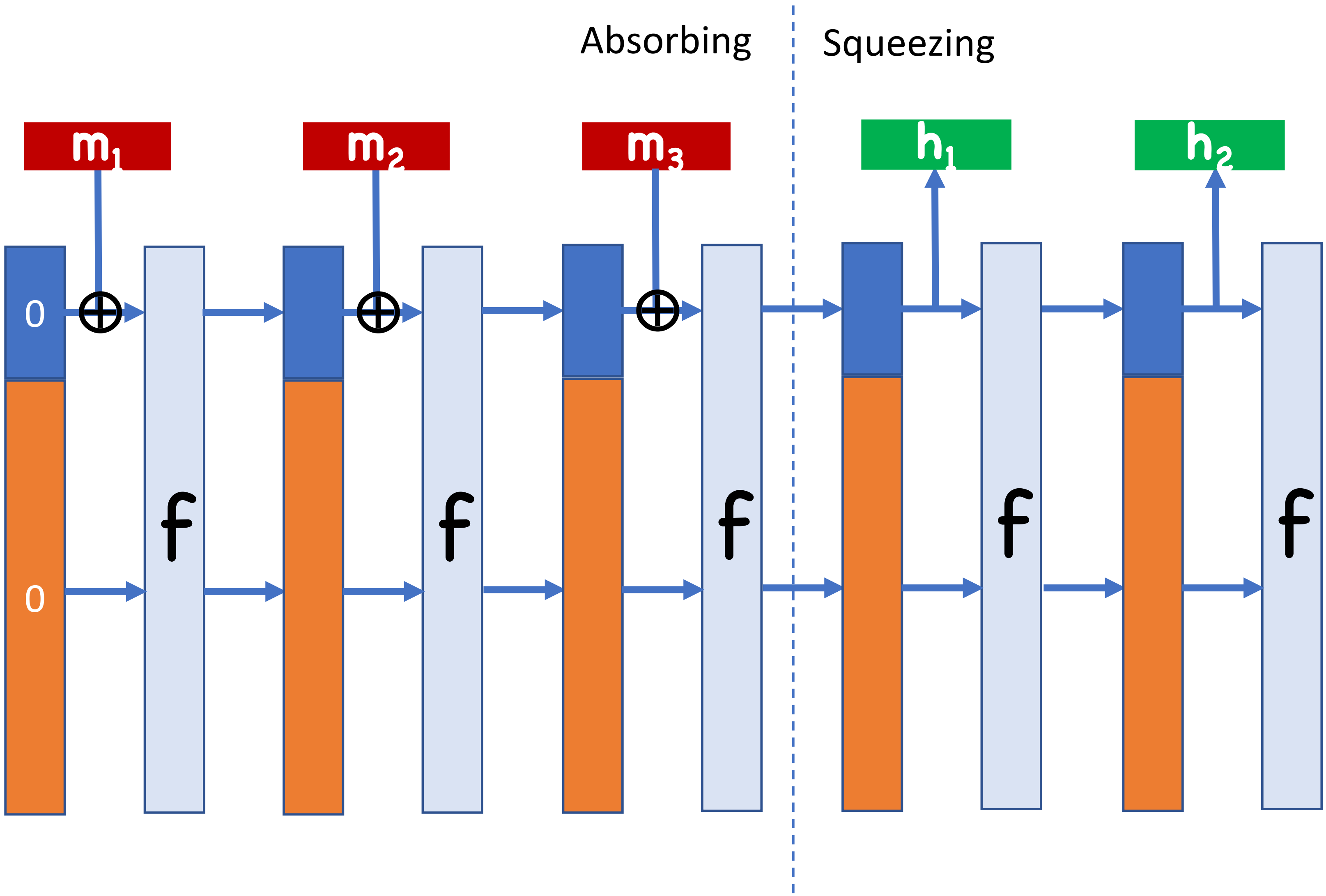
$\Rightarrow$ likely to find a collision

Birthday Attack

Space?

