

COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2020

Announcements

HW4 Due April 2nd

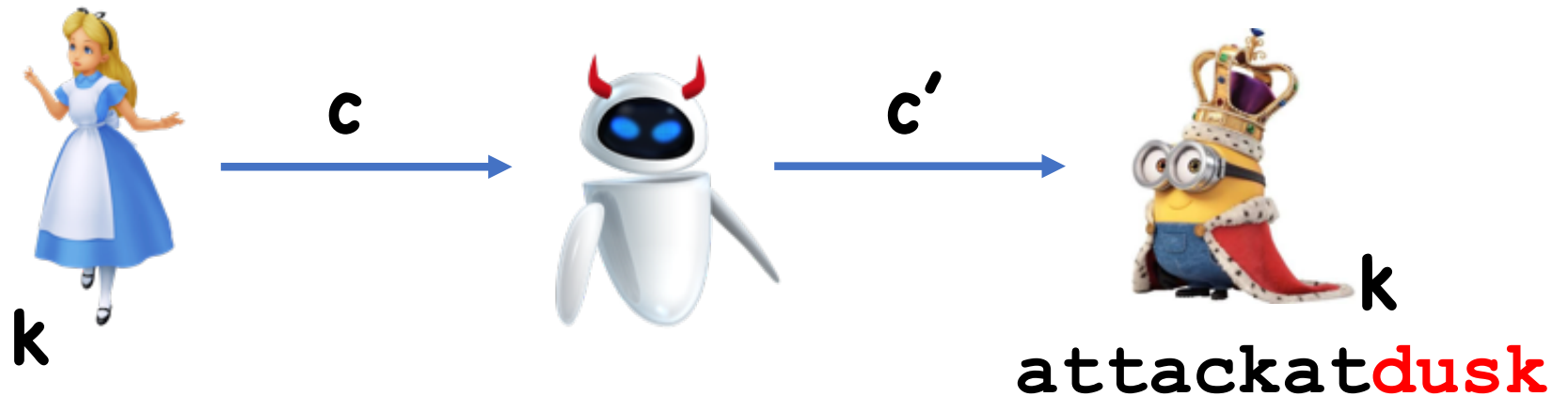
Mark's OH → Fridays 11am (Eastern)

Previously on COS 433...

Authenticated Encryption

Authenticated Encryption

`attackatdawn`



Goal: Eve cannot learn nor change plaintext

- Authenticated Encryption will satisfy two security properties

Syntax

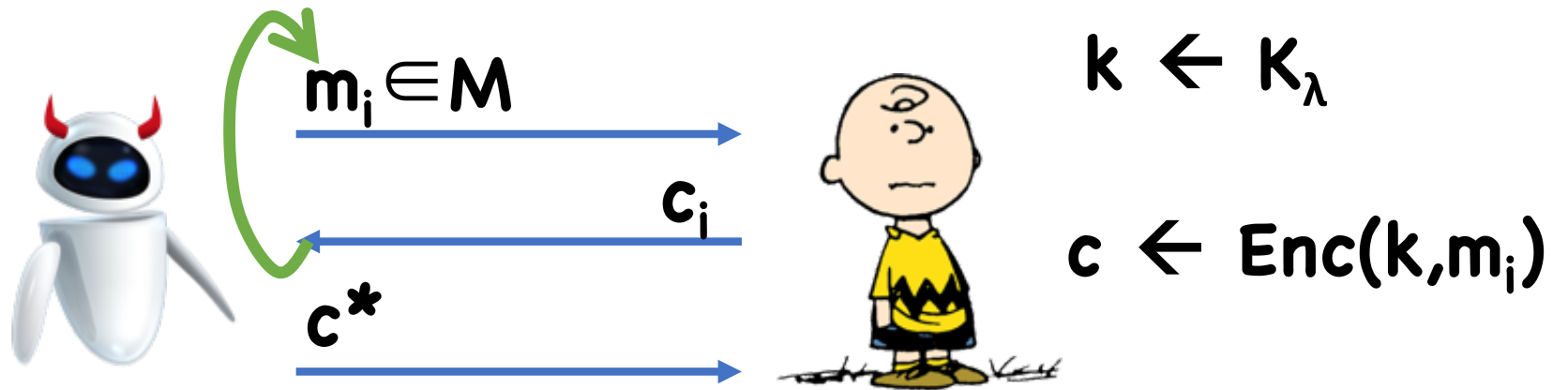
Syntax:

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

Correctness:

- For all $k \in K$, $m \in M$,
 $\Pr[\text{Dec}(k, \text{Enc}(k, m)) = m] = 1$

Unforgeability



- Output 1 iff:
- $c^* \notin \{c_1, \dots\}$
 - $\text{Dec}(k, c^*) \neq \perp$

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

Today

Authenticated encryption, continued

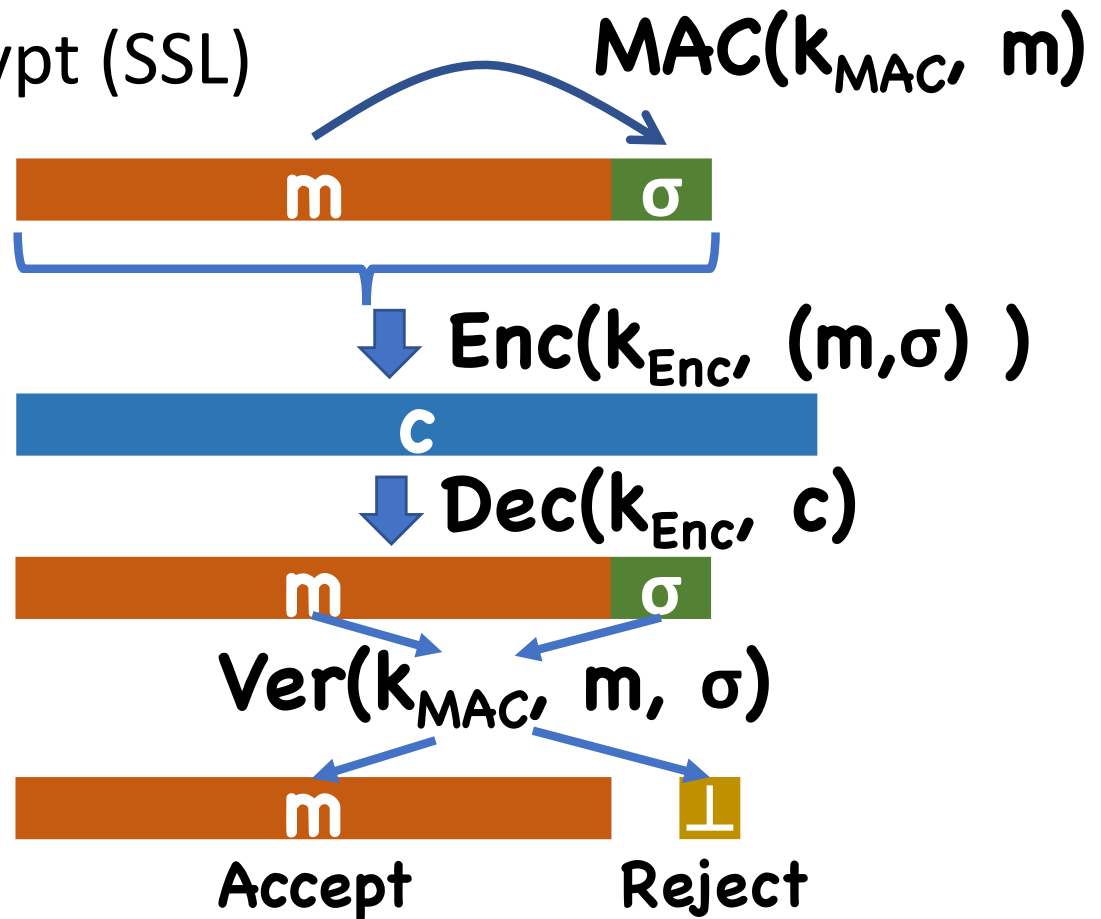
Hash functions

Constructing Authenticated Encryption

Three possible generic constructions:

1. MAC-then-Encrypt (SSL)

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

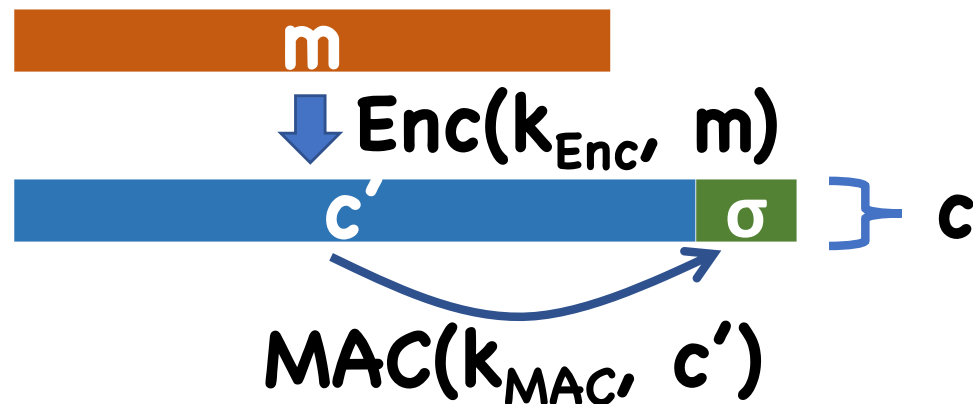


Constructing Authenticated Encryption

Three possible generic constructions:

2. Encrypt-then-MAC (IPsec)

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

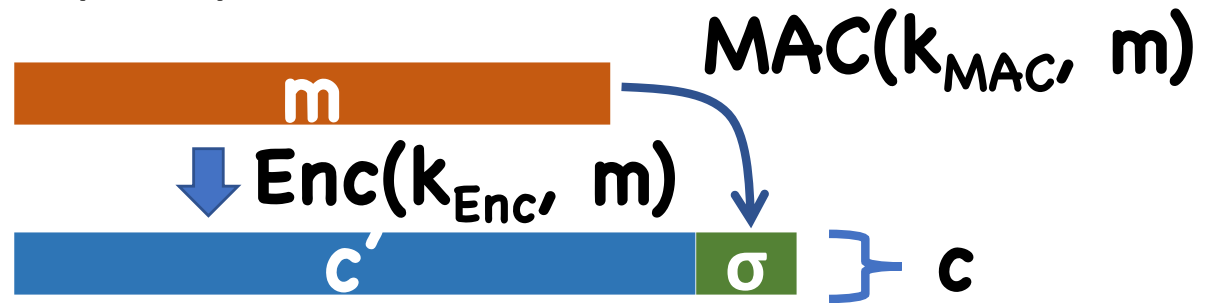


Constructing Authenticated Encryption

Three possible generic constructions:

3. Encrypt-and-MAC (SSH)

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



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

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: $\text{Enc}(k,m) = (0, \text{Enc}'(k,m))$
 $\text{Dec}(k, (b,c)) = \text{Dec}'(k,c)$



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

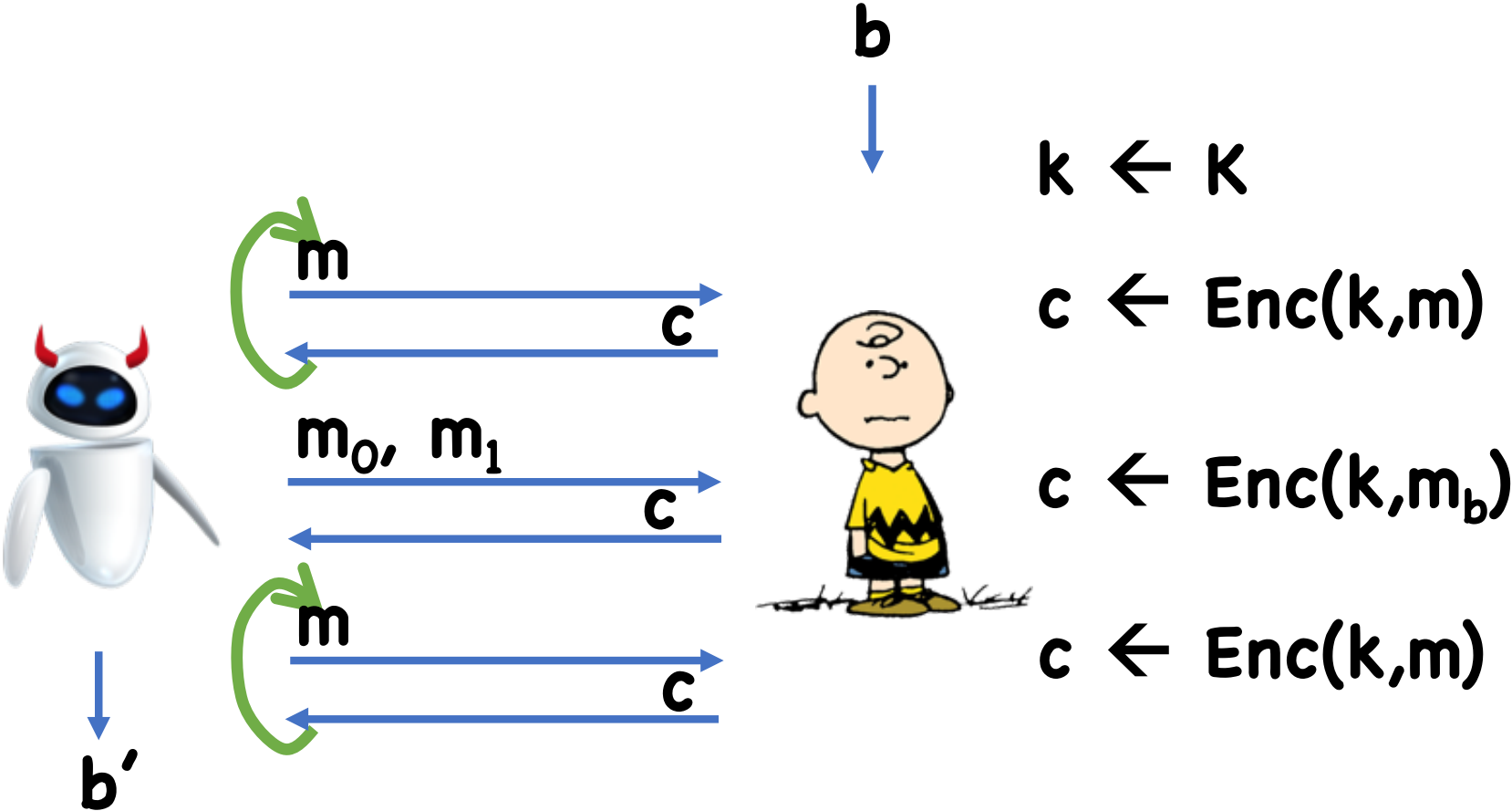
Chosen Ciphertext Attacks

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

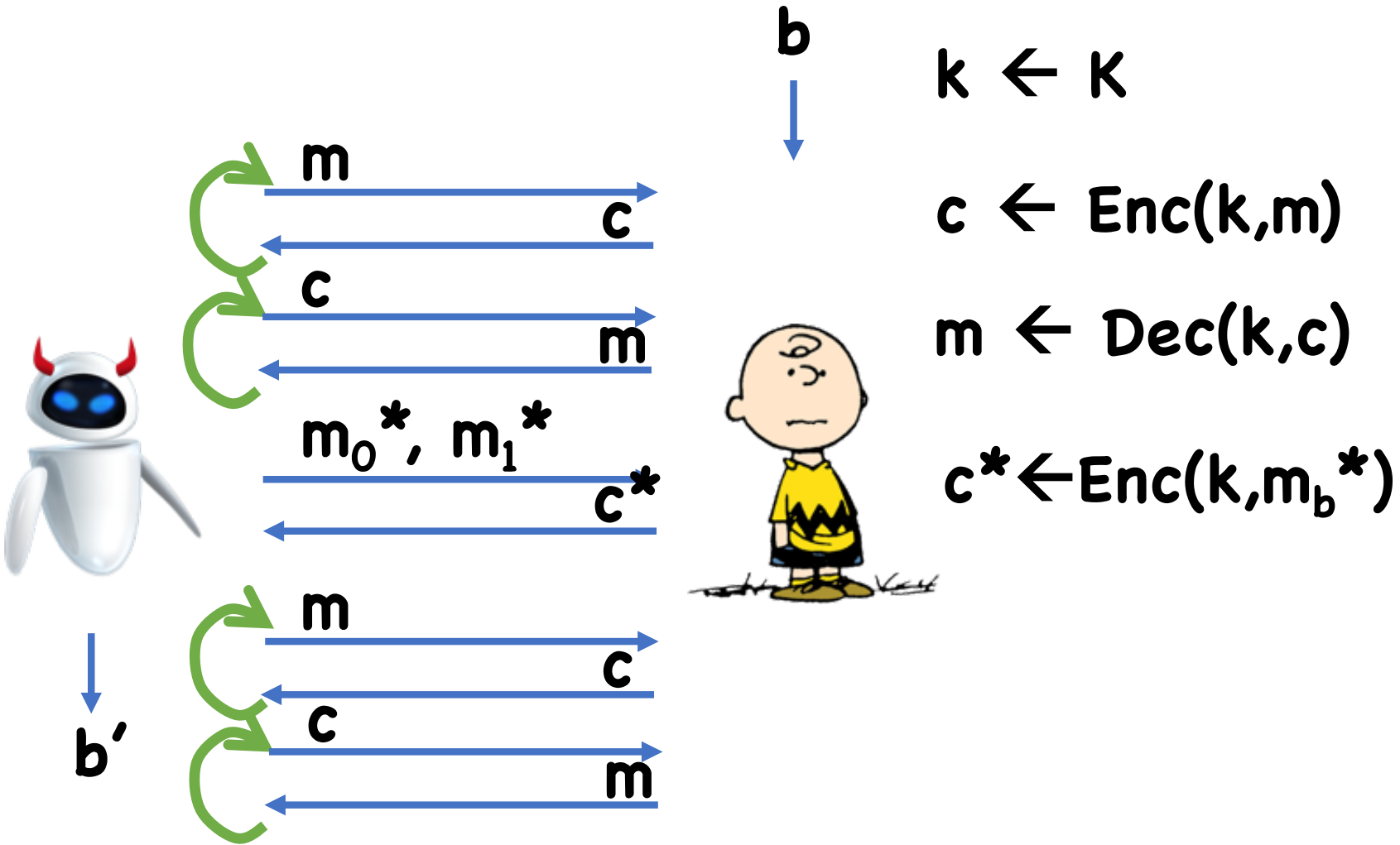
Therefore, want security even if adversary can mount decryption queries

