COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2017

Authenticated Encryption Syntax

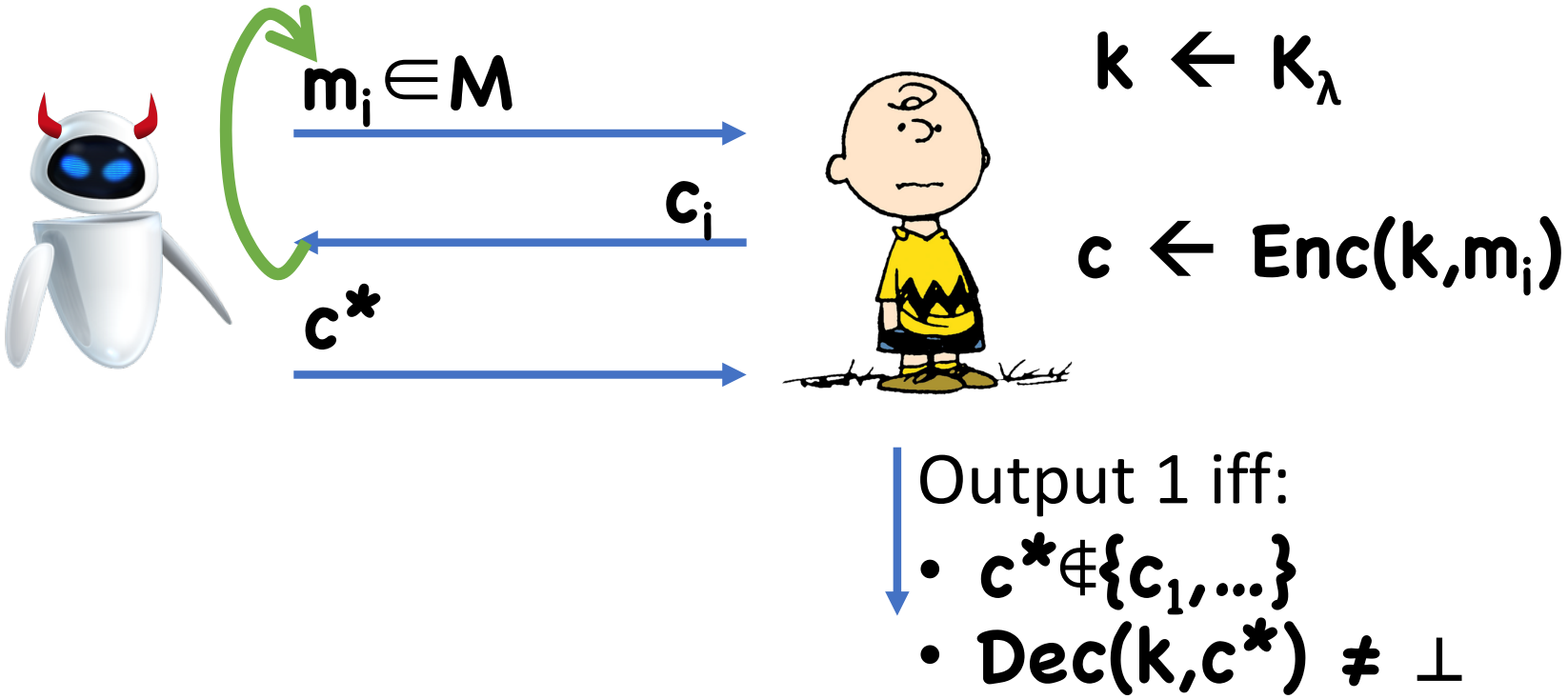
Syntax:

- **Enc: K×M → C**
- **Dec: K×C → M∪{⊥}**

Correctness:

- For all **k∈K, m∈M, Dec(k, Enc(k,m)) = m**

Unforgeability



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

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

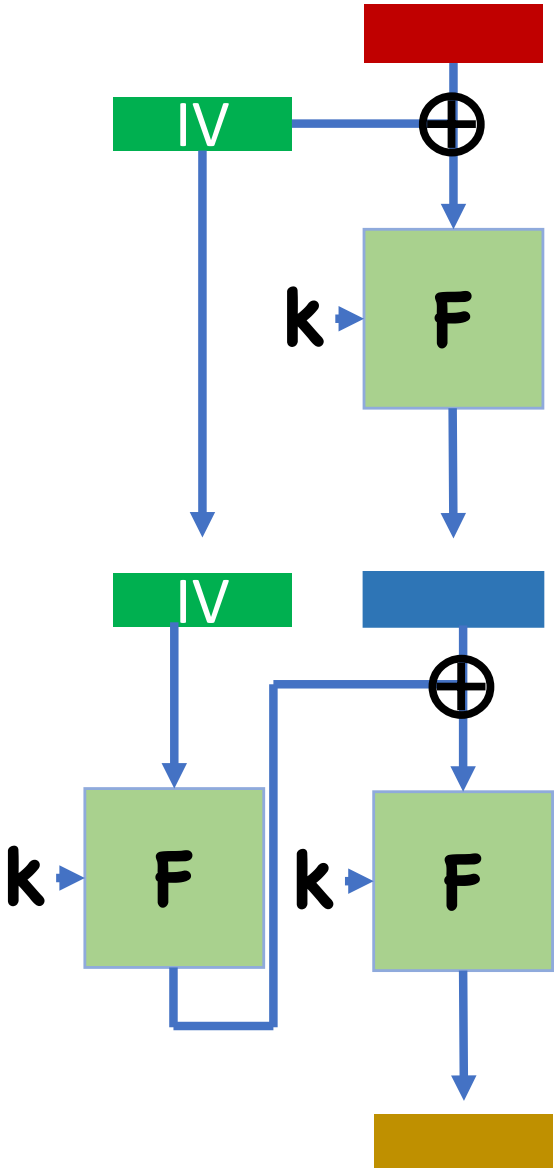
Constructing Authenticated Encryption

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



In general, don’t use the same key for encryption and authentication

Using Same Key for Encrypt and MAC

In general, do not use same key for multiple purposes

- Schemes may interact poorly when using the same key

However, some modes of operation do allow same key to be used for both authentication and encryption

CCM Mode

CCM = Counter Mode with CBC-MAC in Authenticate-then-Encrypt combination

Possible to show that using same key for authentication and encryption still provides security

Today

More Authenticated Encryption

Collision Resistance

Efficiency of Authenticated Encryption

So far, most modes can be implemented well in streaming applications

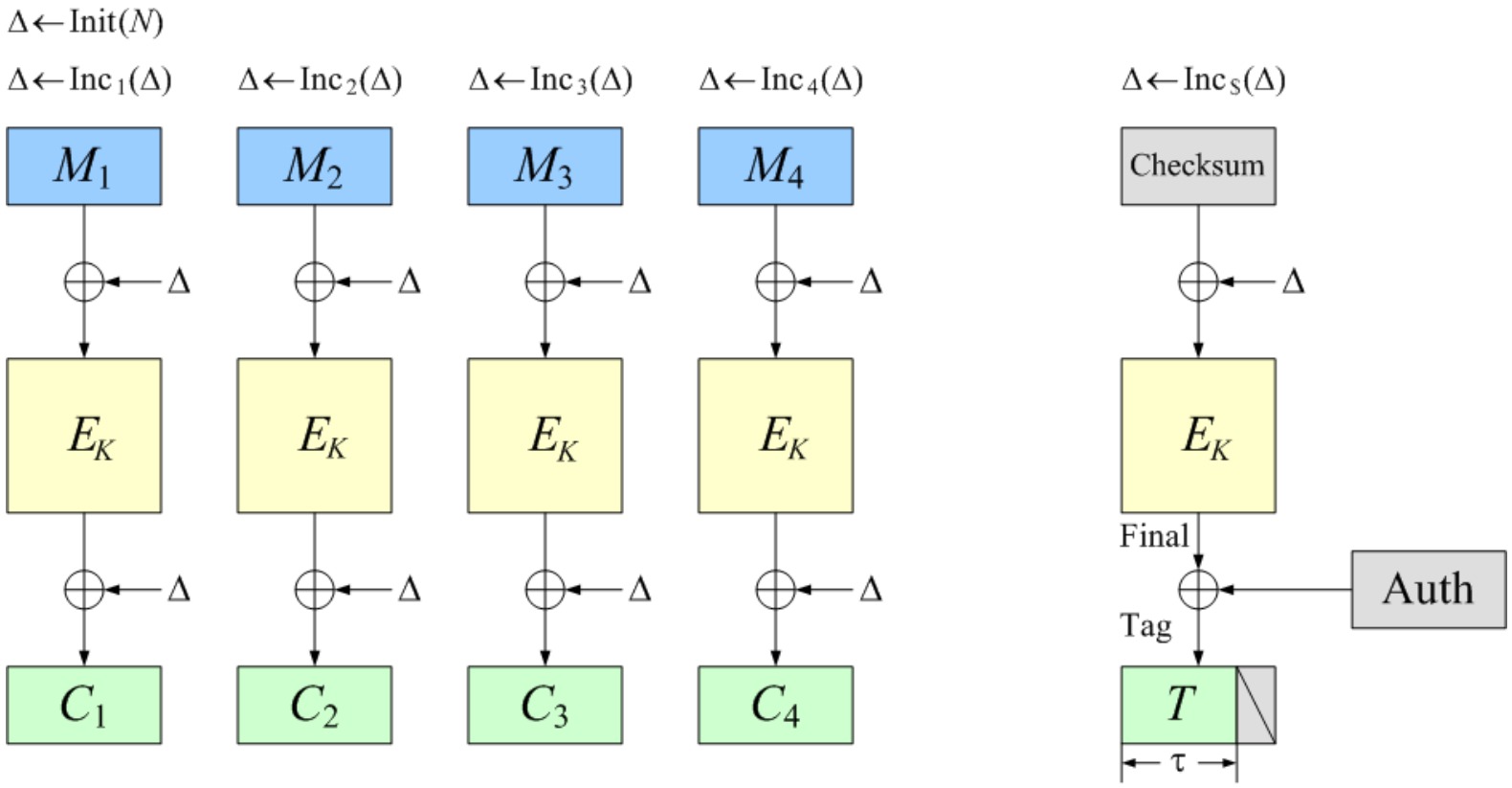
- Only need to read ciphertext once
- Can compute MAC and ciphertext at the same time

However, most modes seen require two block cipher operations per block

- 1 for encryption
- 1 for authentication

Ideally, would have only 1 block cipher op per block

OCB Mode


Δ ← Init(N)
Δ ← Inc1(Δ)
Δ ← Inc2(Δ)
Δ ← Inc3(Δ)
Δ ← Inc4(Δ)
Δ ← IncS(Δ)
M1
M2
M3
M4
Checksum
Δ
EK
Final
Auth
Tag
C1
C2
C3
C4
T
τ

OCB Mode

Twice as fast as other block cipher modes of operation

However, not used much in practice

- Patents!