Possible to reduce memory requirements to **$O(1)$**

Sponge Construction



Sponge Construction

Advantages:

- Round function **f** can be public invertible function
  (i.e. unkeyed SPN network)

- Easily get different input/output lengths

SHA-1,2,3

SHA-1,2 are hash functions built as follows:

- Build block cipher (SHACAL-1, SHACAL-2)
- Convert into compression function using Davies-Meyer
- Extend to arbitrary lengths using Merkle-Damgard

SHA-3 is based on sponge construction

SHA-1,2,3

SHA-1 (1995) is no longer considered secure

- 160-bit outputs, so collisions in time 2^{80}
- 2017: using some improvements over birthday attack, able to find a collision

SHA-2 (2001)

- Longer output lengths (256-bit, 512-bit)
- Few theoretical weaknesses known

SHA-3 (2015)

- NIST wanted hash function built on different principles

Basing MACs on Hash Functions

Idea: **MAC(k,m) = H(k || m)**

Thought: if **H** is a "good" hash function and **k** is random, should be hard to predict **H(k || m)** without knowing **k**

Unfortunately, cannot prove secure based on just collision resistance of **H**

Random Oracle Model

Pretend **H** is a truly random function

Everyone can query **H** on inputs of their choice

- Any protocol using **H**
- The adversary (since he knows the key)

A query to **H** has a time cost of 1

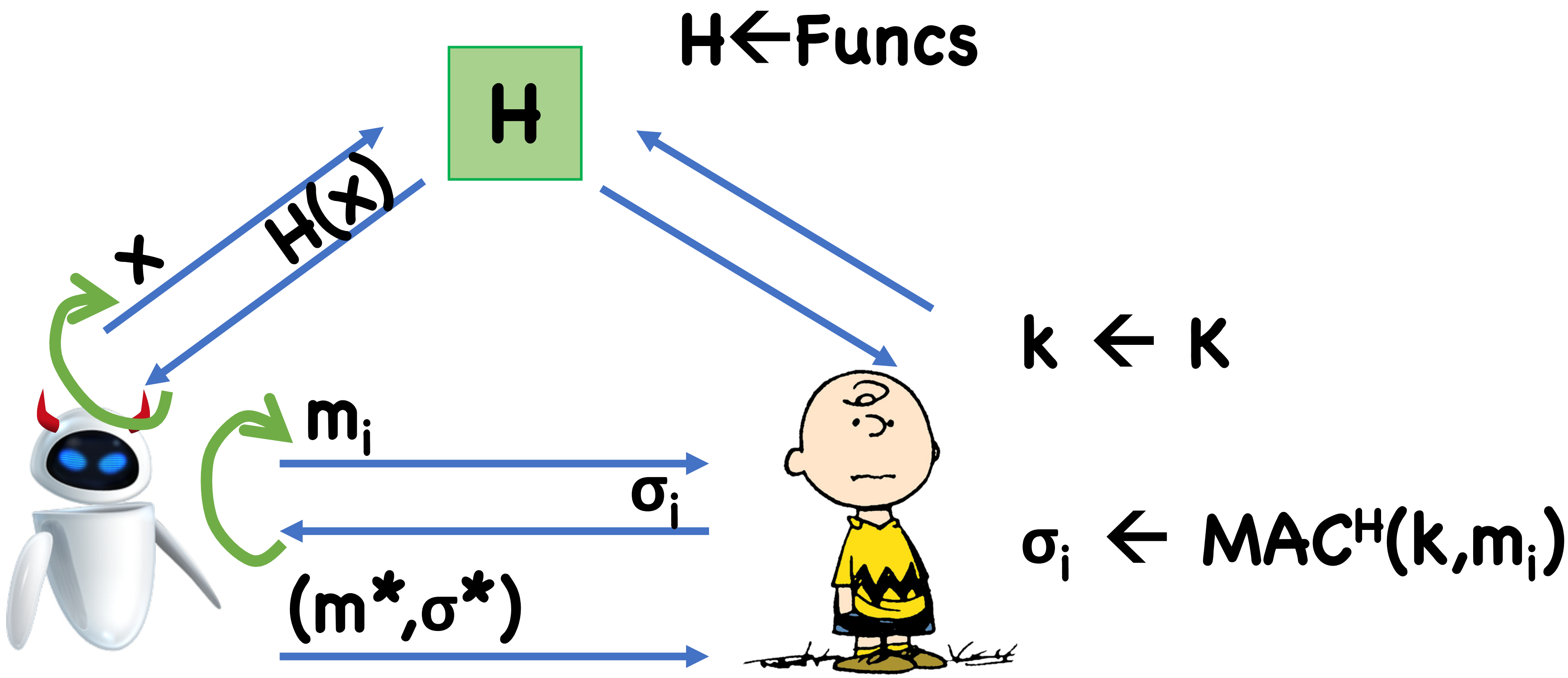
Intuitively captures adversaries that simple query **H**, but don’t take advantage of any structure