Chosen Plaintext Security

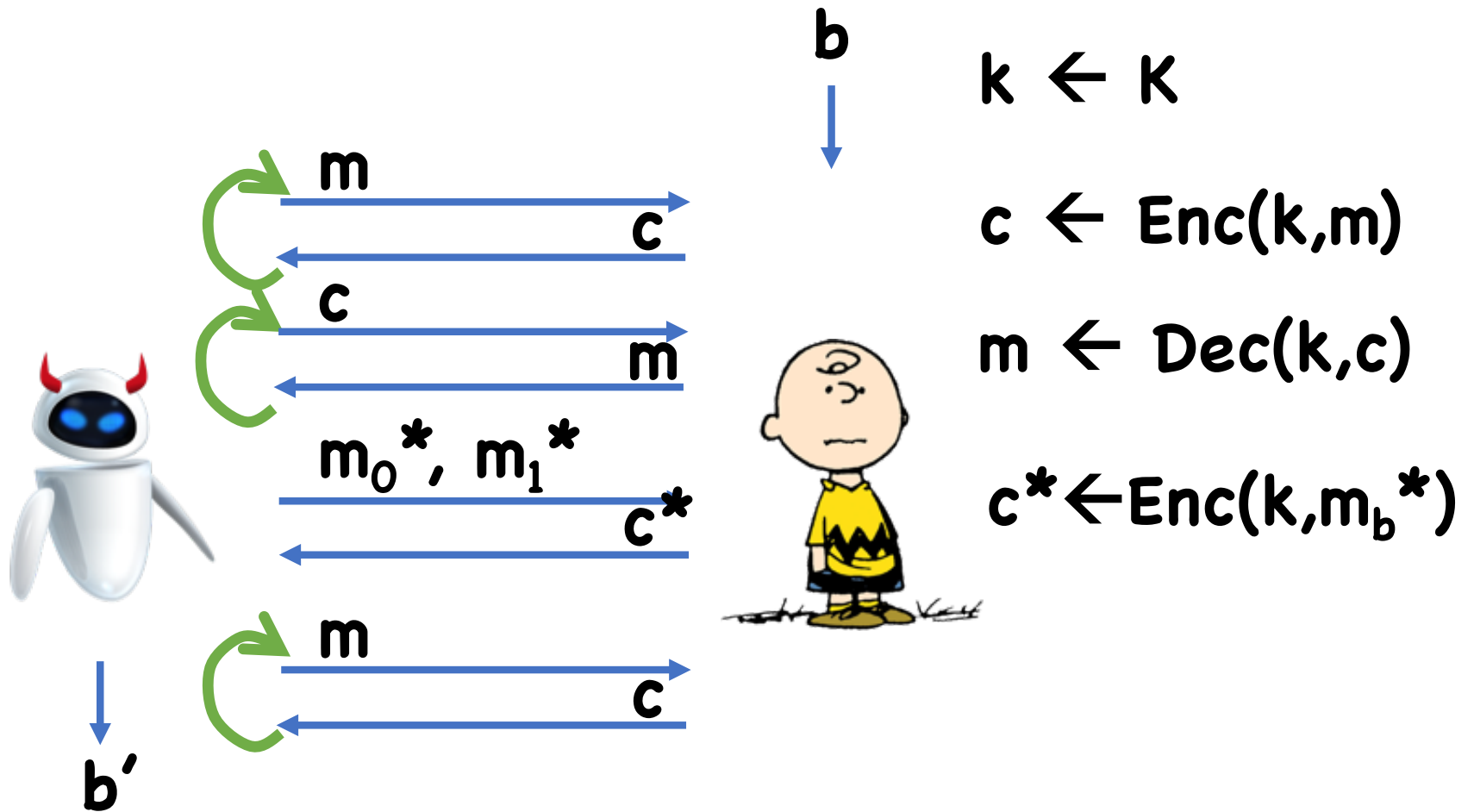


$\text{CPA-Exp}_b(\text{robot}, \lambda)$

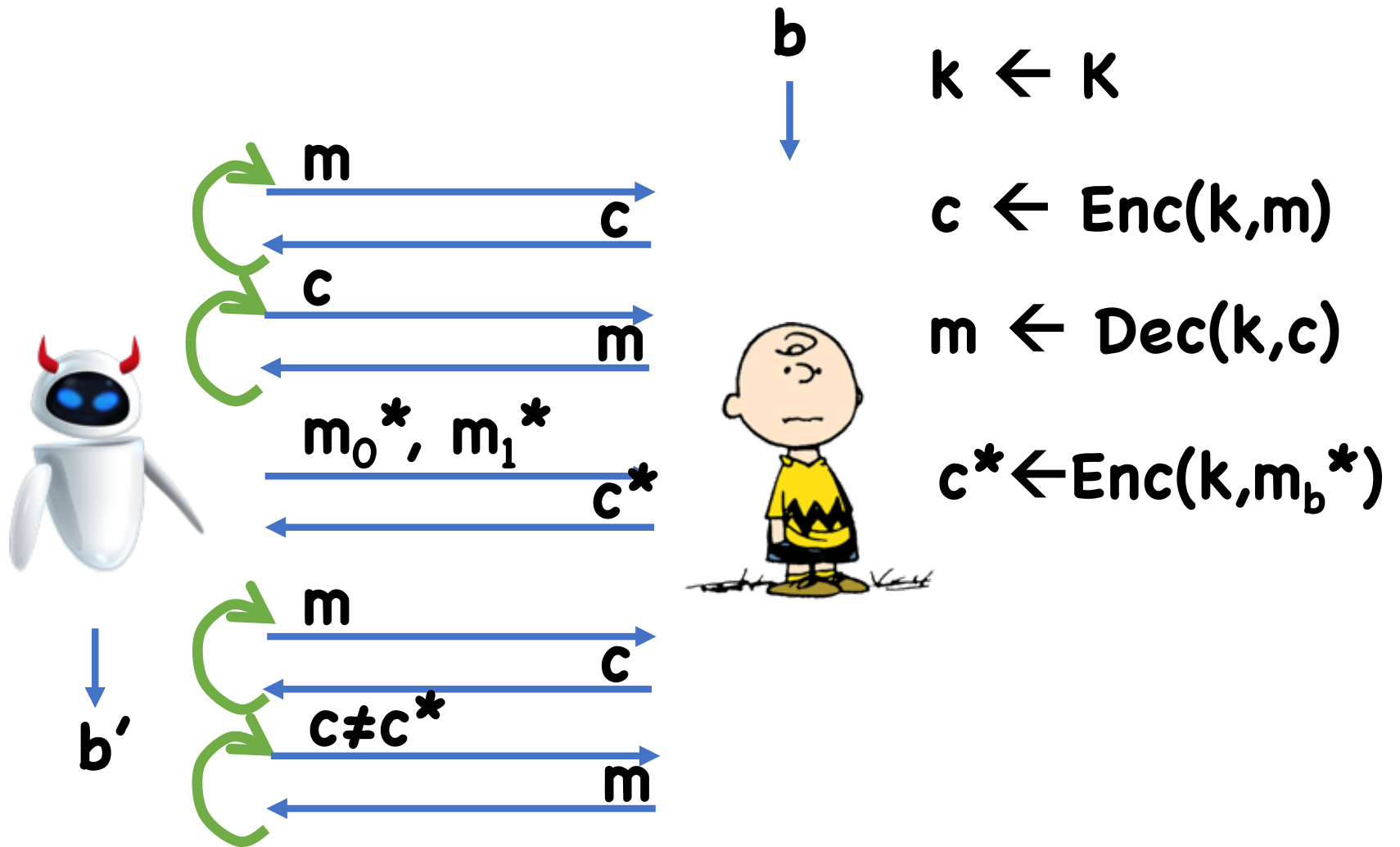
Chosen Ciphertext Security?



Lunch-time CCA (CCA1)



Full CCA (CCA2)



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, $n \ll l$

$$\text{MAC}'(k,m) = \text{MAC}(k, h(m))$$

$$\text{Ver}'(k,m,\sigma) = \text{Ver}(k, h(m), \sigma)$$

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: $n \ll l$

Security?

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

$$\Pr[H(x_0) = H(x_1) \wedge x_0 \neq x_1 : (x_0, x_1) \leftarrow \langle \cdot \rangle] < \epsilon(\lambda)$$

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 \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 collision resistant if, for all  running in polynomial time, \exists negligible ϵ such that:

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

Collision Resistance and MACs

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

$$\mathbf{MAC}'(k_{\text{MAC}}, m) = \mathbf{MAC}(k_{\text{MAC}}, \mathbf{h}(m))$$

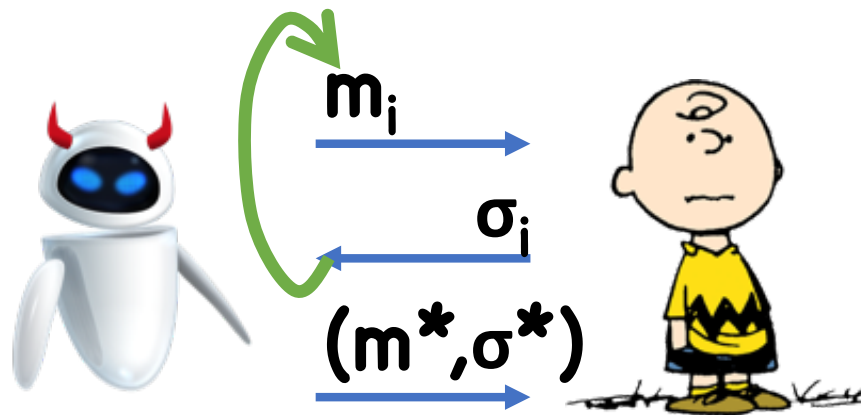
$$\mathbf{Ver}'(k_{\text{MAC}}, m, \sigma) = \mathbf{Ver}(k_{\text{MAC}}, \mathbf{h}(m), \sigma)$$

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

Theorem: If (MAC, Ver) is CMA-secure and H is collision resistant, then $(\text{MAC}', \text{Ver}')$ is CMA secure

Proof

Hybrid 0



$$k_H \leftarrow K_H$$

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

$$t_i \leftarrow H(k_H, m_i)$$

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

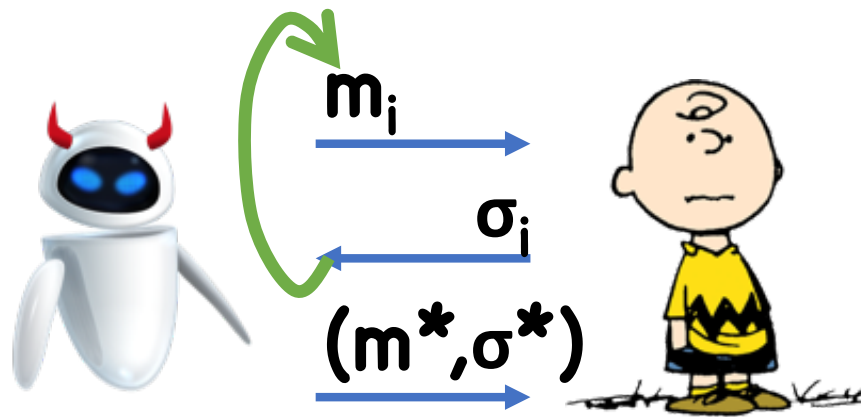
Output 1 iff:

- $m^* \notin \{m_1, \dots\}$

- $Ver(k, t^*, \sigma^*)$ where
 $t^* \leftarrow H(k_H, m^*)$

Proof

Hybrid 1



$$k_H \leftarrow K_H$$

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

$$t_i \leftarrow H(k_H, m_i)$$

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

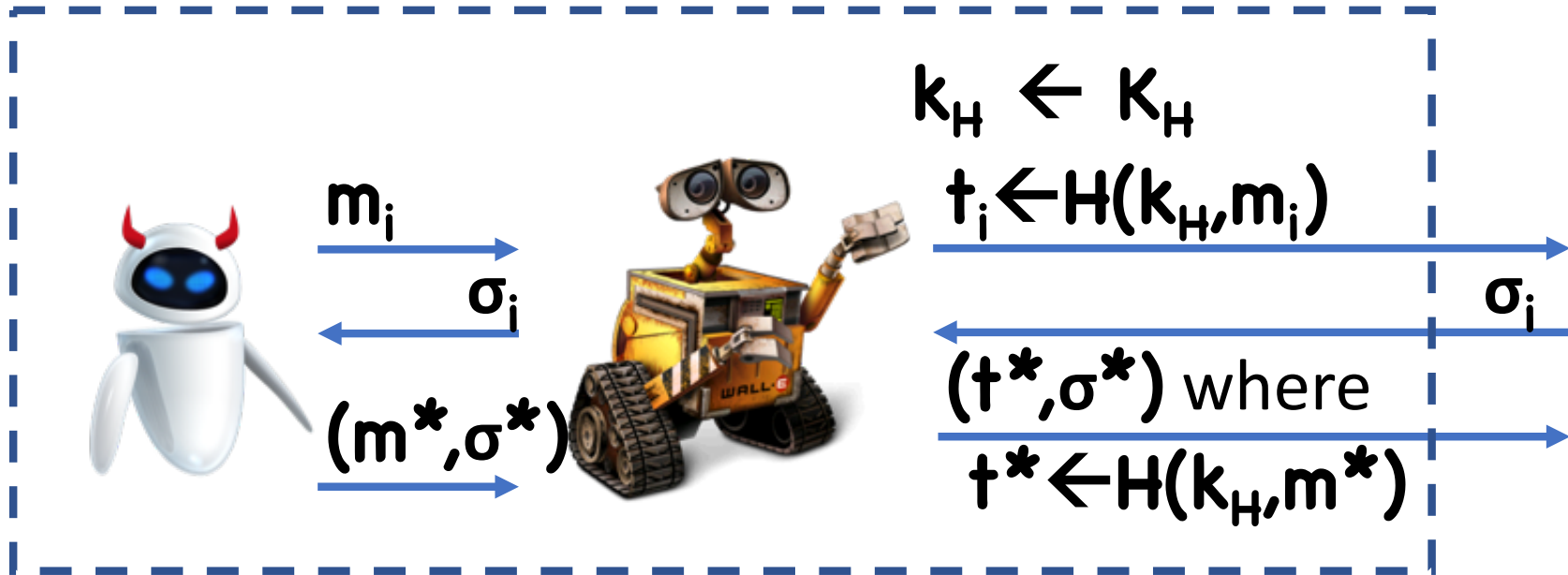
Output 1 iff:

- $t^* \notin \{t_1, \dots\}$

- $Ver(k, t^*, \sigma^*)$ where $t^* \leftarrow H(k_H, m^*)$


Proof

In Hybrid 1, negligible advantage using MAC security



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

Proof

If  succeeds in Hybrid 0 but not Hybrid 1, then

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

Suppose $t^* = t_i$

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

- Straightforward to construct collision finder

Constructing Hash Functions

Domain Extension

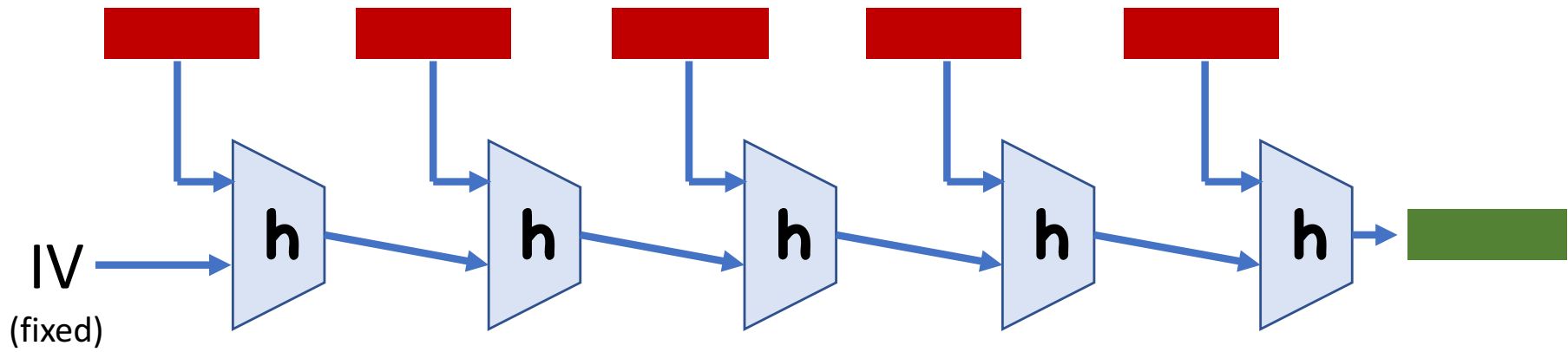
Goal: given h that compresses small inputs, construct H that compresses large inputs

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

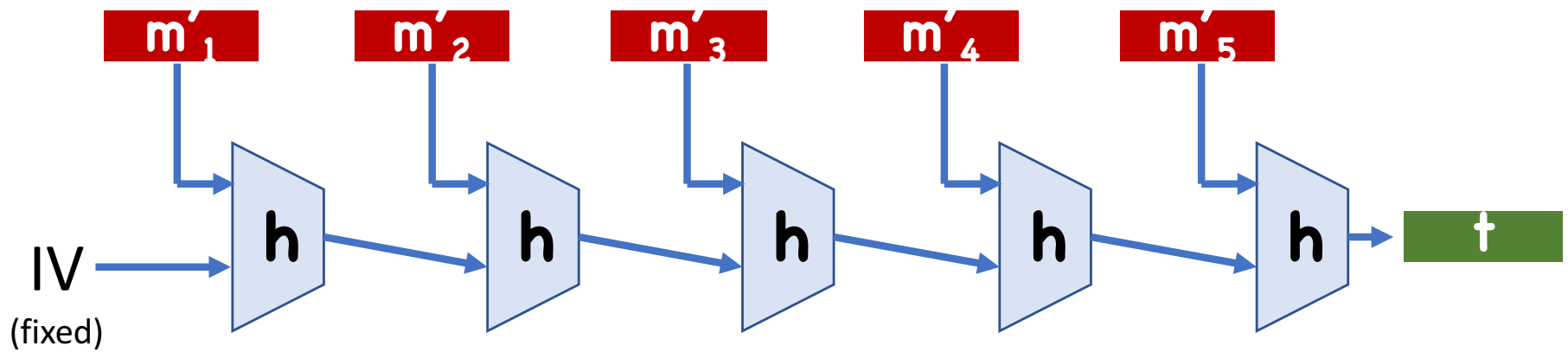
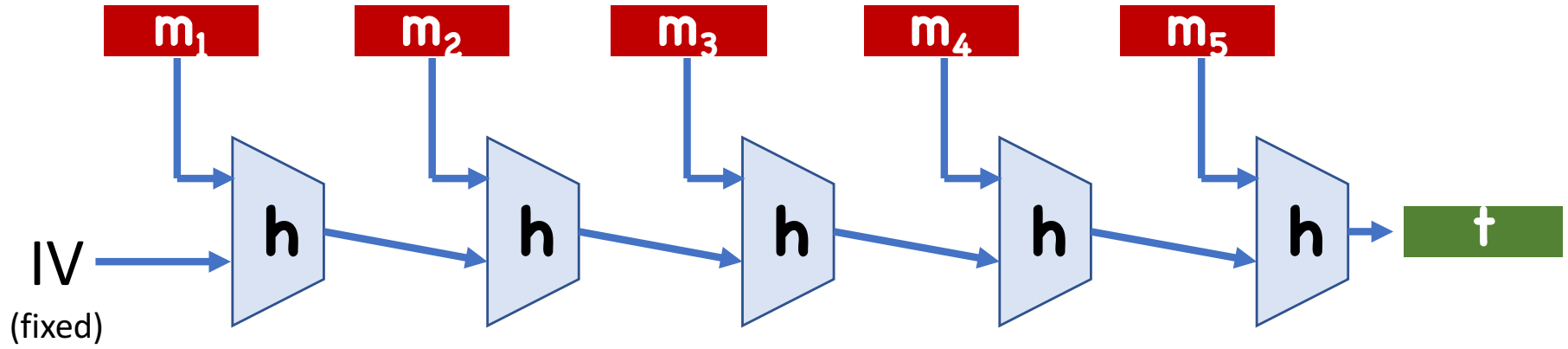
- Called compression functions
- Easier to design

