Security and Cryptography 1

Stefan Köpsell, Thorsten Strufe

Module 6: Integrity

Disclaimer: large parts from Mark Manulis, Dan Boneh, Stefan Katzenbeisser

Dresden, WS 17/18

TECHNISCHE UNIVERSITÄT DRESDEN

Reprise from the last modules

You have an overview of cryptography and cryptology

You know different adversary models and their corresponding games

You know what symmetric cryptography is

You recall the difference of stream and block ciphers

You can explain the OTP and constructions for stream ciphers

You can prove that the OTP has perfect secrecy

You can tell PRFs and PRP apart and you know constructions for block ciphers

You can explain different modes of operation and their properties

Module Outline

Verification of message integrity as a goal

Adversary and security models

Hashes and cryptographic hash functions

Collisions and how to create them (also: the birthday paradox)

The Merkle-Damgard construction and some real hash functions (MD5, SHA-1)

Block ciphers as compression functions

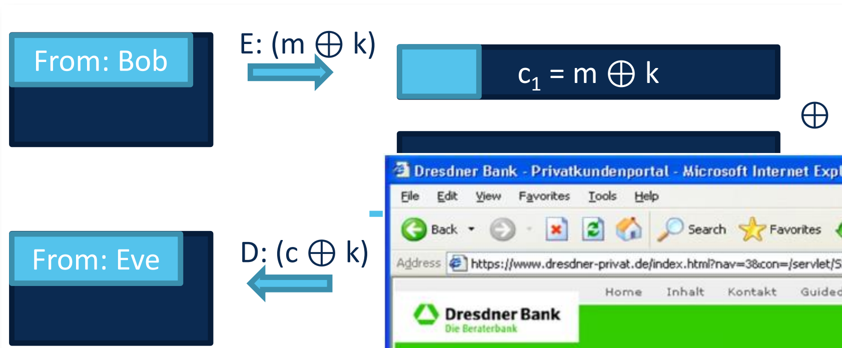
Secure MACs from hash functions and PRFs

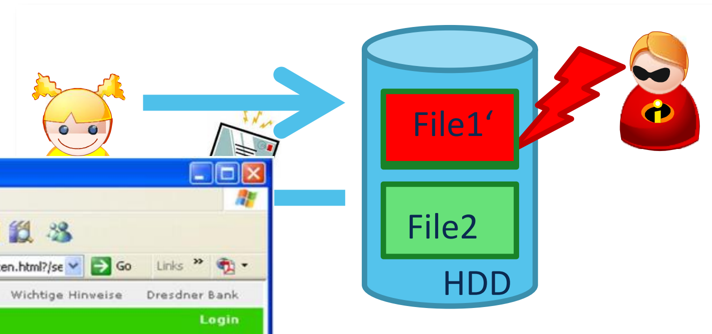
MACs using block ciphers (CBC, NMAC, HMAC)