MAC in ROM

$MAC^H(k,m) = H(k||m)$
$Ver^H(k,m,\sigma) = (H(k||m) == \sigma)$

Theorem: $H(k \; || \; m)$ is a **$(t,\; q,\; qt/2^n)$–**CMA-secure MAC in the random oracle model

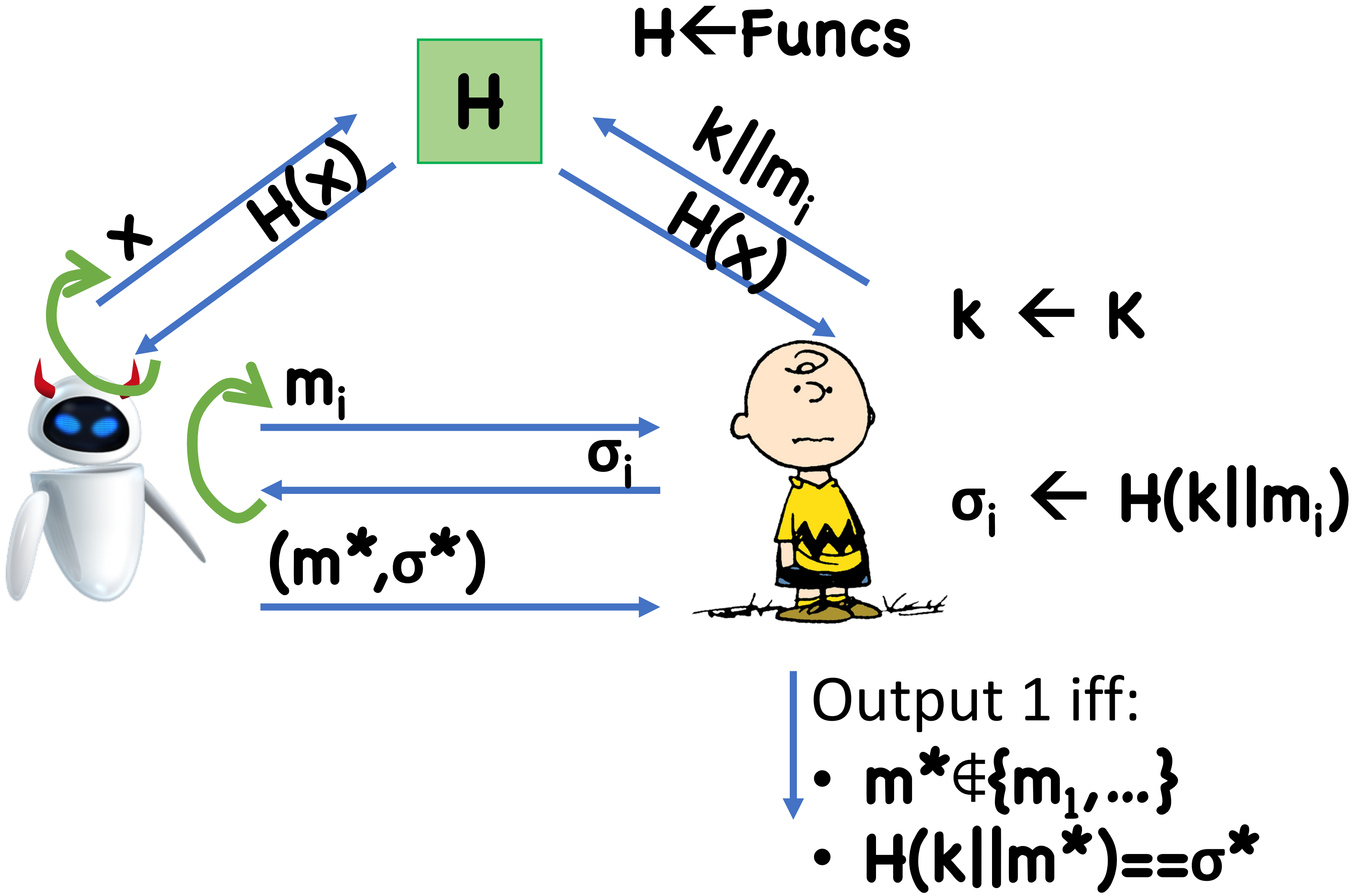
Meaning



Output 1 iff:

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

Meaning



Proof Idea

Value of **H(k||m*)** independent of adversary's view unless she queries **H** on **k||m***

- Only way to forge better than random guessing is to learn **k**

Adversary only sees truly rand and indep **H** values and MACs, unless she queries **H** on $\mathbf{k||m_i}$ for some **i**

- Only way to learn **k** is to query **H** on $\mathbf{k||m_i}$

However, this is very unlikely without knowing **k** in the first place

The ROM