Another mode: GCM:

- Roughly CTR mode then Carter-Wegman MAC

Deterministic Encryption

Deterministic Encryption

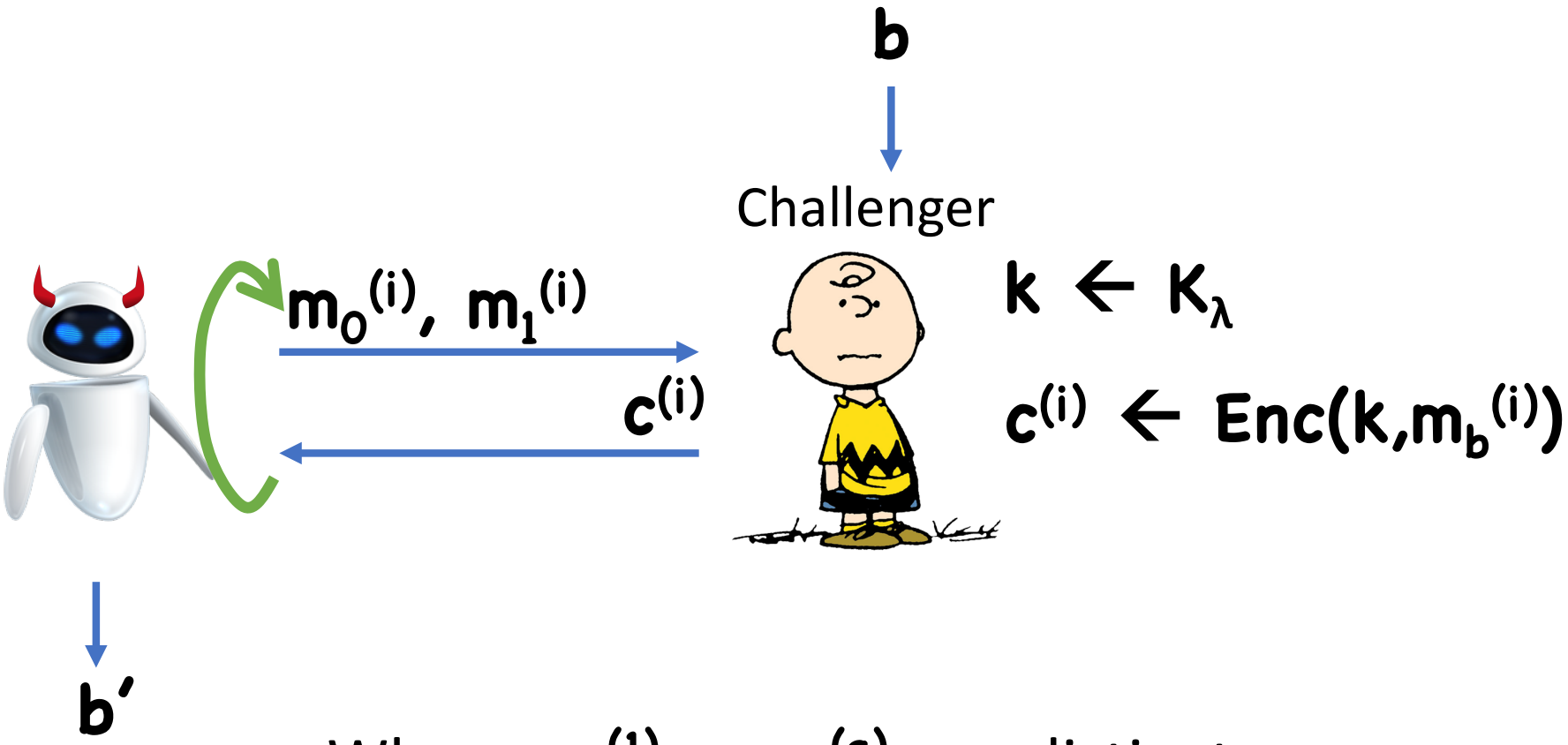
So far, we have insisted on CPA/CCA/Auth Enc security, which implies scheme must be randomized

However, sometimes deterministic encryption is necessary

- E.g. encrypting database records

How to resolve discrepancy?

Deterministic CPA Security



Where $m_1^{(1)},\ldots,m_1^{(q)}$ are distinct
and $m_1^{(1)},\ldots,m_1^{(q)}$ are distinct

Achieving Det. CPA Security

Idea? used fixed det. IV

- CTR mode?
- CBC mode?

Better options:

- Derive IV as **IV = PRF(k',m)**
  - If using Auth Enc, get Det. Auth Enc
- Use "large" PRP: **c = PRP(k,m)**
  - Can get Det. Auth Enc by padding message

Collision Resistant Hashing

Expanding Message Length for MACs

Suppose I 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 $h:\{0,1\}^n \rightarrow \{0,1\}^m$ be a function, $m << n$

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

Correctness is straightforward

Security?

- Pigeonhole principle: $\exists\, m_0 \neq m_1$ s.t. $h(m_0)=h(m_1)$

Collision Resistance Hashing

Syntax:

- Key Space K_λ (typically $\{0,1\}^\lambda$)
- Domain D_λ (typically $\{0,1\}^{l(\lambda)}$ or $\{0,1\}^*$)
- Range R_λ (typically $\{0,1\}^{l'(\lambda)}$ where $l'(\lambda)<l(\lambda)$)
- Function $H: K_\lambda \times D_\lambda \rightarrow R_\lambda$

Security: for every PPT  , $\exists$ negl ε

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

Collision Resistance and MACs

Let **h(m) = H(k,m)** for a random choice of **k**

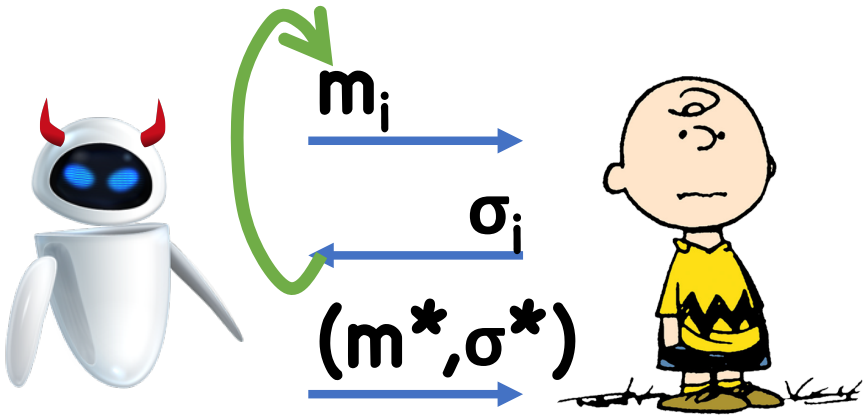
MAC′(k_{MAC},m) = MAC(k_{MAC}, h(m))
Ver′(k_{MAC},m,σ) = Ver(k_{MAC}, h(m), σ)

For now, think of **k** as part of key for **MAC′**

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

Proof

Hybrid 0



$k_H \leftarrow K_H$

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

$t_i \leftarrow H(k_H,m_i)$

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

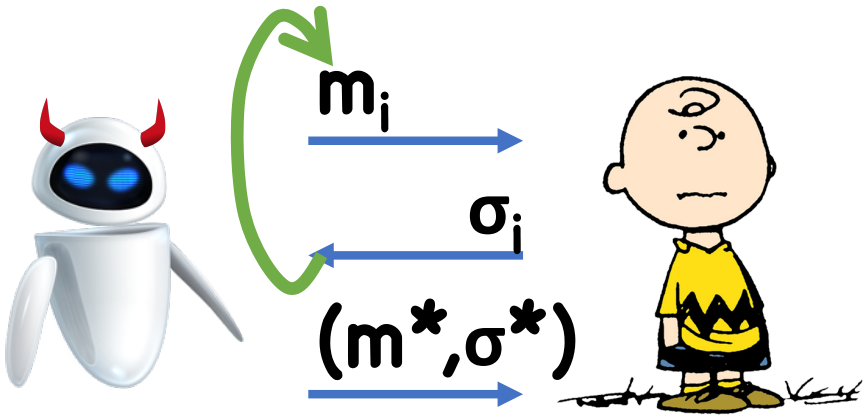
Output 1 iff:

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

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

Proof

Hybrid 1



$k_H \leftarrow K_H$

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

$t_i \leftarrow H(k_H,m_i)$

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

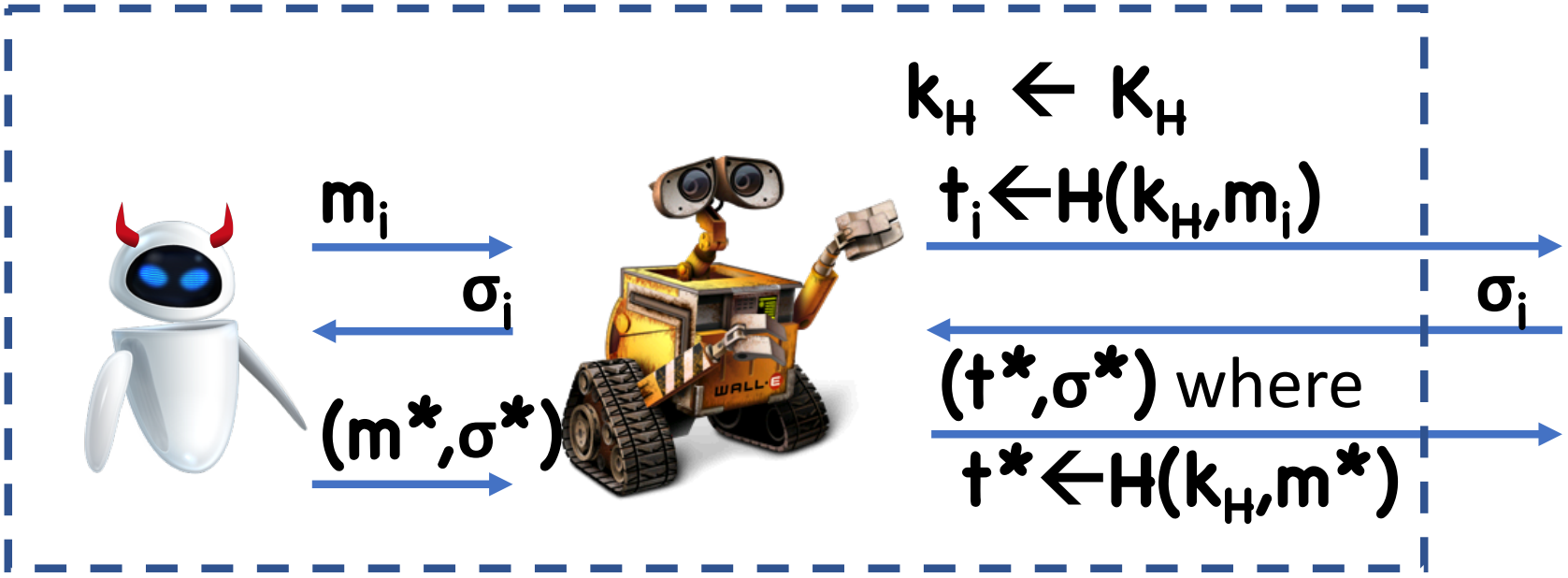
Output 1 iff:

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

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

Suppose $t^* = t_i$

Then (m_i, m^*) is a collision for H

Theory vs. Practice

Hashing key is *public*: even adversary can know it

- What role does it play?

In practice,

- Hash functions are un-keyed
- Hash functions have fixed output size, say $\{0,1\}^{256}$

Problem?

We said 128 bit security is usually enough

Why 256-bit outputs?

Birthday Attack

If the range of a hash function is 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

≈ (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

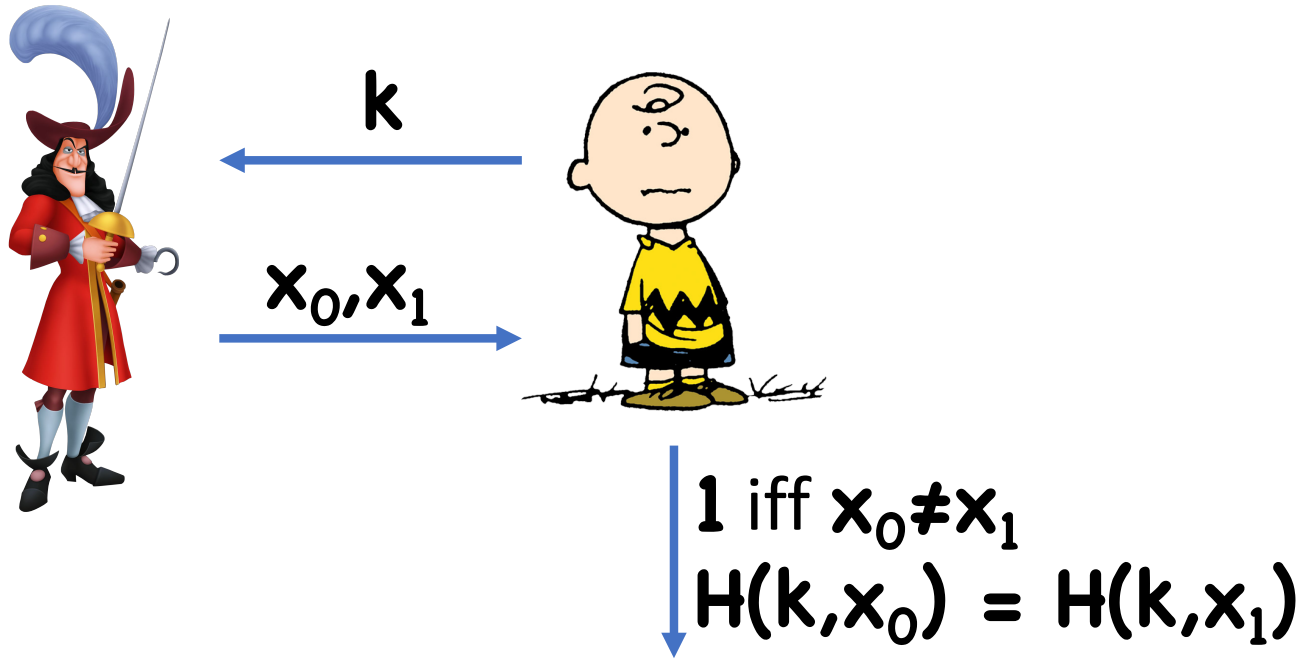
Birthday Attack

Space?

Possible to reduce memory requirements to $\boldsymbol{O(1)}$

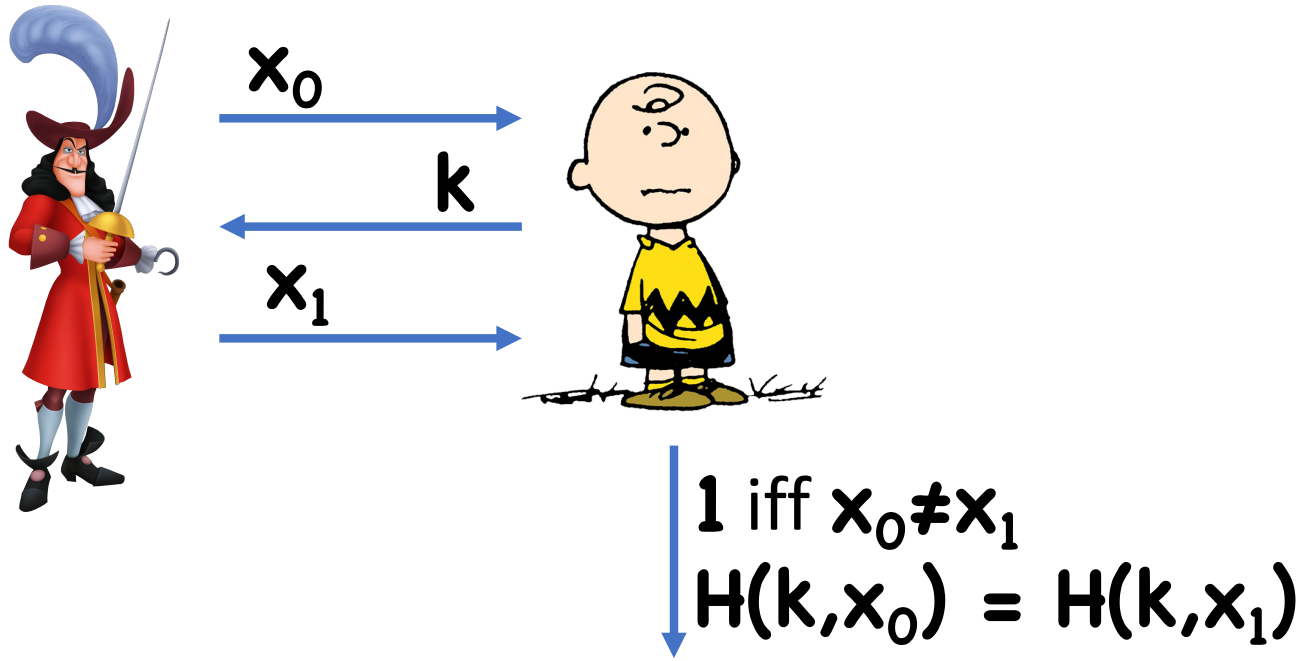
Other Security Notions

Collision resistance as a game:



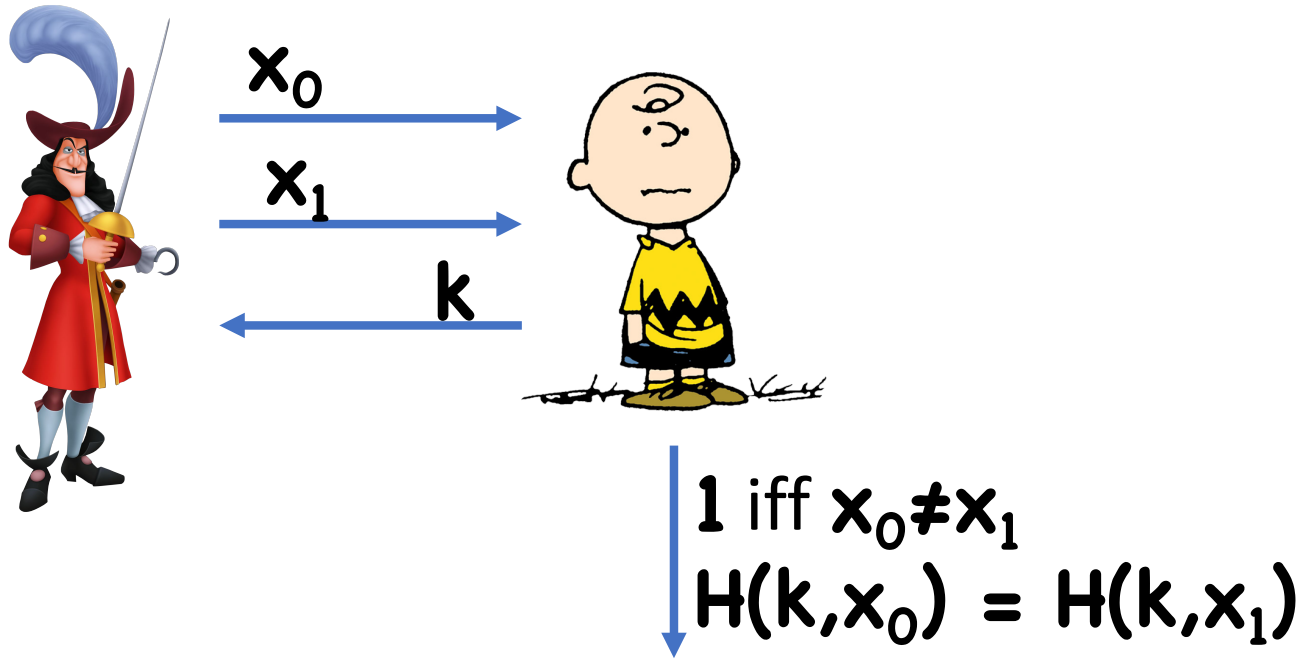
Other Security Notions

2nd Preimage Resistance (or target collision resistance):



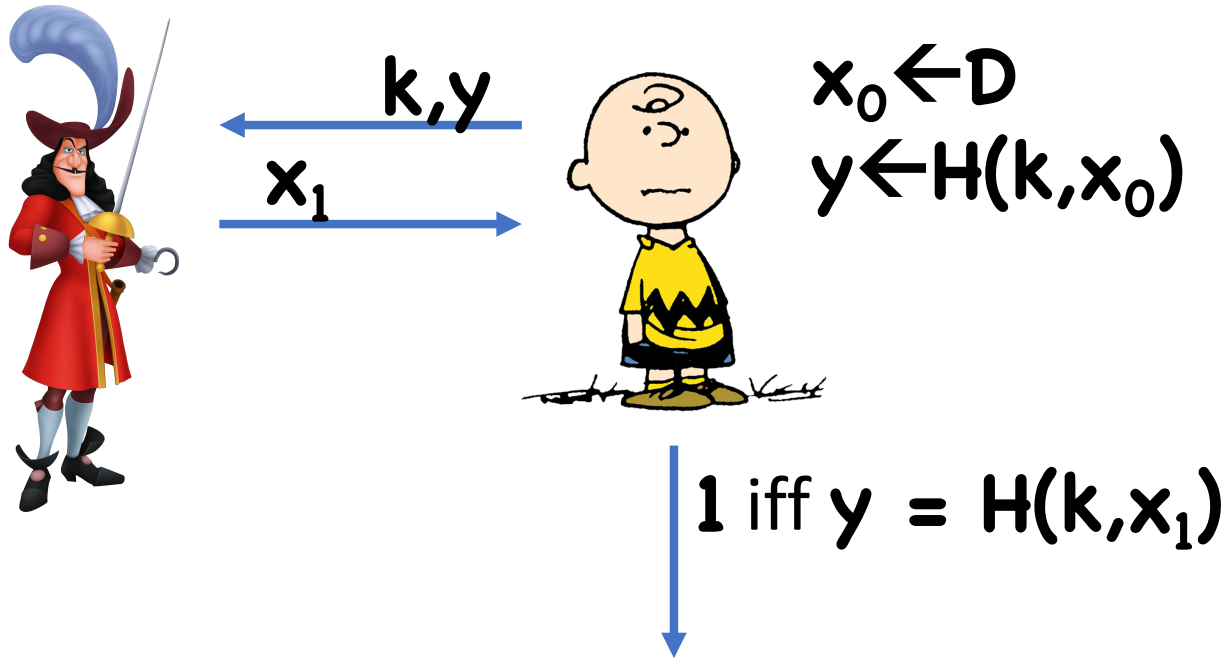
Other Security Notions

2-Universal:



Other Security Notions

One-wayness (or pre-image resistance):



Implications

Collision Resistance

2nd Pre-image Resistance

One-wayness

Domain Extension

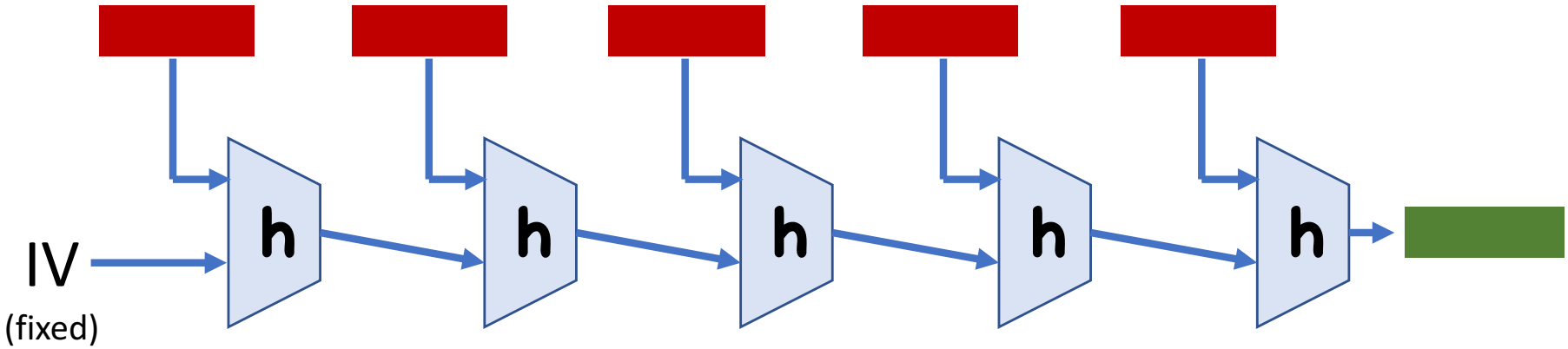
Goal: given h that compresses small inputs, construct H that compresses large inputs

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

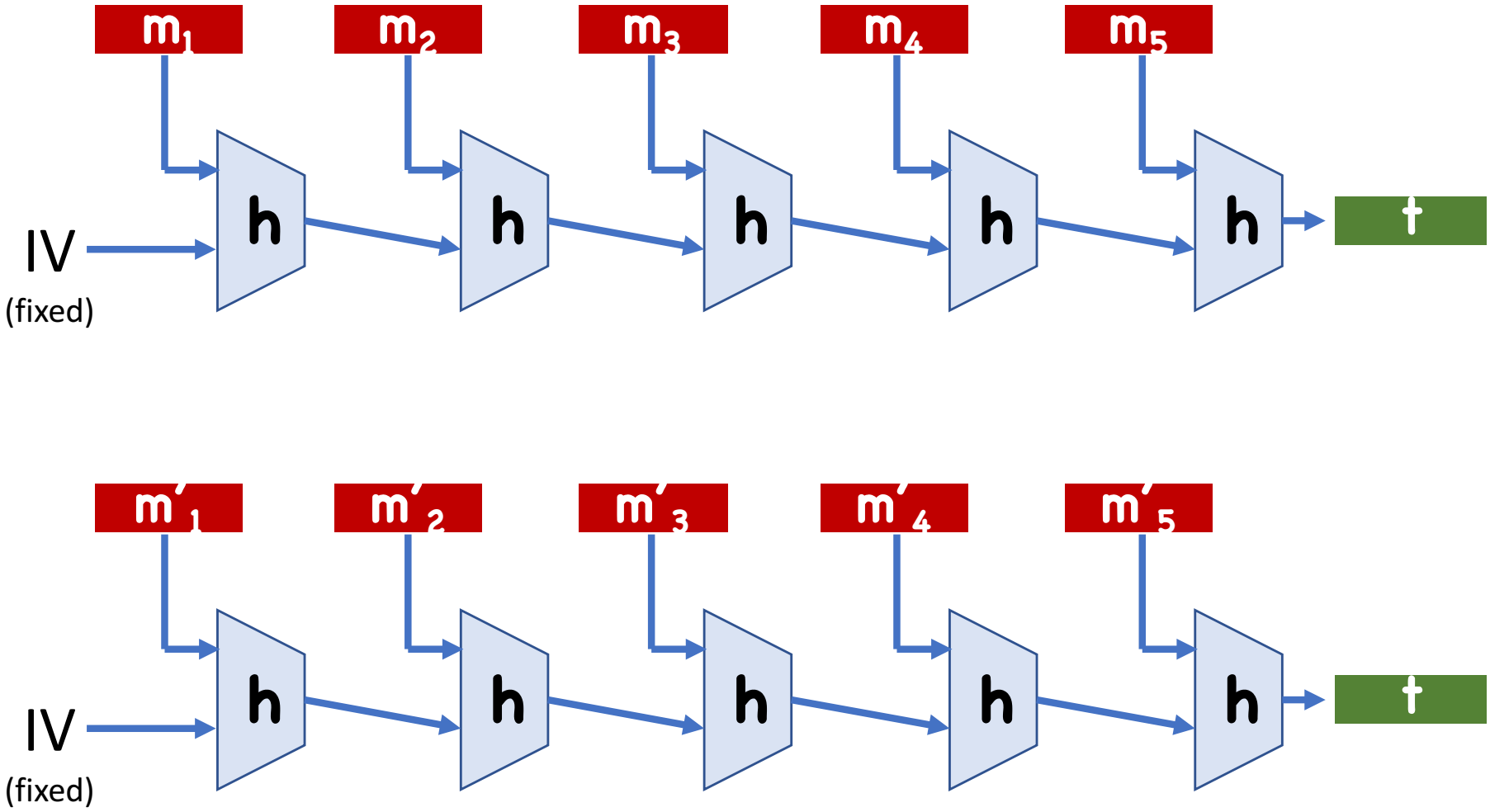
- Called compression functions
- Easier to design