Merkle-Damgard

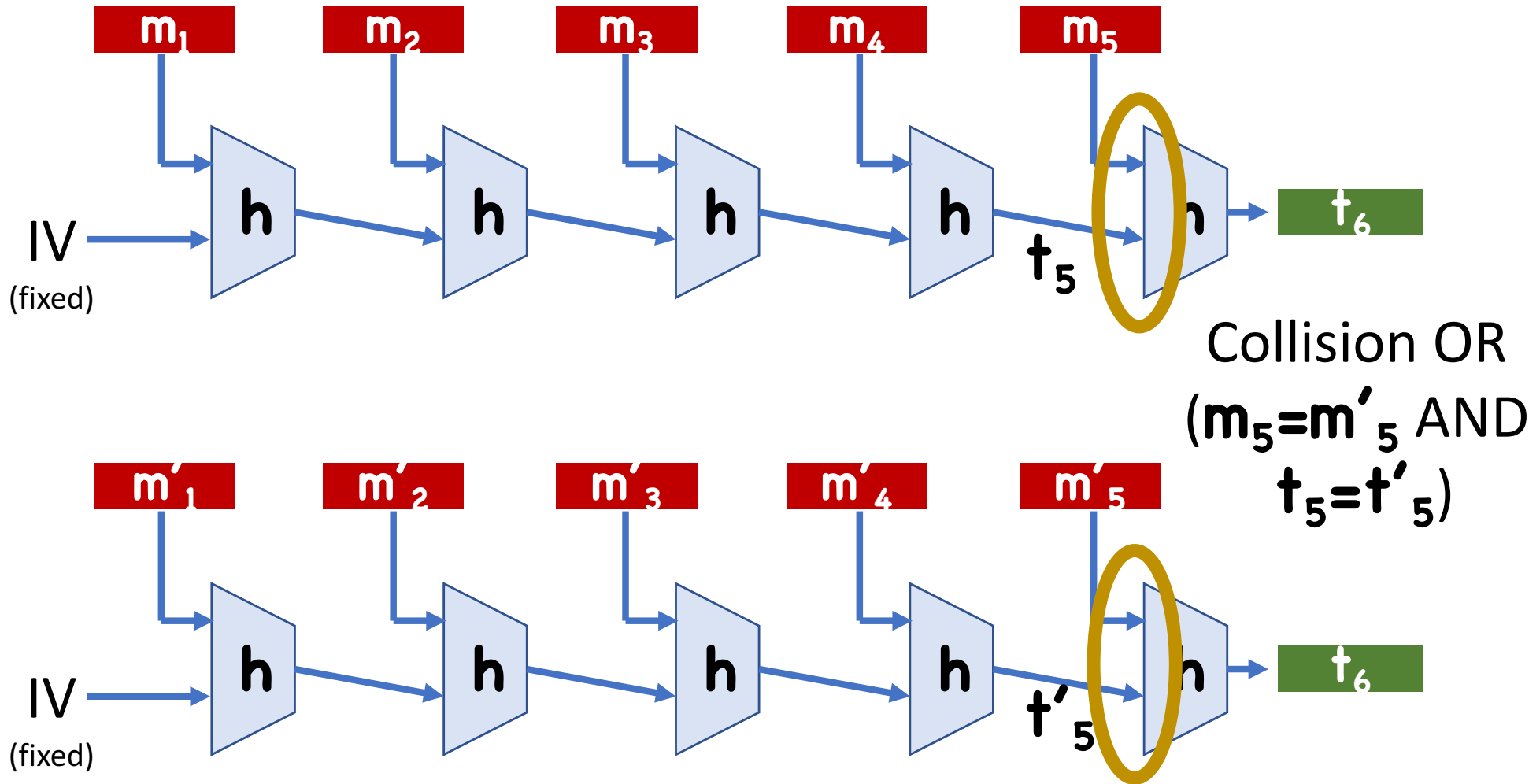


Theorem: If an adversary knows a collision for fixed-length Merkle-Damgard, it can also compute a collision for h

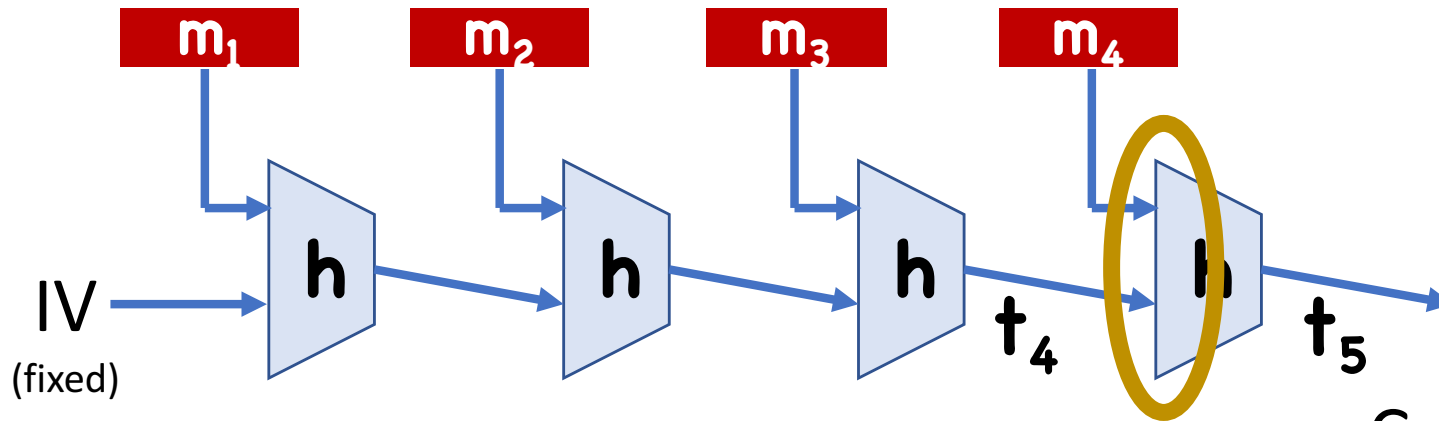
Proof



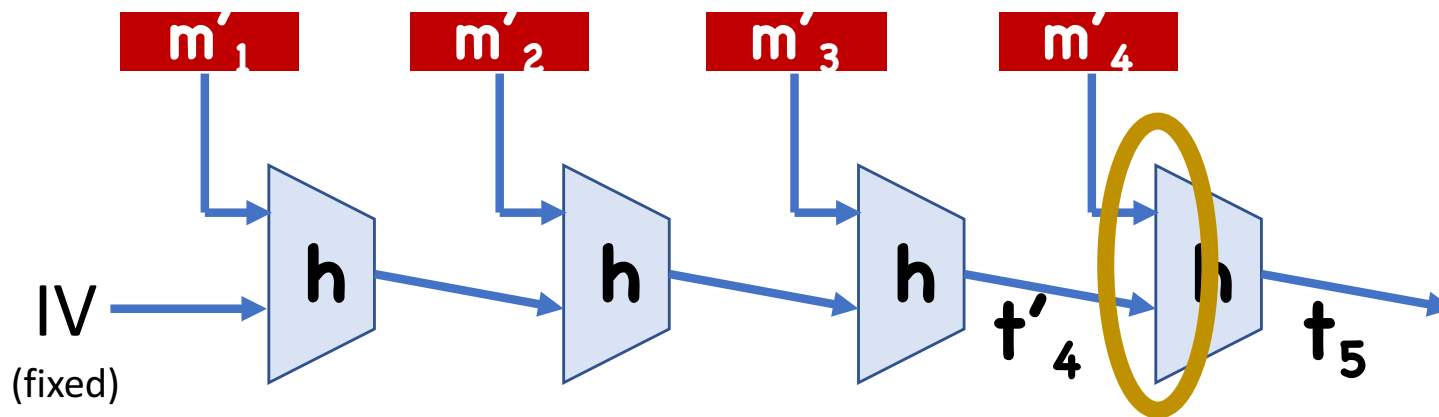
Proof



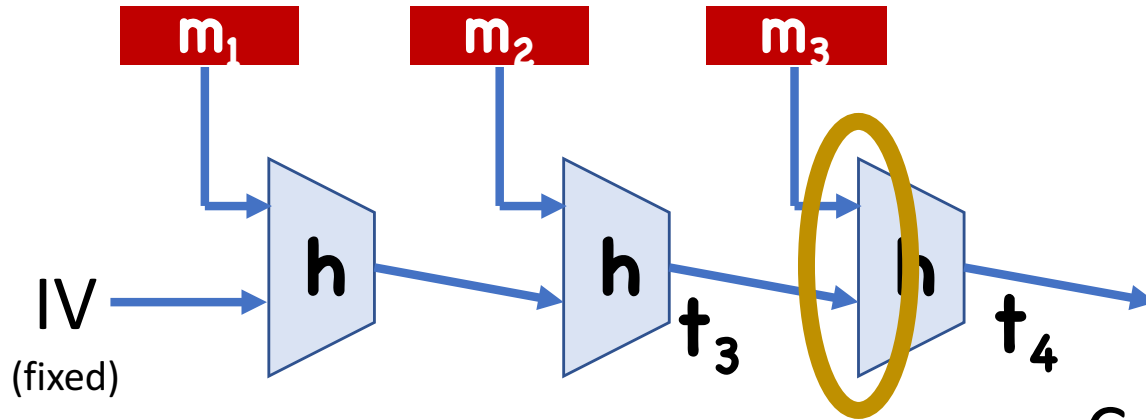
Proof



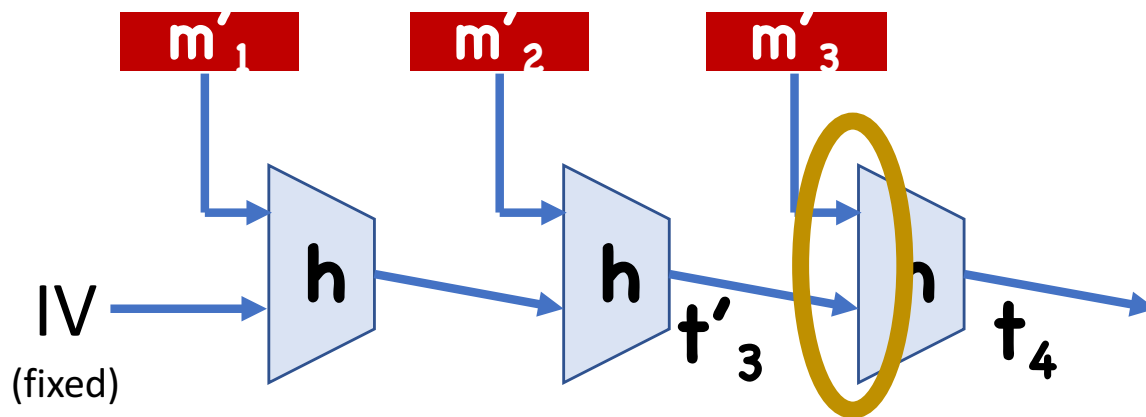
Collision OR
($m_4 = m'_4$ AND
 $t_4 = t'_4$)