A random oracle is a good

- PRF: $F(k,x) = H(k||x)$

- PRG (assuming H is expanding):
  - Given a random x, $H(x)$ is pseudorandom since adv is unlikely to query H on x

- CRHF:
  - Given poly-many queries, unlikely for find two that map to same output

The ROM

The ROM is very different from security properties like collision resistant

What does it mean that "Sha-1 behaves like a random oracle"?

- No satisfactory definition

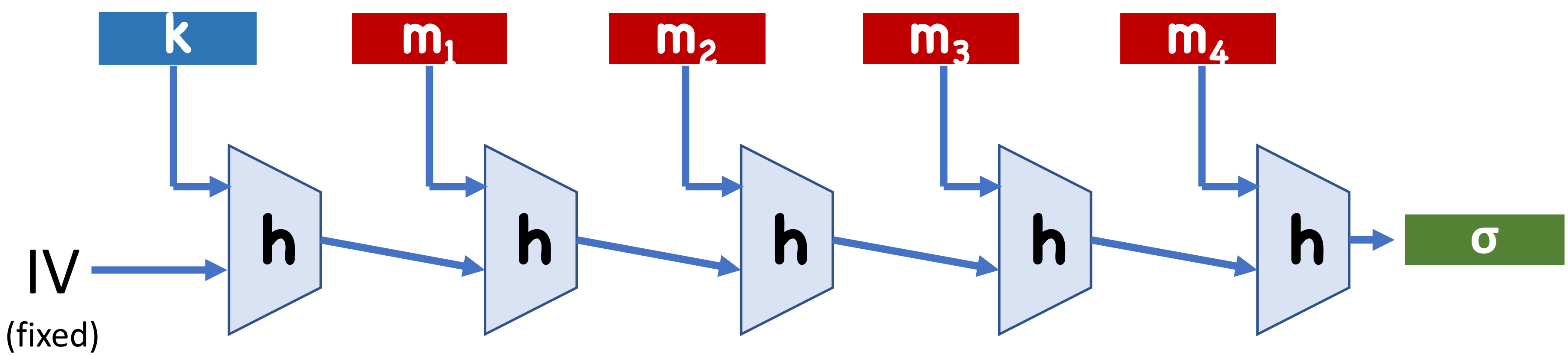
Therefore, a ROM proof is a heuristic argument for security

- If insecure, adversary must be taking advantage of structural weaknesses in **H**

When the ROM Fails

$MAC^H(k,m) = H(k||m)$
$Ver^H(k,m,\sigma) = (H(k||m) == \sigma)$

Instantiate with Merkle-Damgard (variable length)?