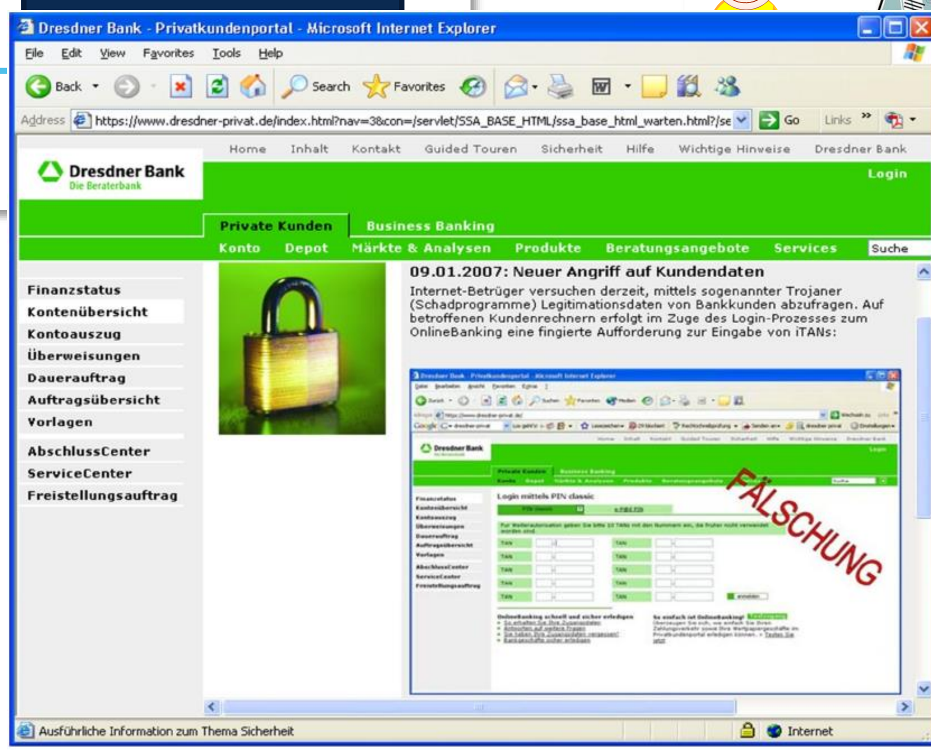
Integrity and Authenticity

So far messages can be kept confidential

Integrity of messages not given







Message Integrity

Algorithms:

Tag S: $M \rightarrow T$ $M = \{0,1\}^n$; $S = \{0,1\}^t$ with $n \gg t$

Verify V: $M \times T \rightarrow \{yes, no\}$

Bad Examples

Cross sum:

f(x) : calculate the cross sum of all bytes in the message

Is this secure? Why (not)?

f(5 23) = f(23 5)

CRC:

tag ← CRC(m) ; Verify tag: return CRC(m) == tag

Is this secure? Why (not)?

Integrity requires a secret

Adversary can create new message and recompute CRC

Simple Encryption:

tag ← Enc(k,m) $|_{1,\ldots,6}$ Verify tag: return tag == Enc(k,m) $|_{1,\ldots,6}$

Is this secure? Why (not)?

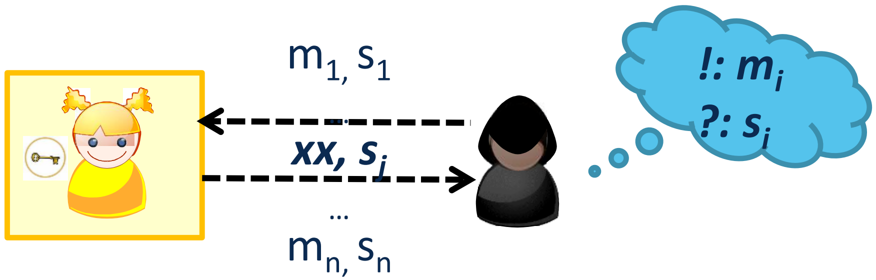
Security depends on |S|

Adversary can guess tag for a message in 2^6

Existential Forgery

Chosen Message Attack:

- given s_1, s_2, … s_n for chosen m_i



(variations are: known key / known signature attacks)

Existential Forgery:

Produce ***some*** new valid tuple (m,s) (any message, even gibberish)

⇒ adversary cannot produce a valid tag for a new message

⇒ adversary cannot even produce (m,t') for (m,t) and t' ≠ t

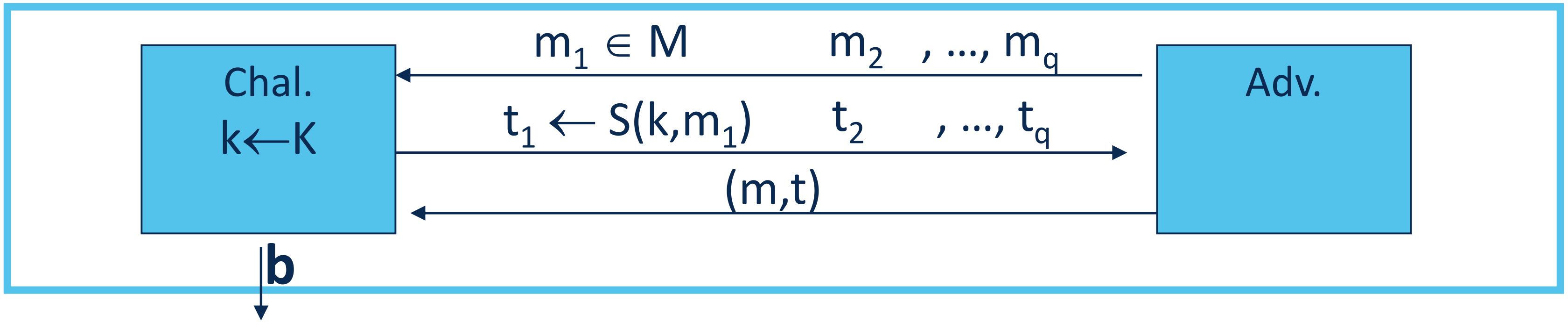
Breaches in general:

Exist. forgery < selective forgery < universal forgery < total break

TECHNISCHE UNIVERSITÄT DRESDEN

Defining the MAC game

For a MAC I=(S,V) and adversary A, where (S,V) additionally take *k:*



b=1 if V(k,m,t) = `yes' and (m,t) $\notin$ { (m_1,t_1) , … , (m_q,t_q) }

b=0 otherwise

Def: I=(S,V) is a **secure MAC** if for all "efficient" A:

$$\text{Adv}_{\text{MAC}}[A,I] \;=\; \Pr[\text{Chal. outputs } 1] \le \boldsymbol{\varepsilon}$$

So how can we build a secure MAC?

Variations:

Existential forgery may forge tag on a message that seems gibberish (like, for instance, a random-looking bitstring like a ciphertext or a key…)

Secure MAC from a PRF

Assume a PRF $F: K \times X \rightarrow Y$

Define integrity scheme $I_F = (S,V)$:

- $S(k,m) := F(k,m)$
- $V(k,m,t) :=$ test $t==F(k,m)$

- *Is this a secure integrity scheme?*

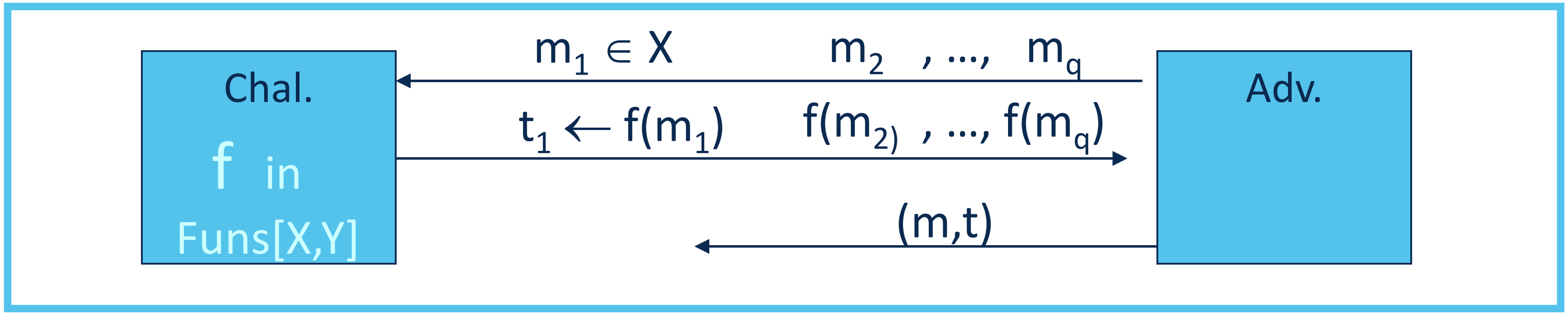
suppose F is PRF, m given, what is the chance of the adv. to guess s?

$$Adv_{MAC}[A,I_F] =$$

- *What are its downsides?*

Security of the simple PRF MAC

Assume **f: X⟶Y** is a random function and $1/|Y|$ is negligible



Assume **F: K x X⟶Y** is a secure PRF, construct MAC I_F:

For every efficient MAC adversary A attacking I_F, there must be an efficient PRF adversary B attacking F, s.t.:

$$Adv_{MAC}[A, I_F] \leq Adv_{PRF}[B, F] + 1/|Y|$$

What happens if we truncate the output?

$\Rightarrow I_F$ is secure as long as:

- F is a secure PRF (given), and $|Y|$ is large ($|Y| \geq 2^{80}$)

Interlude: Collision Resistant Hash Functions

Goal:

Map a message of arbitrary length to a characteristic digest (fingerprint)

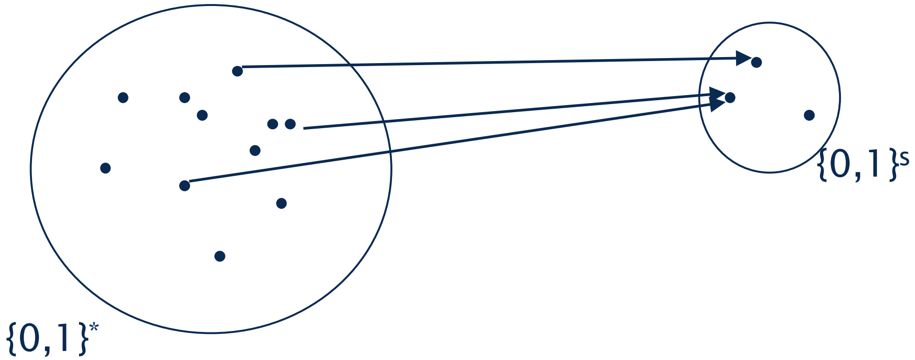
Hash $H: M \rightarrow S$ with $M = \{0,1\}^*$ and $S = \{0,1\}^s$