Merkle-Damgard

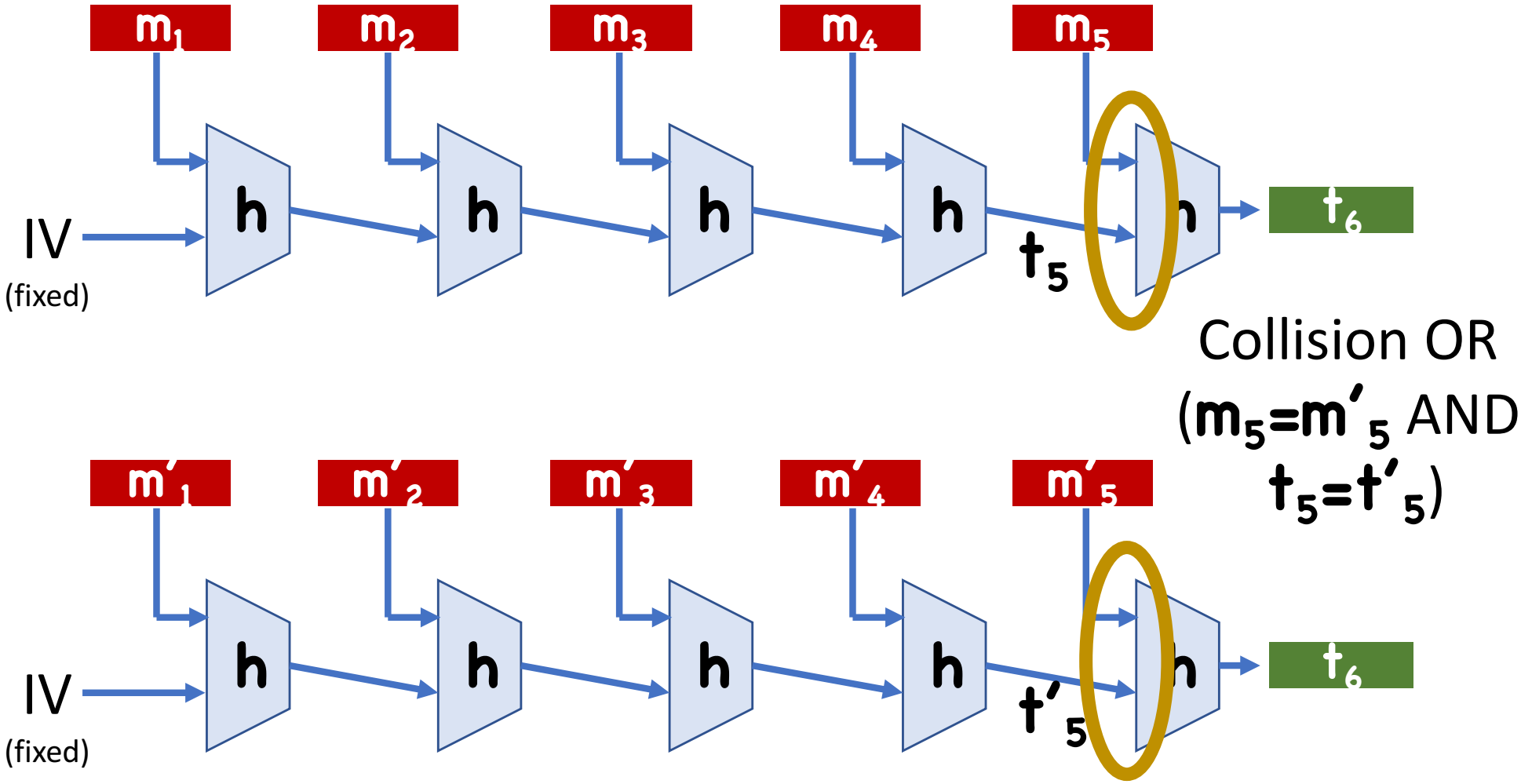


Theorem: If an adversary knows a collision for fixed-length Merkle-Damgard, he can also compute a collision for **h**