When the ROM Fails

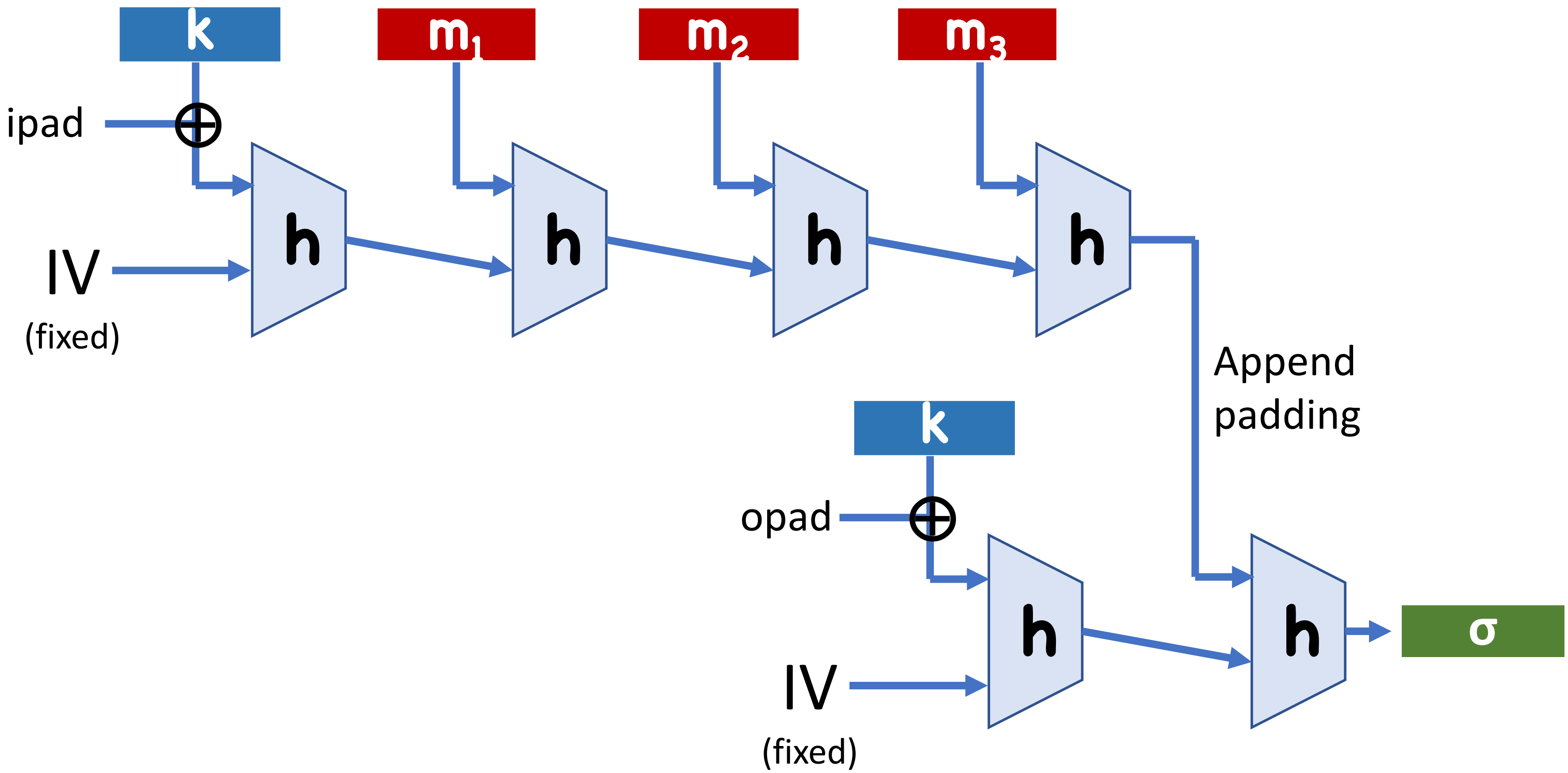
ROM does not apply to regular Merkle-Damgard