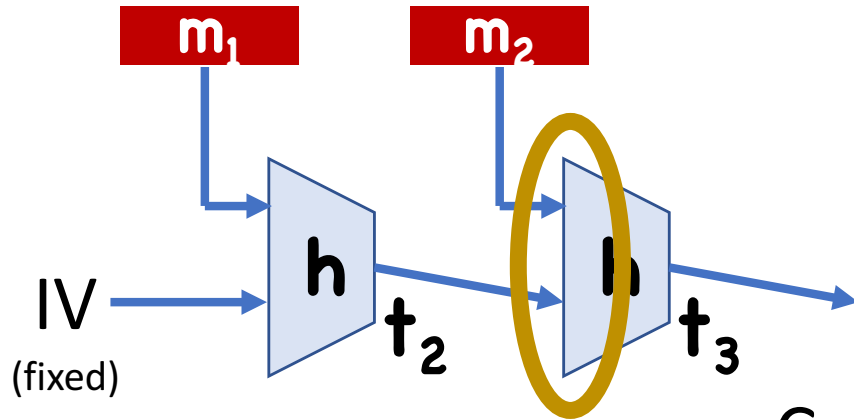
Proof



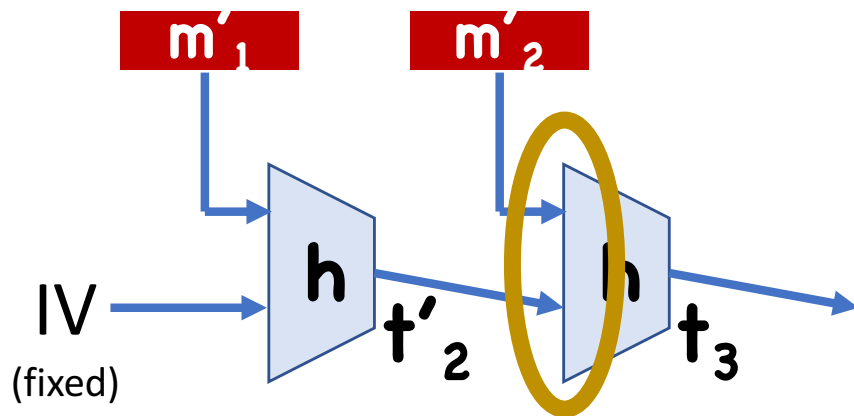
Collision OR
($m_3 = m'_3$ AND
 $t_3 = t'_3$)



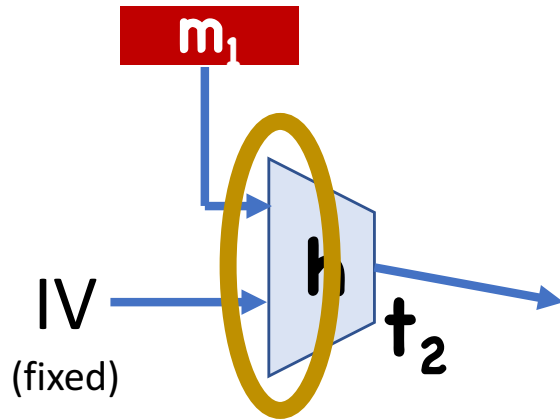
Proof



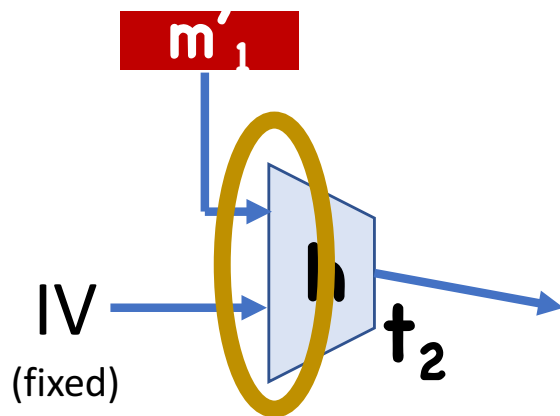
Collision OR
($m_2 = m'_2$ AND
 $t_2 = t'_2$)



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 $h(x||IV) = IV$?

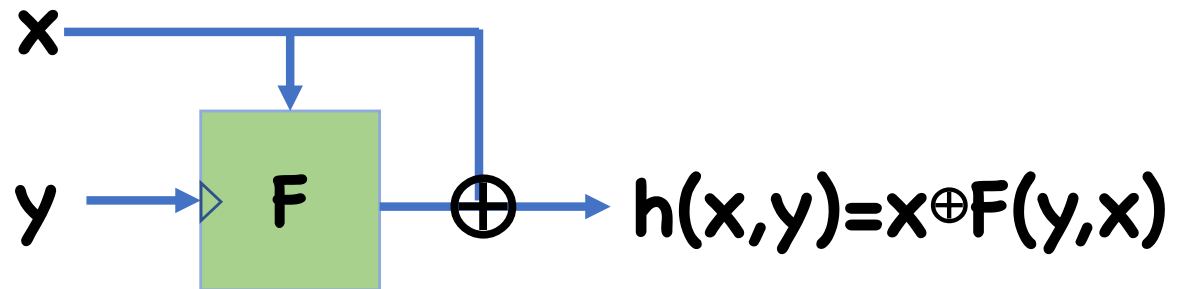
Need proper padding to enable security proof

- Ex: append message length to end of message

Constructing **h**

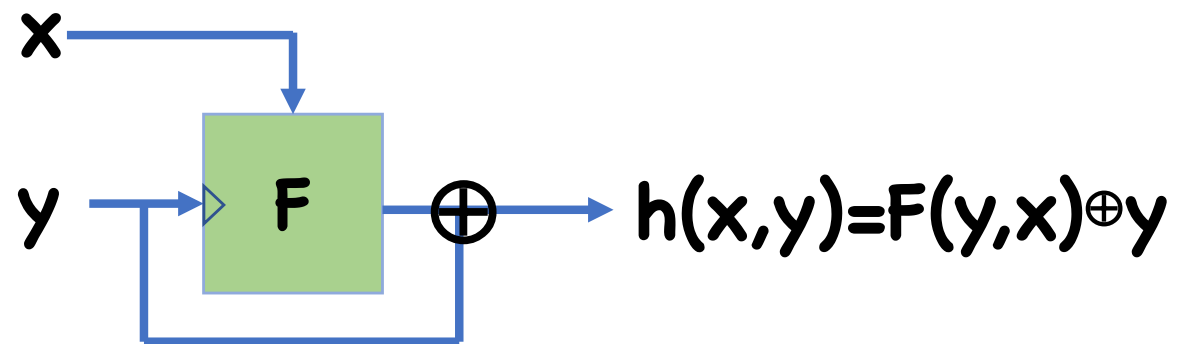
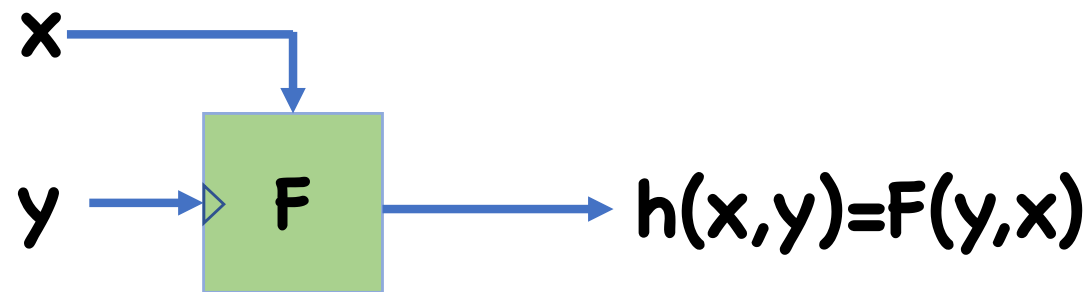
Common approach: use block cipher

Davies-Meyer

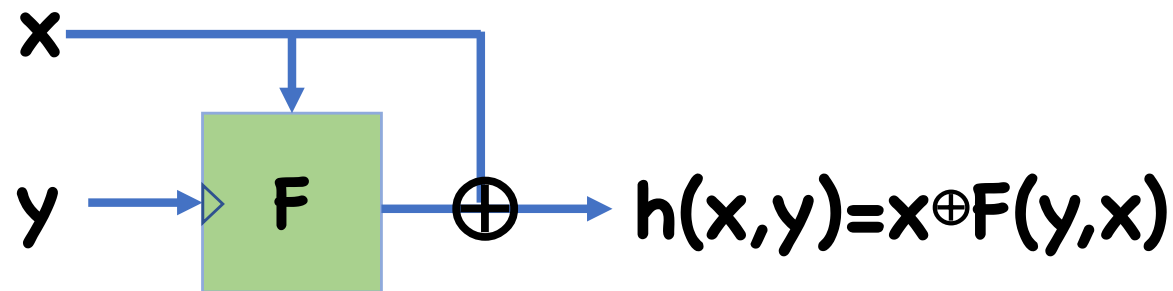


Constructing h

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 $\mathbf{i=1, \dots, T}$,
 - 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

$$\begin{aligned} &= \text{Number of pairs} \times \text{Prob each pair is collision} \\ &\approx \mathbf{(T \text{ choose } 2)} \times \mathbf{1/|R|} \end{aligned}$$