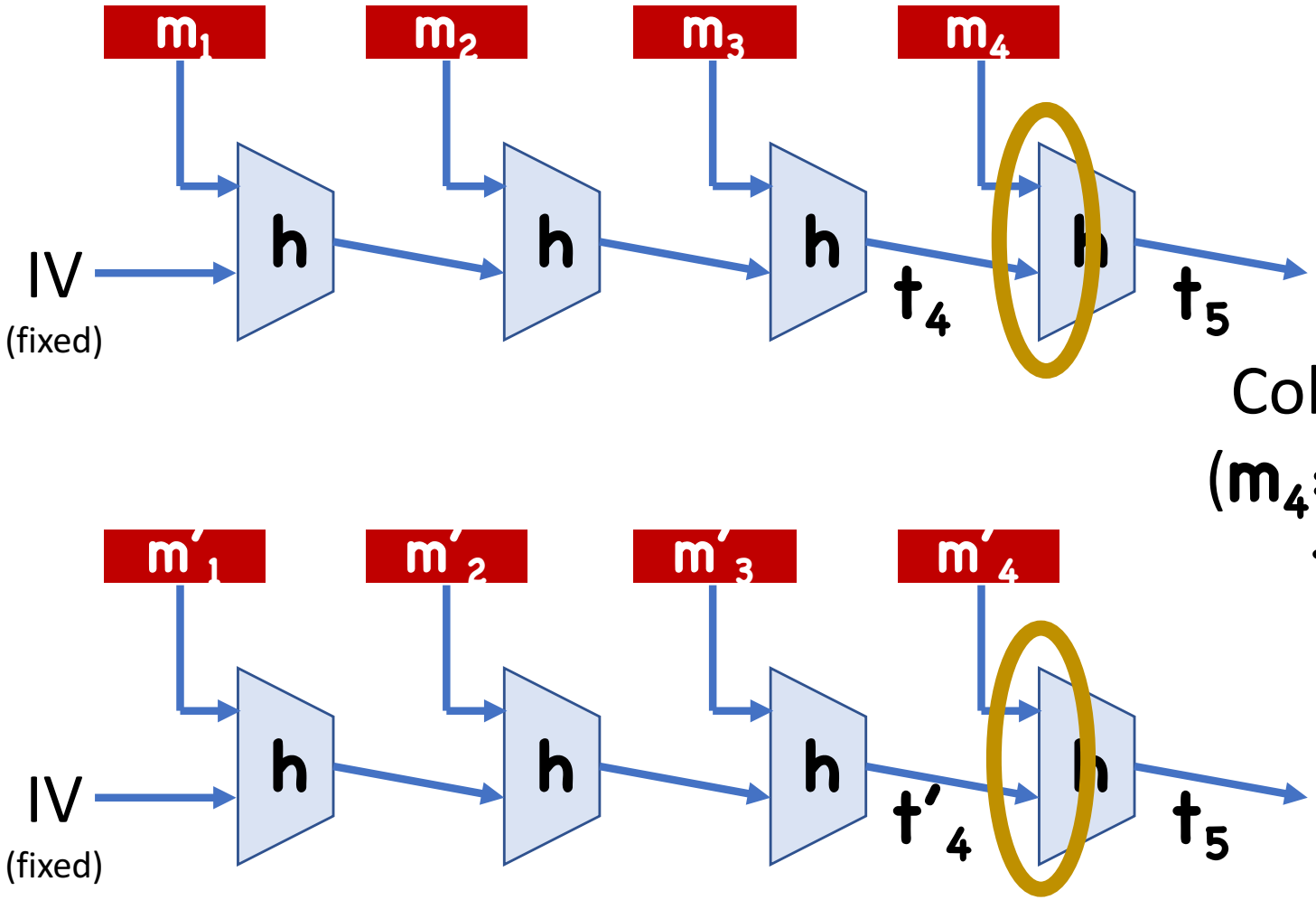
Proof



Proof

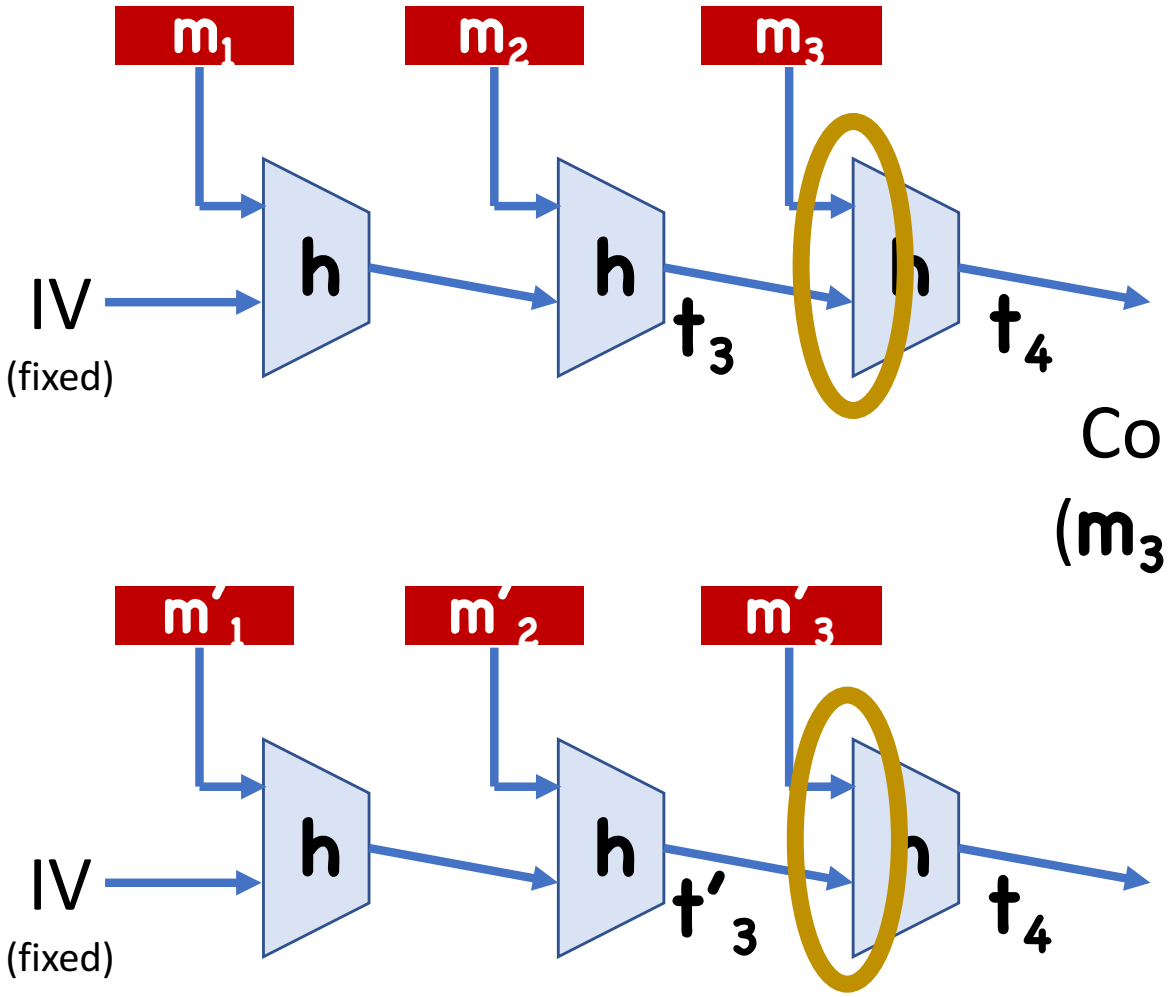

m1
m2
m3
m4
m5
h
h
h
h
h
IV
(fixed)
t5
t6
Collision OR
(m5=m'5 AND
t5=t'5)
m'1
m'2
m'3
m'4
m'5
h
h
h
h
h
IV
(fixed)
t'5
t6

Proof



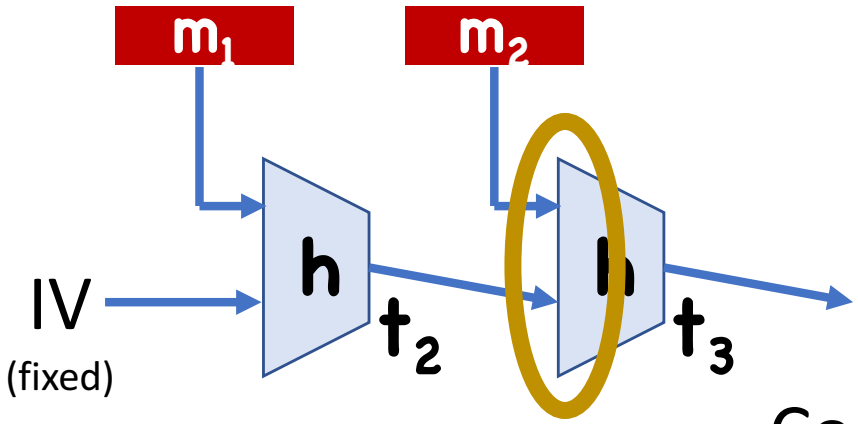
Collision OR
($m_4 = m'_4$ AND
$t_4 = t'_4$)

Proof

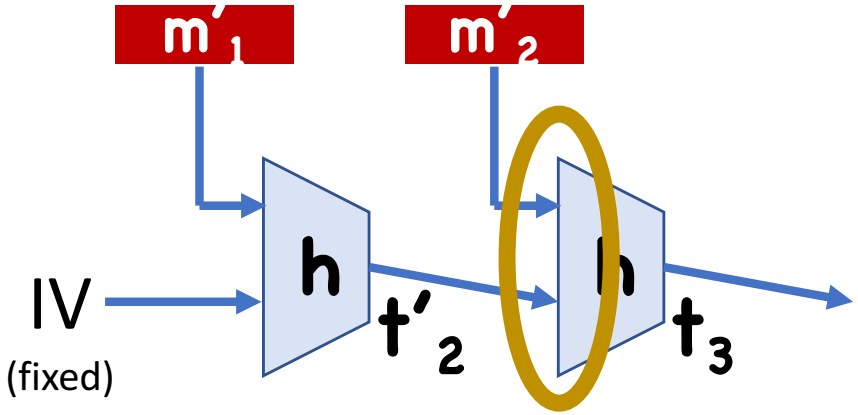


Collision OR
($m_3=m'_3$ AND
$t_3=t'_3$)

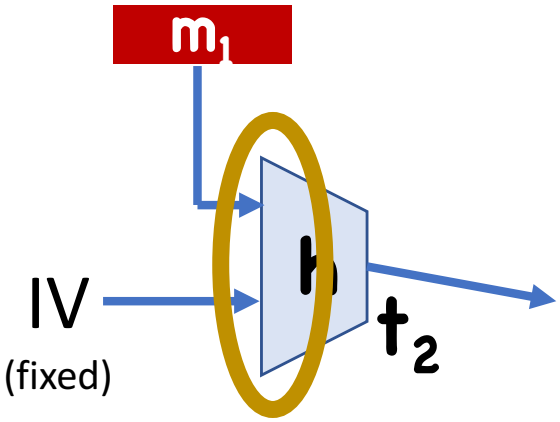
Proof



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

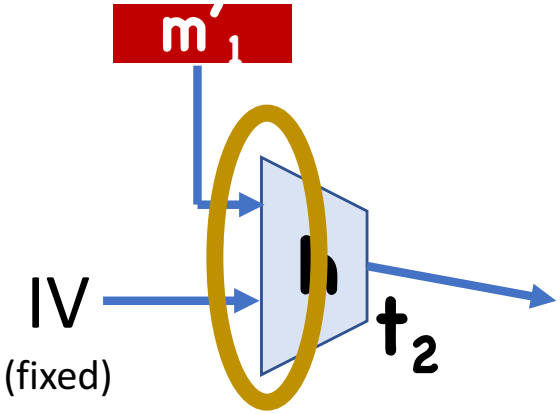


Proof



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 I know an input **x** such that **h(x||IV) = IV**?