- has an efficient algorithm to evaluate H(x)
- is an „onto“ function (surjective, Im(H) = S)
- avoids collision (→ maps uniformly to S)
- creates chaos (slight changes in m yield large differences in s)

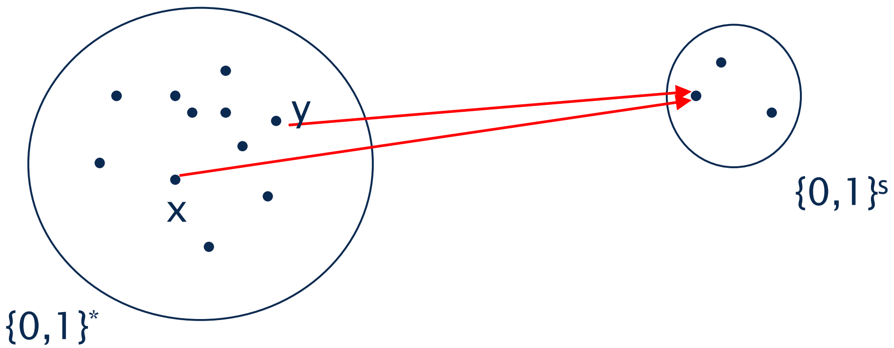
Cryptographic Hash Functions

Further properties / requirements of hash functions for security:

- Compression is irreversible



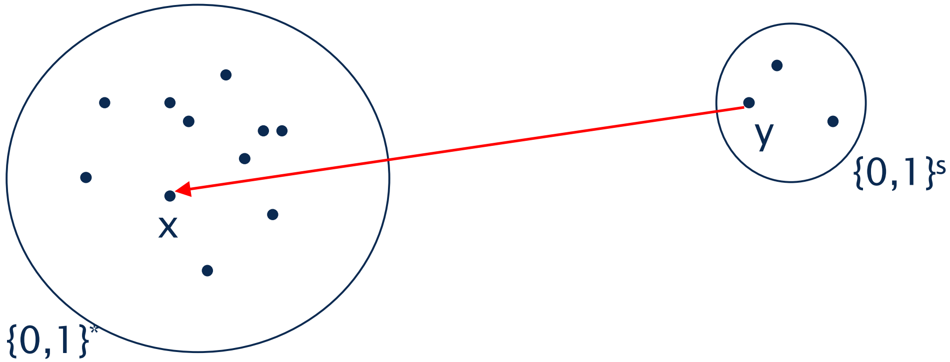
- Collision resistance



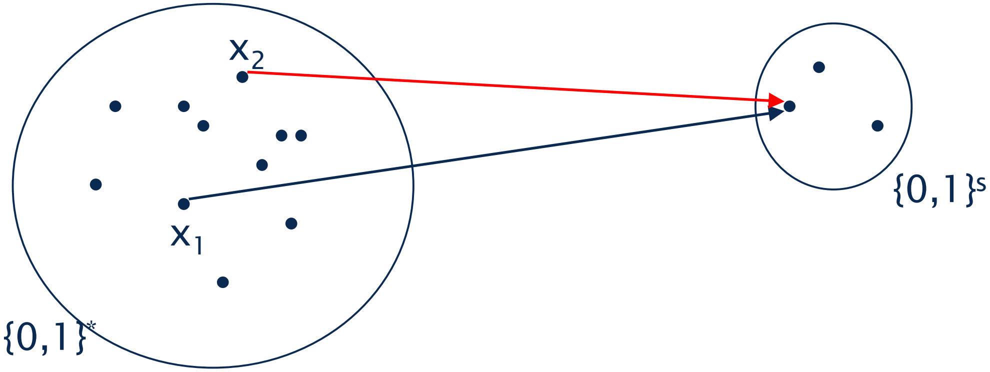
Cryptographic Hash Functions

Further requirements to hash functions for security:

- Pre-image resistance



- 2nd pre-image resistance



Collision Resistance

Let $H: M \to S$ be a hash function ($|M| >> |S|$)

A **collision** for H is a pair $m_0, m_1 \in M$ such that:

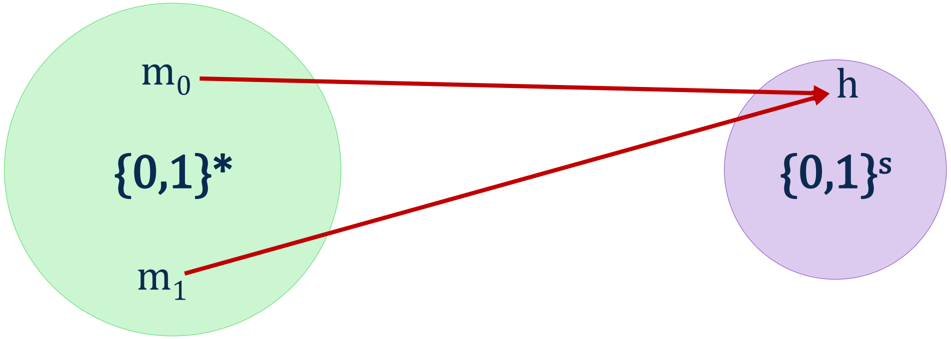
$$H(m_0) = H(m_1) \quad \text{and} \quad m_0 \neq m_1$$

A function H is **collision resistant** if for all (explicit) "eff" algs. A:

$$\mathbf{Adv_{CR}[A,H] = Pr[\text{ A outputs collision for H}] \leq \varepsilon}$$

Brute Forcing Collisions

Do collisions exist?



Yes.

but it should be hard to find collisions (in polynomial time)

Trivial Collision-Finder (Brute Force)

s must be sufficiently large

compute $H \stackrel{def}{=} \{H(m) \mid \text{for all } m \in \{0,1\}^s\}$

if no collision found compute $H(m^*)$ for any $m^* \notin \{0,1\}^s$

time needed $O(2^s)$

there must be at least one m with $H(m) \in H$ such that $H(m) = H(m^*)$

Generic Attack on C.R. Hash Function

Let $H: M \rightarrow \{0,1\}^s$ be a hash function ($|M| >> 2^s$)

Generic alg. to find a collision **in time $O(2^{s/2})$** hashes

Algorithm:

1. Choose **$2^{s/2}$** random messages in M: $m_1, \ldots, m_{(2^{s/2})}$ (distinct w.h.p)
2. For i = 1, …, $2^{s/2}$ compute $t_i = H(m_i) \in \{0,1\}^s$
3. Look for a collision ($t_i = t_j$). If not found, got back to step 1.

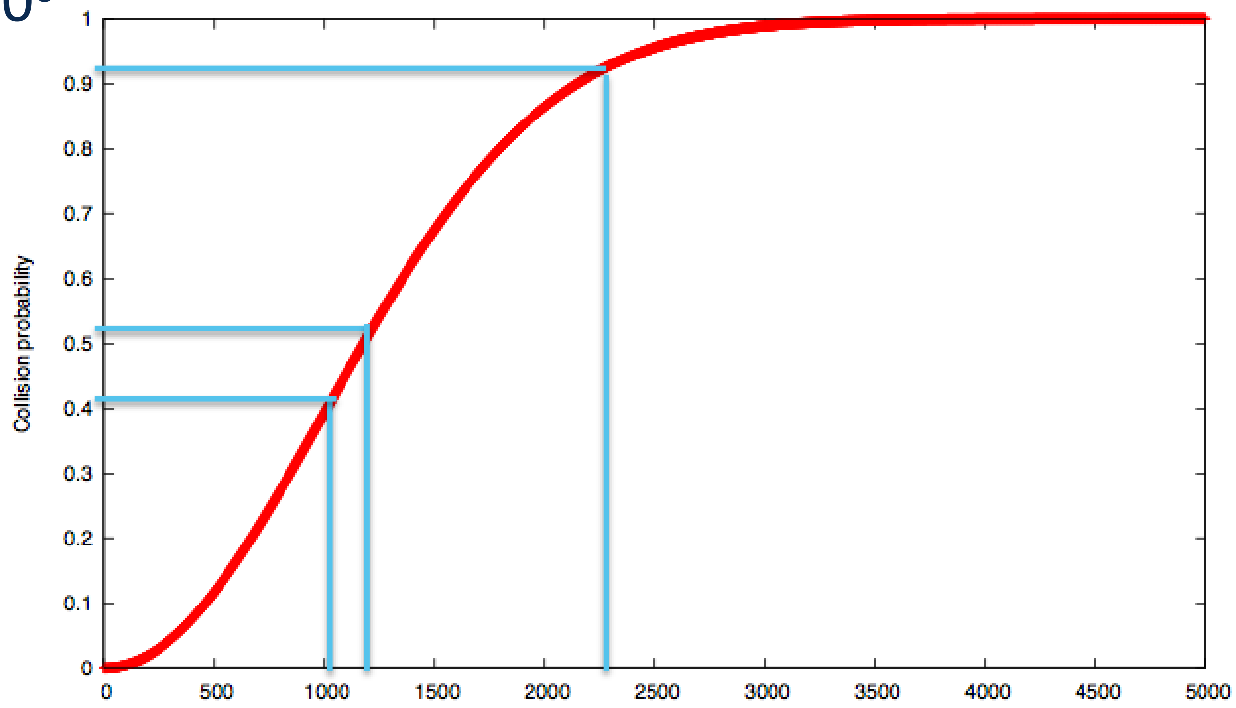
How well will this work?