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

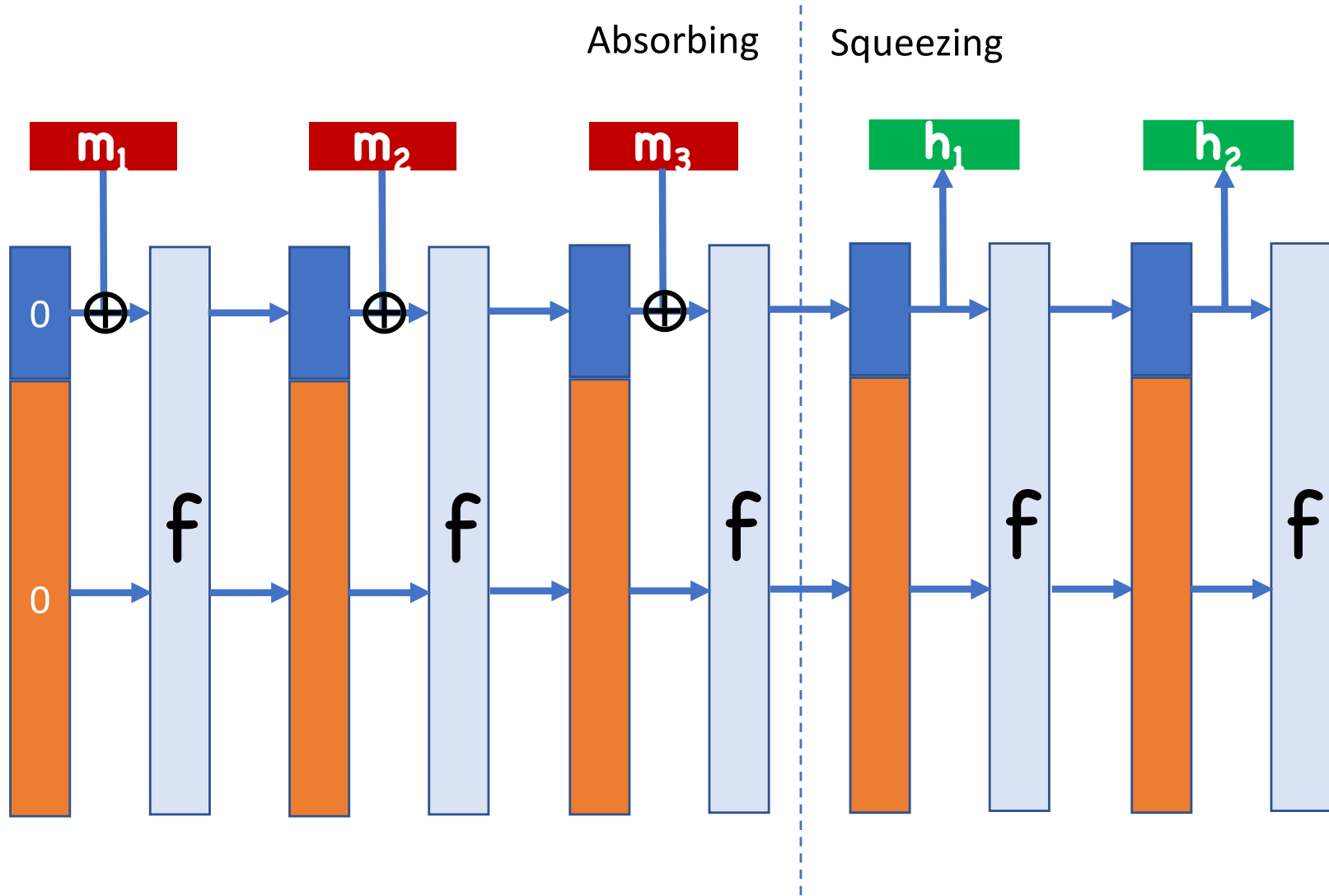
\Rightarrow likely to find a collision

Birthday Attack

Space?

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

Sponge Construction



Sponge Construction

Advantages:

- Round function \mathbf{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: $\mathbf{MAC(k,m) = H(k \parallel m)}$

Thought: if \mathbf{H} is a “good” hash function and \mathbf{k} is random, should be hard to predict $\mathbf{H(k \parallel m)}$ without knowing \mathbf{k}

Unfortunately, cannot prove secure based on just collision resistance of \mathbf{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 simply query H , but don't take advantage of any structure

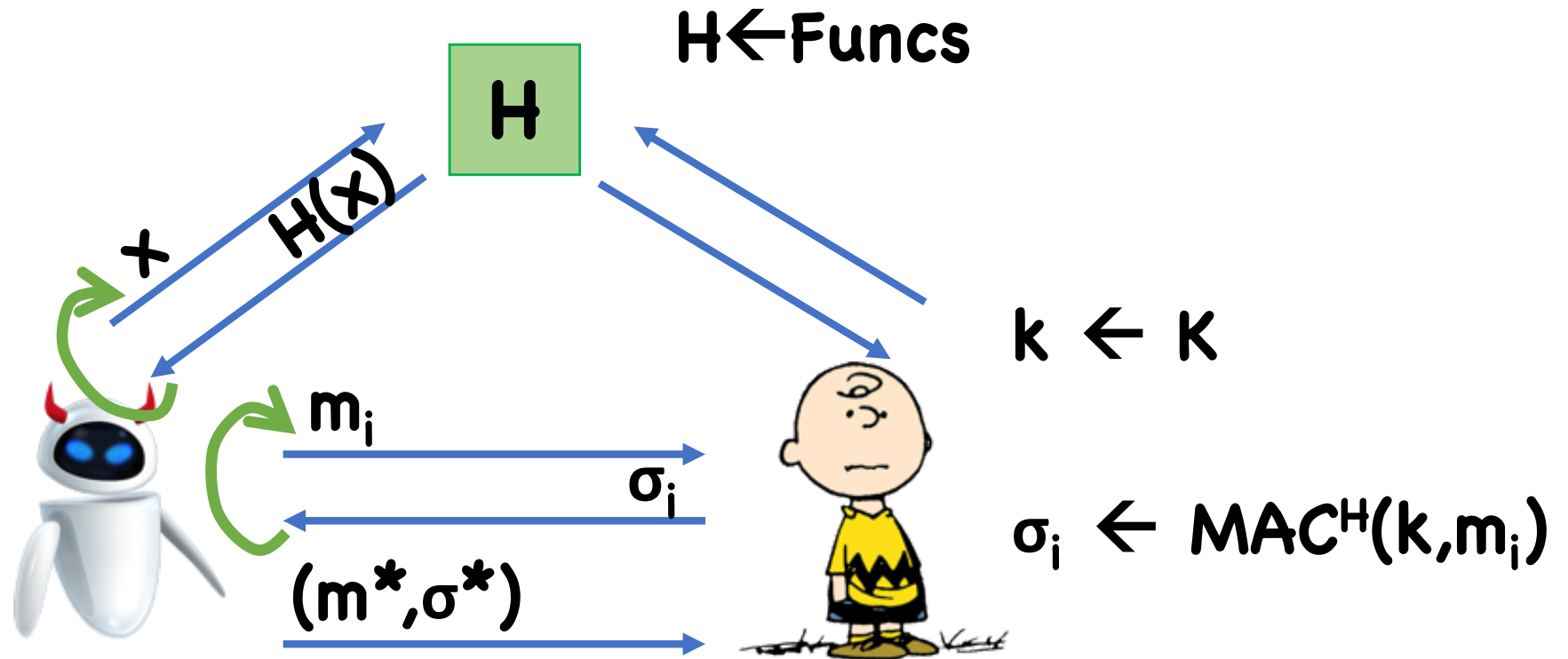
MAC in ROM

$$\text{MAC}^H(k,m) = H(k||m)$$

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

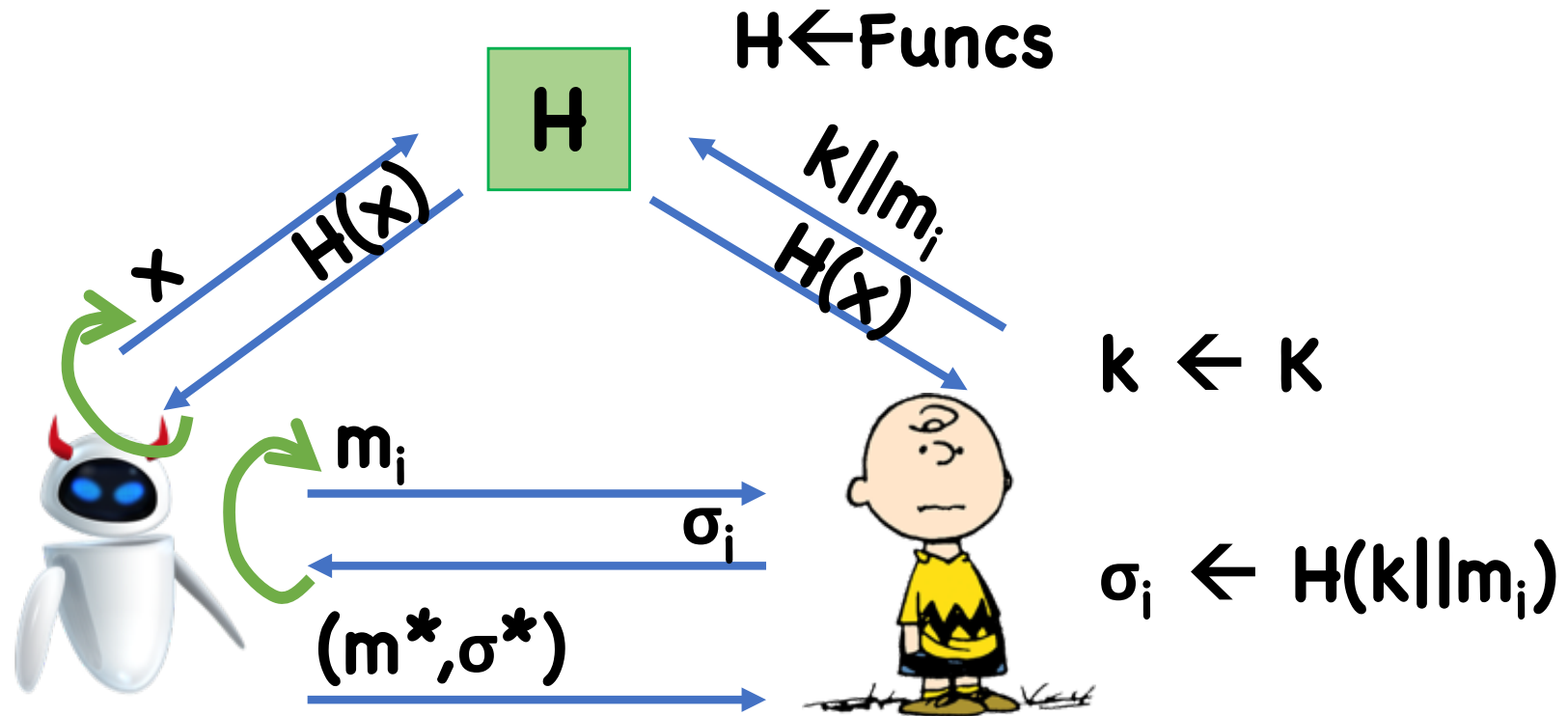
Theorem: $H(k || m)$ is a $(t, q, qt/2^n)$ -CMA-secure MAC in the random oracle model