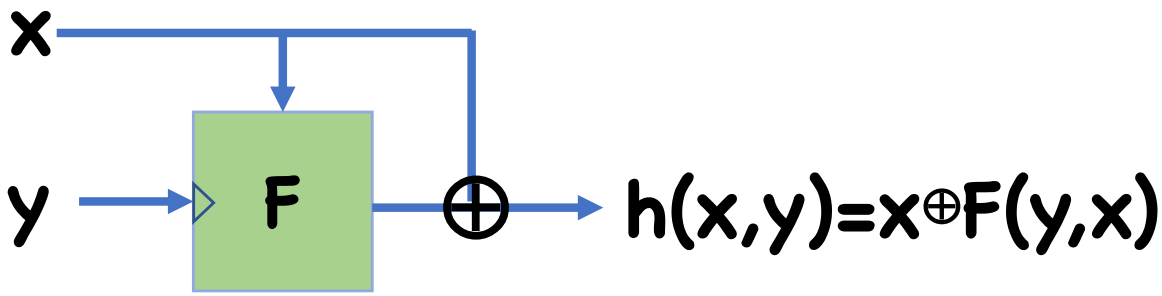
Need proper padding

- Ex: append message length to end of message

Constructing **h**

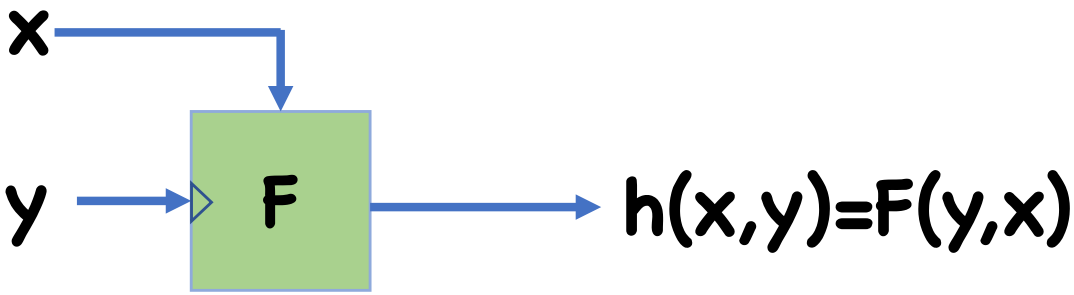
Common approach: use block cipher

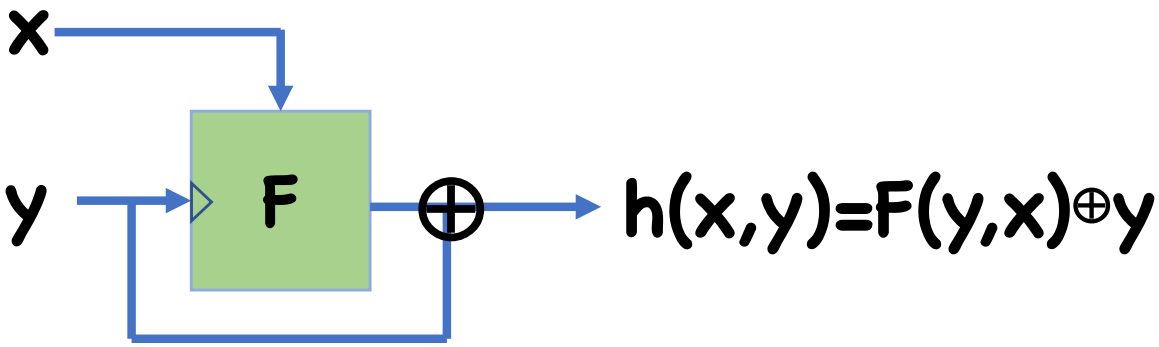
Davies-Meyer



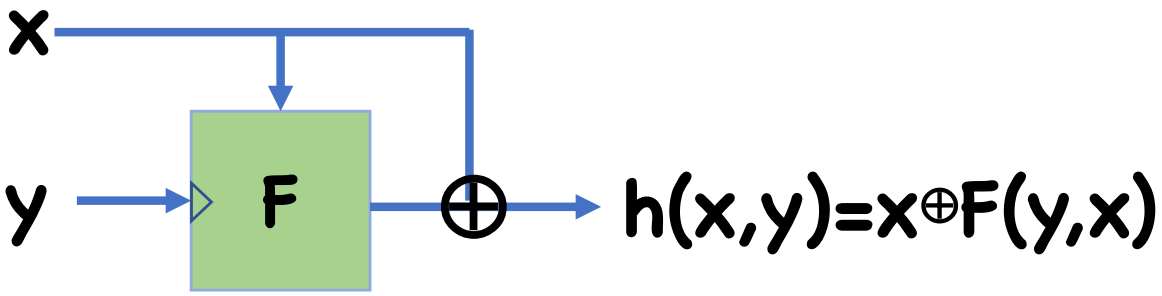
Constructing **h**

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

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 very different

- Compression function built using unkeyed permutation
- Extension to arbitrary lengths via “sponge construction”

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)

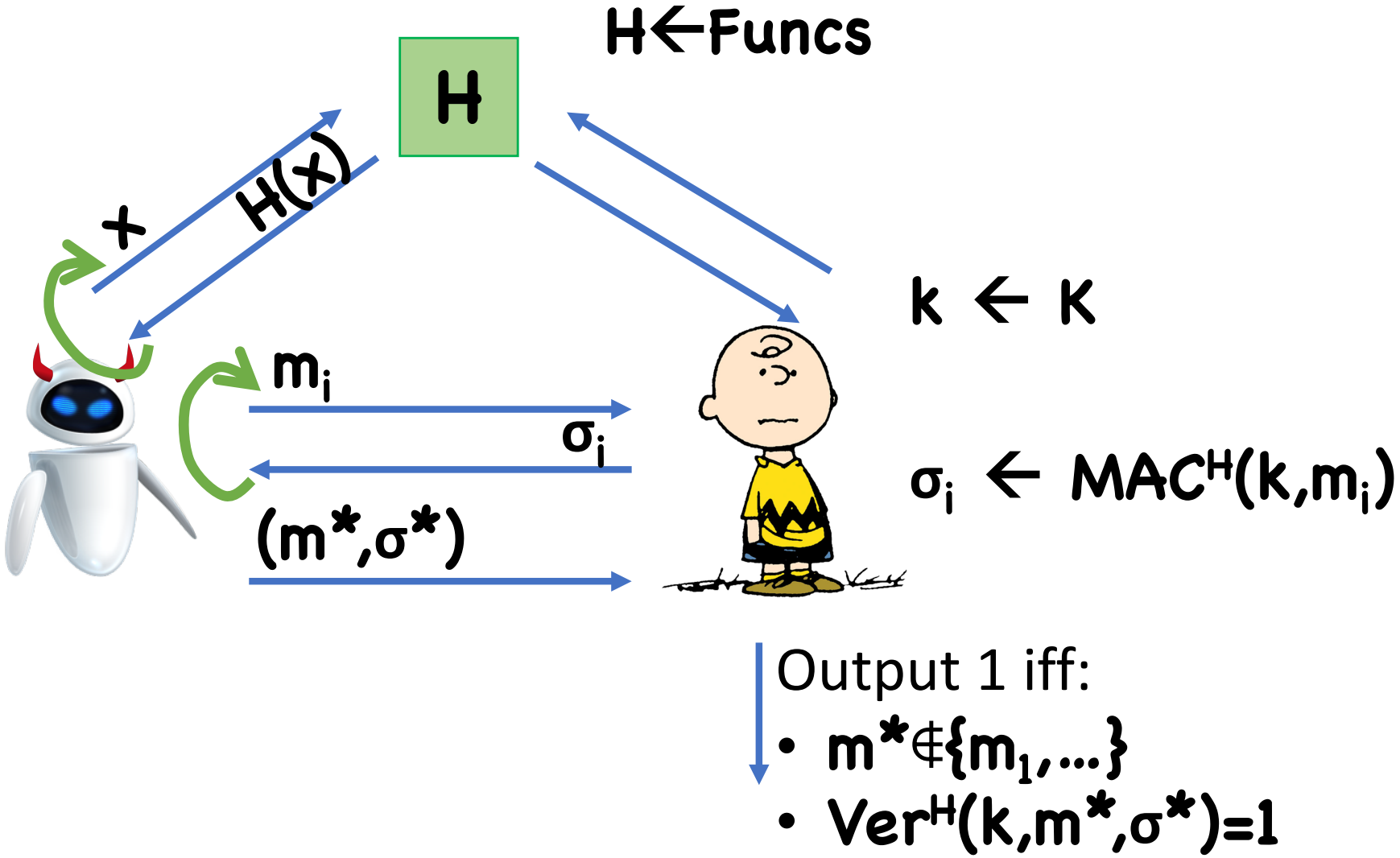
A query to **H** has a time cost of 1

MAC in ROM

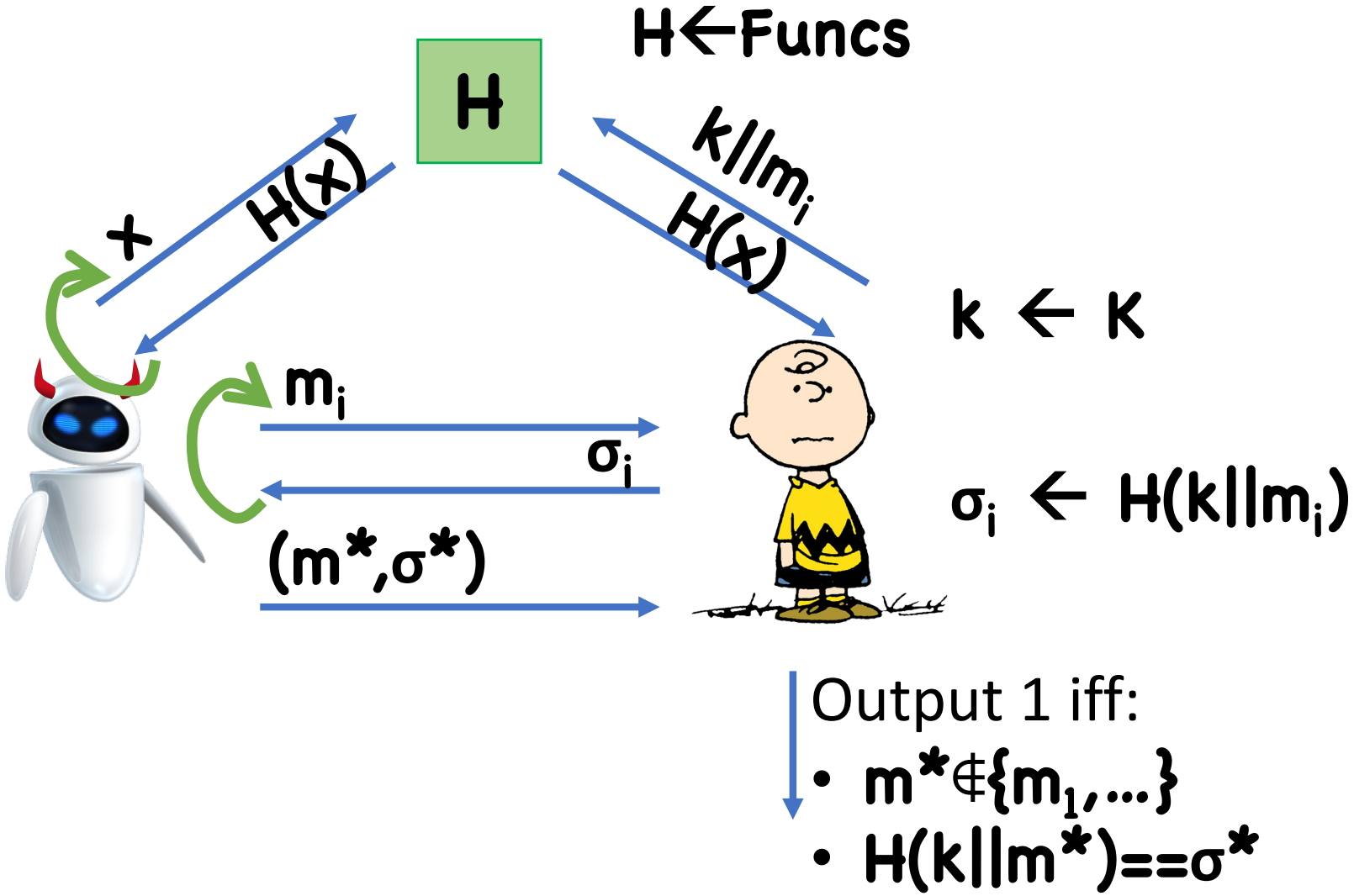
$\mathbf{MAC^H(k,m) = H(k||m)}$
$\mathbf{Ver^H(k,m,\sigma) = (H(k||m) == \sigma)}$

Theorem: H(k || m) is a secure MAC in the random oracle model

Meaning


H←Funcs
H
x
H(x)
k ← K
m_i
σ_i
$\sigma_i \leftarrow MAC^H(k,m_i)$
(m^*,σ^*)
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 $k||m_i$ for some i