The Birthday Paradox

Let $r_1, \ldots, r_n \in \{1,\ldots,B\}$ be random integers, chosen iid.

BP states: when **n**= 1.2 × **B^1/2** then $\Pr[\,\exists i \neq j:\ r_i = r_j\,] \geq ½$

B=10^6



samples n

TECHNISCHE UNIVERSITÄT DRESDEN

Generic attack due to the B. P.

$H: M \rightarrow \{0,1\}^s$. Collision finding algorithm:

1. Choose $\mathbf{2^{s/2}}$ random elements in M: $m_1, \ldots, m_{2^{s/2}}$
2. For i = 1, …, $2^{s/2}$ compute $t_i = H(m_i)$ $\in \{0,1\}^s$
3. Look for a collision ($t_i = t_j$). If not found, got back to step 1.

Expected number of iterations until success ≈

Running time: $\mathbf{O(2^{n/2})}$ (space $O(2^{n/2})$)

The Merkle-Damgard construction

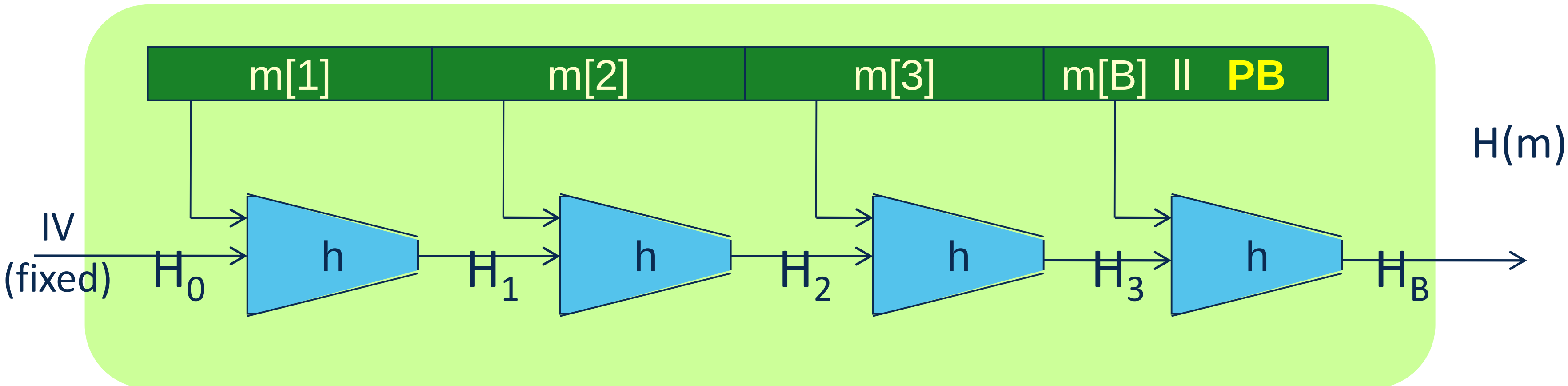
From *short message blocks* to *arbitrarily long messages*…

Given a compression function $h : \{0,1\}^{2s} \longrightarrow \{0,1\}^{s}$ and

Input $m \in \{0,1\}^{*}$ of length L and PB: = 1000…0 ll L

(L: 64 bits)

Construct H of B= $\lceil L/s \rceil$ iterations of h:



If h is a fixed length CRHF, then H is an arbitrary length CRHF

Proof: either M=M‘ (no collision) or H_{B-i}(m[B-i])=H_{B-i}(m‘[B-i]) (collision on h)