- Even if **h** is an ideal hash function

Takeaway: be careful about using ROM for non-"monolithic" hash functions

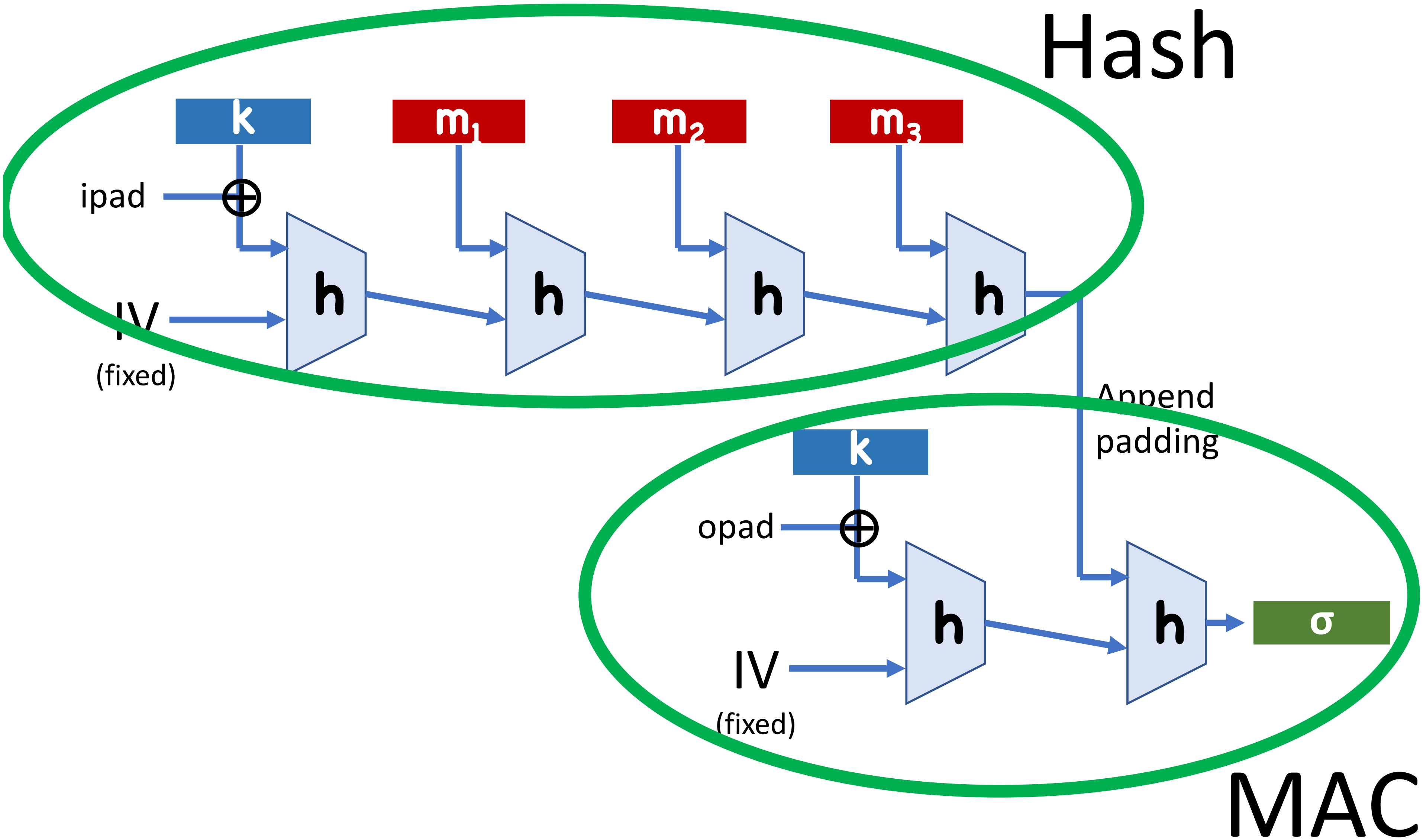
- Though still possible to pad MD in a way that makes it an ideal hash function if **h** is ideal

HMAC



k
m1
m2
m3
ipad
h
h
h
h
IV
(fixed)
Append
padding
k
opad
h
h
σ
IV
(fixed)

HMAC



Hash
k
m1
m2
m3
ipad
h
h
h
h
IV
(fixed)
Append
padding
k
opad
h
h
σ
IV
(fixed)
MAC

HMAC

ipad,opad?

- Two different (but related) keys for hash and MAC
- ipad makes hash a “secret key” hash function
- Even if not collision resistant, maybe still impossible to find collisions when hash key is secret
- Turned out to be useful after collisions found in MD5

Reminders

Homework 4 will be out later today – Due April 3

Project 2 will be out by next class – Due April 17
- Finding collisions in poorly designed hash functions