- Only way to learn k is to query H on $k||m_i$

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

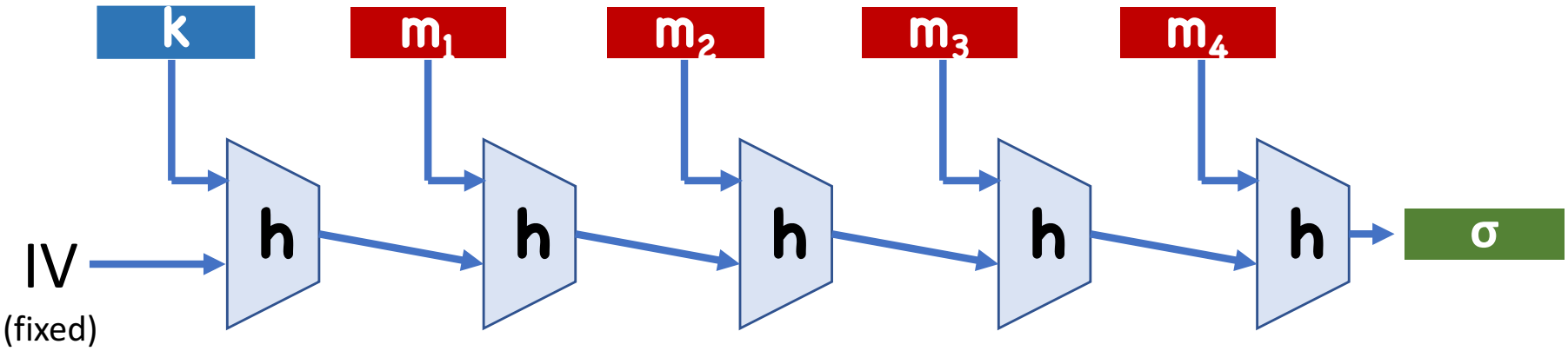
Therefore, a ROM proof is a heuristic argument for security

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

When the ROM Fails

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

Instantiate with Merkle-Damgard (variable length)?



When the ROM Fails

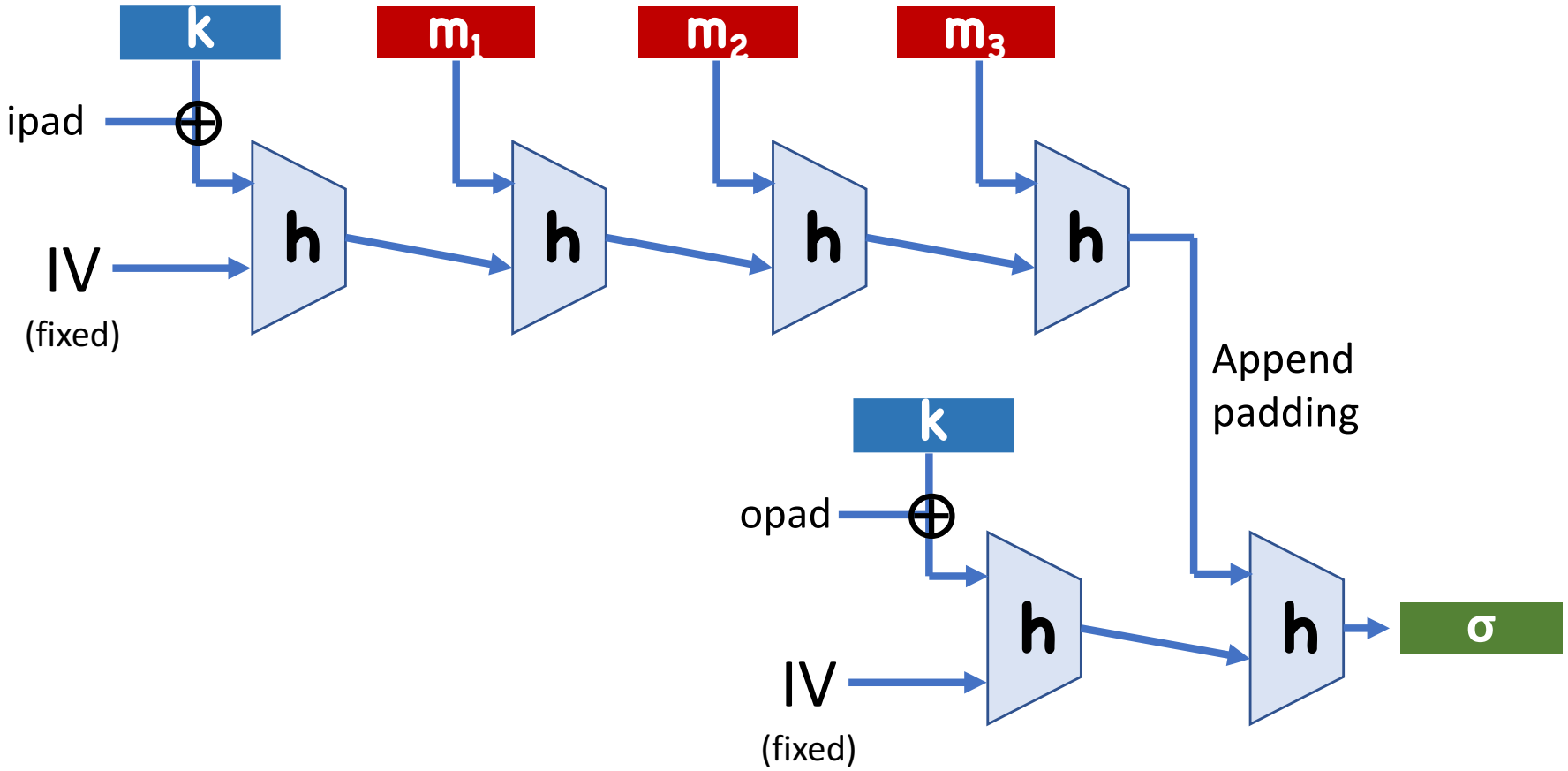
ROM does not apply to regular Merkle-Damgard

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

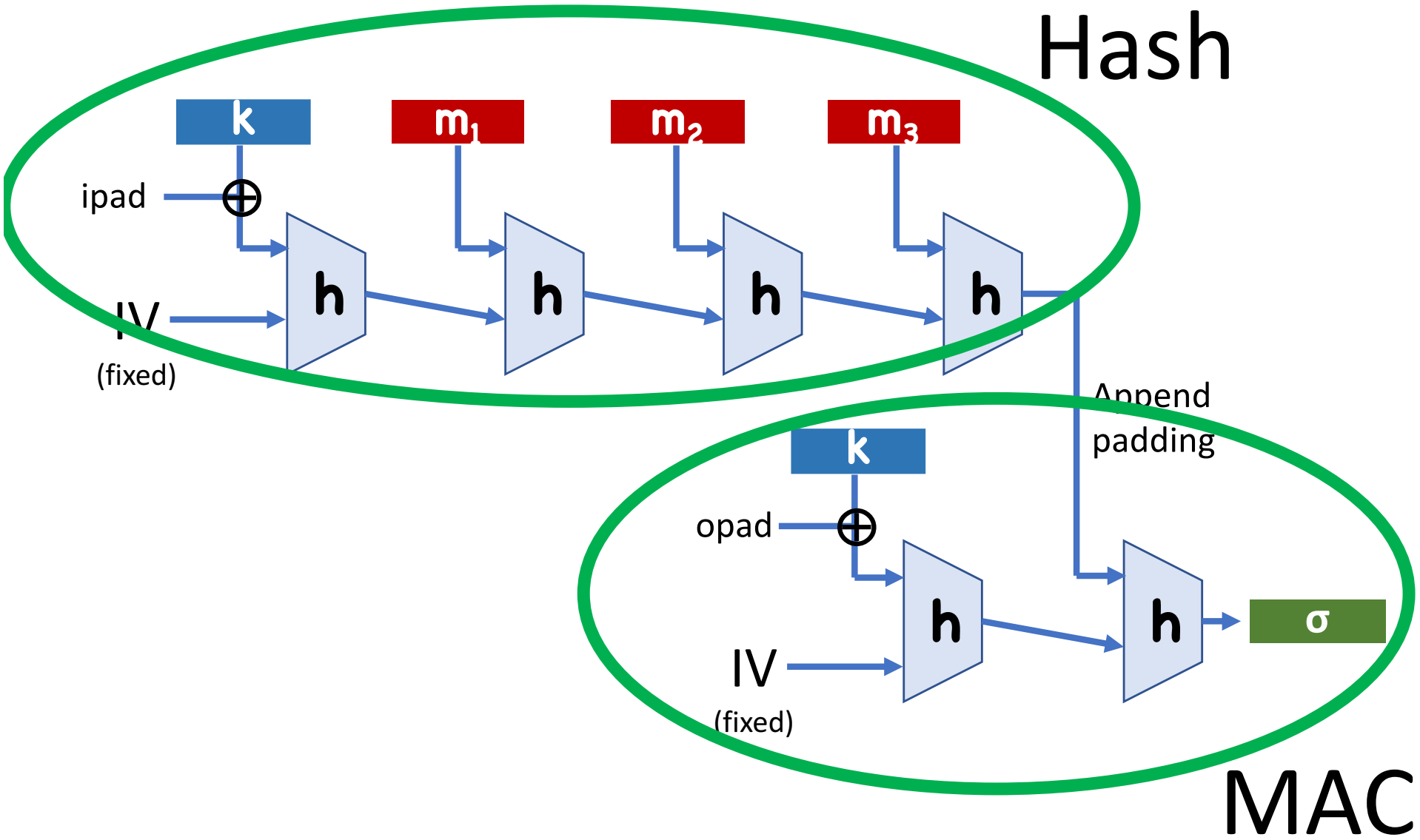
Takeaway: be careful about using ROM for non-”monolithic” hash functions

- Though still possible to pad MD in a way that makes is 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

After Spring Break

Wrap up symmetric key cryptography

- Commitment schemes, relationships between symmetric primitives

Number-theoretic constructions

Public key cryptgoraphy