Meaning



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

Meaning



- Output 1 iff:
- $m^* \notin \{m_1, \dots\}$
 - $H(k \| m^*) = \sigma^*$

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

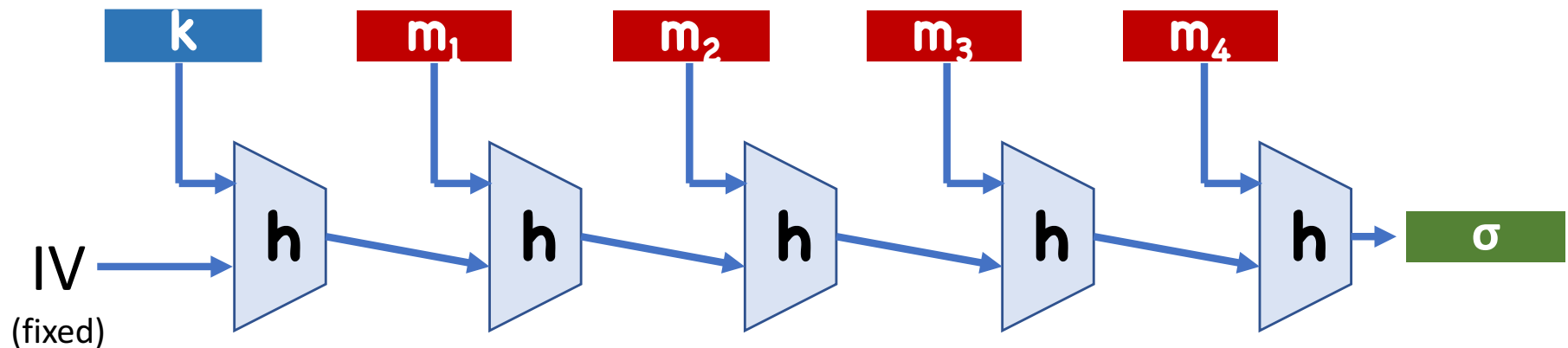
- If insecure, adversary must be taking advantage of structural weaknesses in H

When the ROM Fails

$$\text{MAC}^H(k,m) = H(k||m)$$

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

Instantiate with Merkle-Damgard (variable length)?



When the ROM Fails

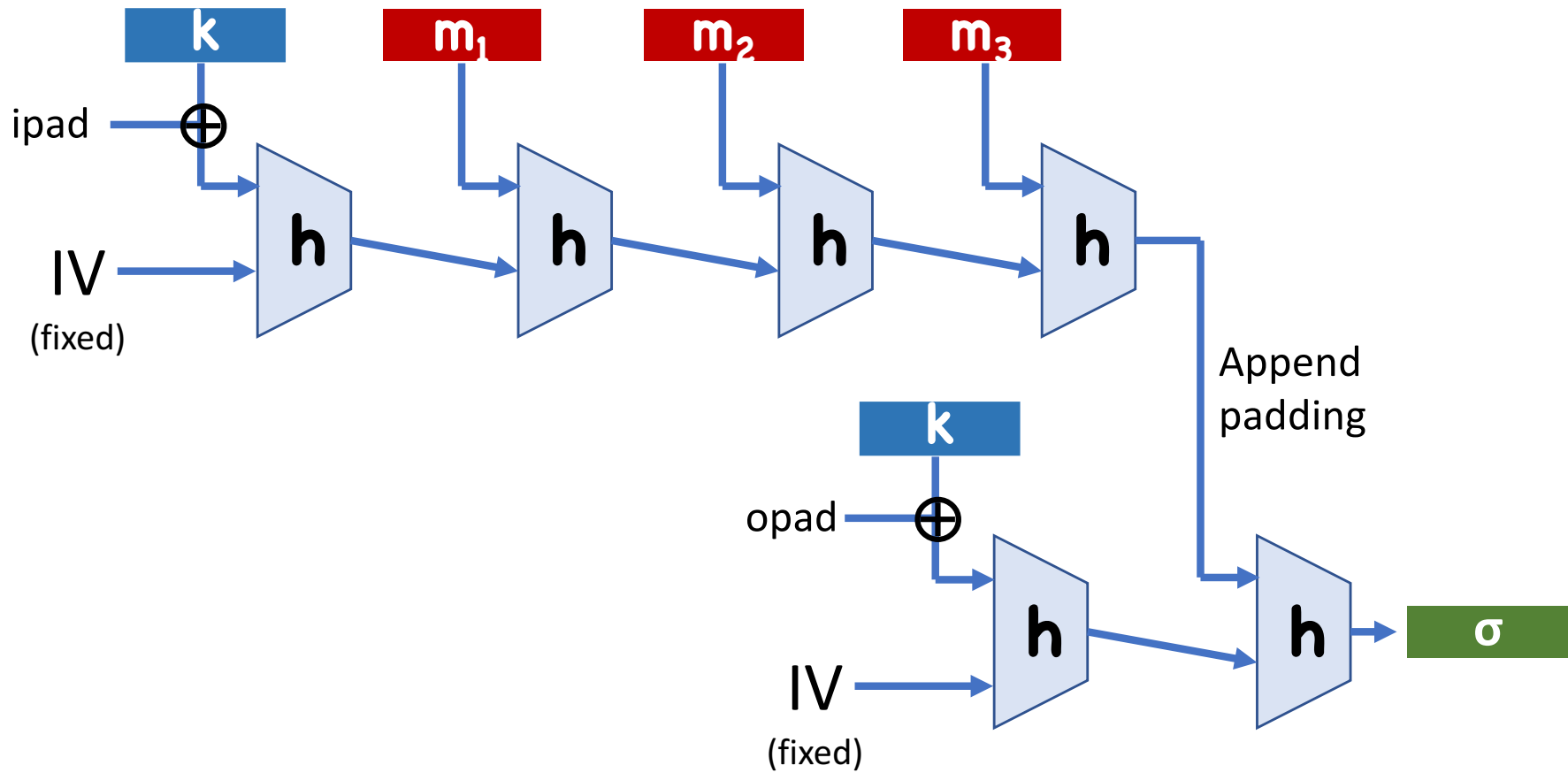
ROM does not apply to regular Merkle-Damgard

- Even if h is an ideal hash function

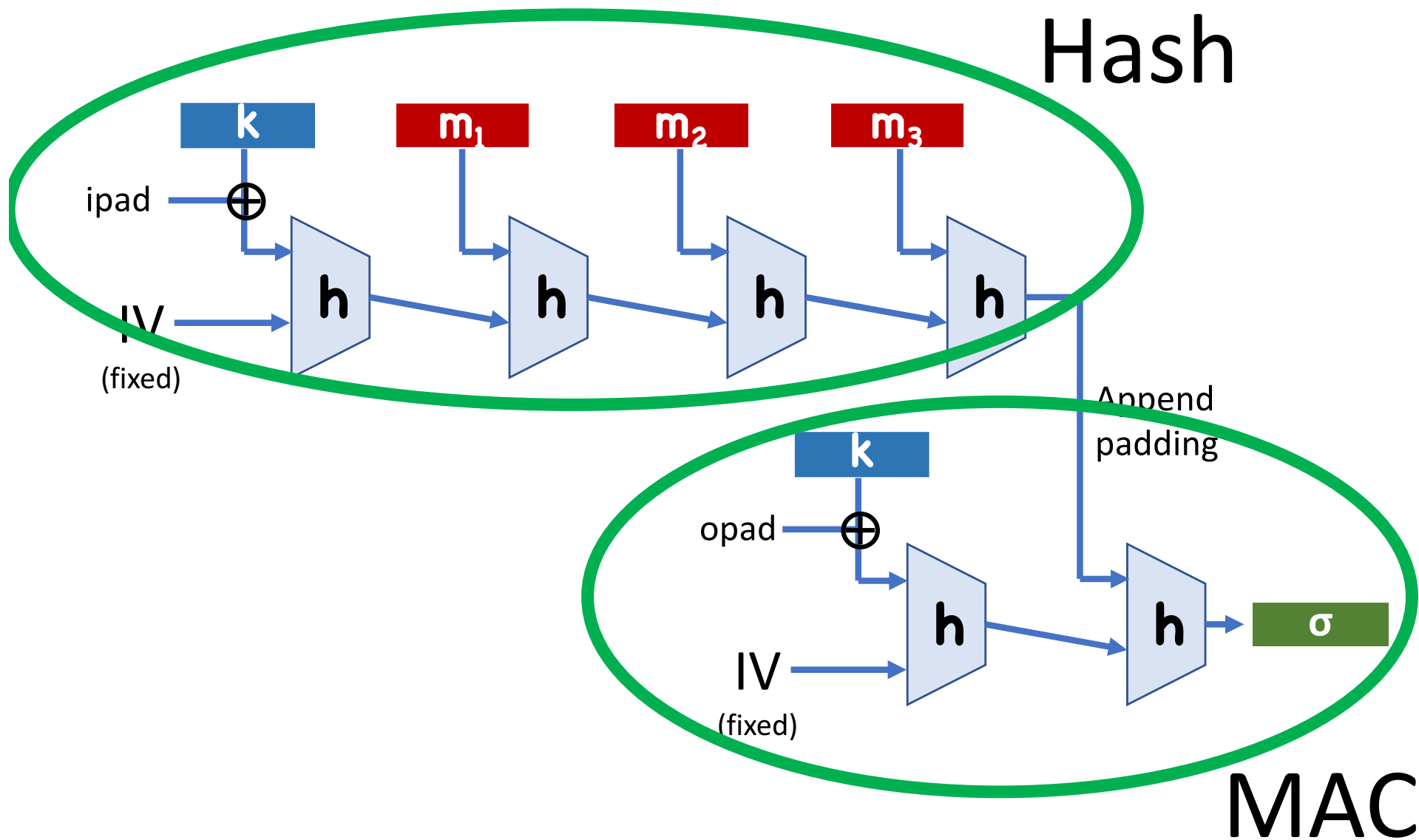
Takeaway: be careful about using ROM for non-“monolithic” hash functions

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

HMAC



HMAC



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