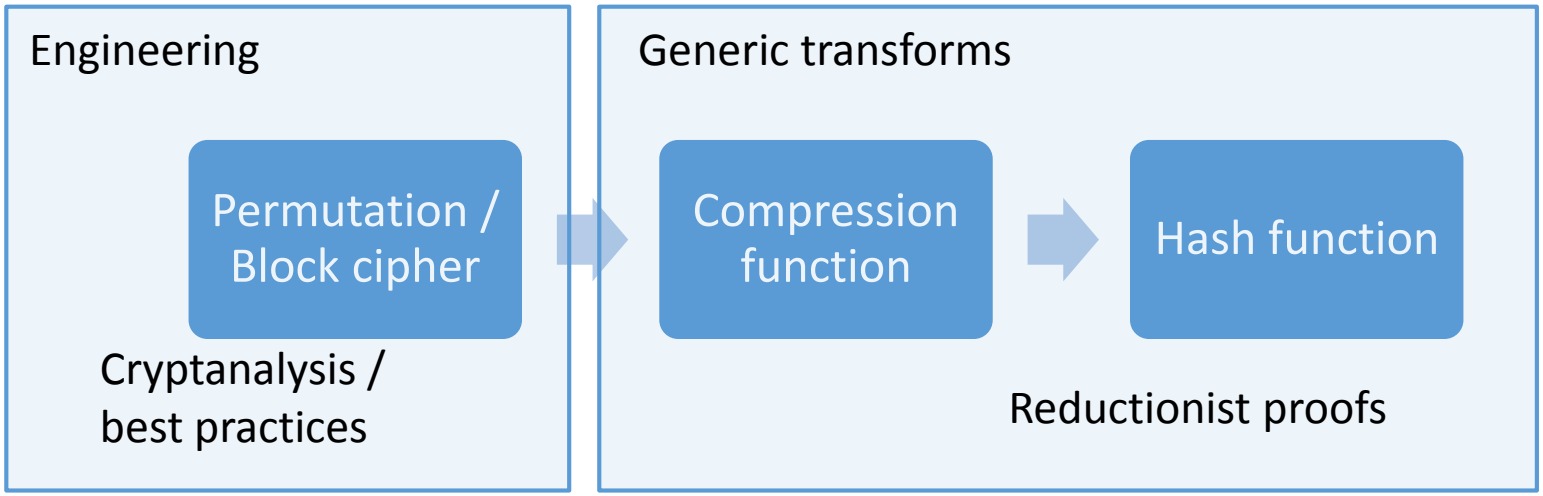
Cryptographic Hash Functions Part II

2MMC10 Cryptology

Andreas Hülsing, TU/e

Hash function design

- Create fixed input size building block
- Use building block to build compression function
- Use “mode“ for length extension



(Length-Extension) Modes

Merkle-Damgård construction

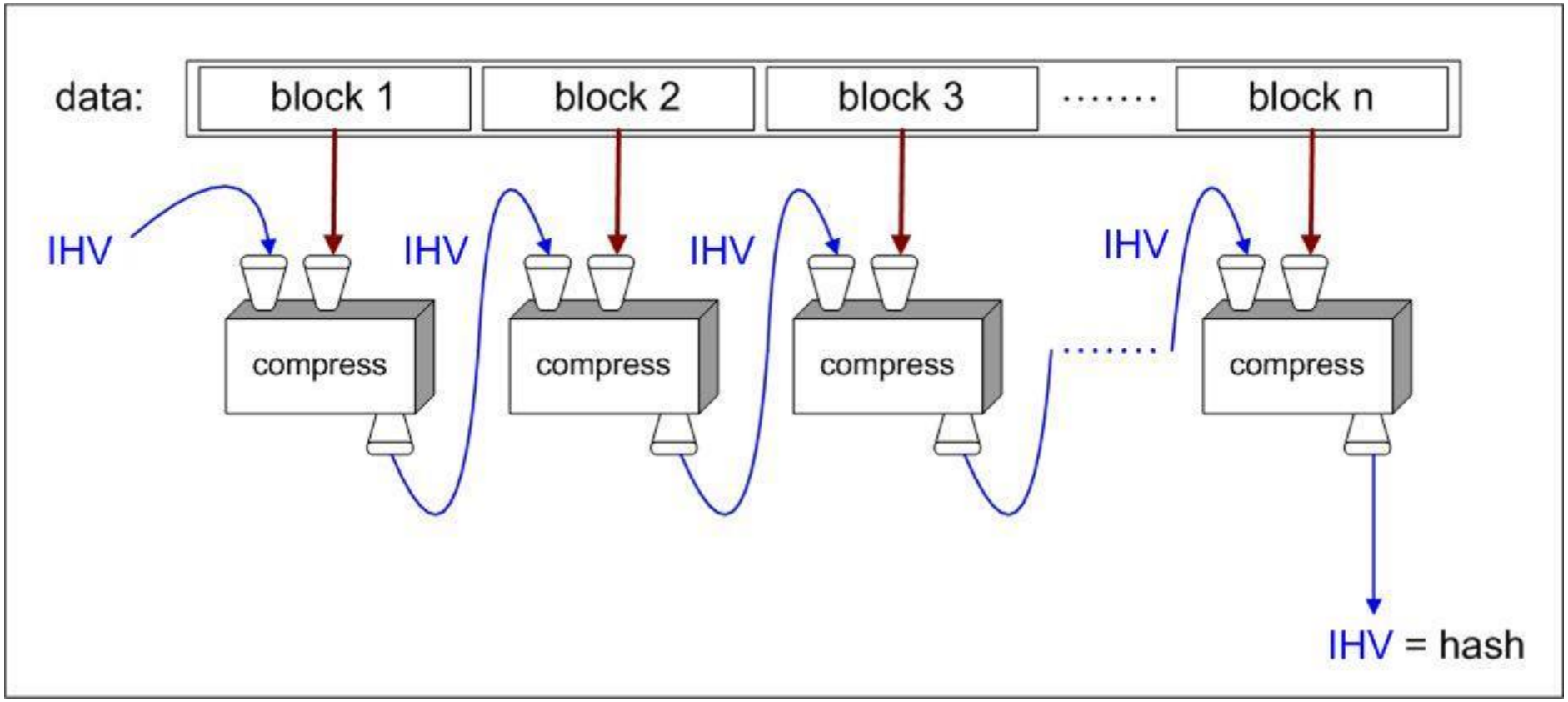
Given:

- *compression function*: $CF : \{0,1\}^n \times \{0,1\}^r \rightarrow \{0,1\}^n$

Goal:

- *Hash function:* h: $\{0,1\}^* \rightarrow \{0,1\}^n$

Merkle-Damgård - iterated compression



Merkle-Damgård construction

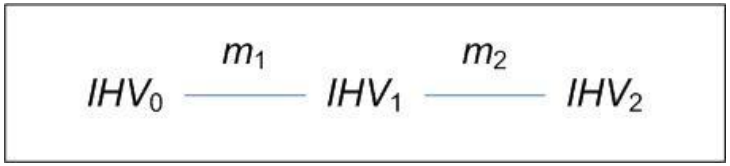
- Assume that message *m* can be split up into blocks m_1, …, m_s of equal block length *r*
  - most popular block length is *r* = 512
- *compression function*: $CF : \{0,1\}^n \times \{0,1\}^r \rightarrow \{0,1\}^n$
- *intermediate hash values* (length *n*) as *CF* input *and* output
- *message blocks* as second input of *CF*
- start with fixed initial IHV_0 (a.k.a. *IV* = *initialization vector*)
- iterate *CF* : $IHV_1 = CF(IHV_0, m_1)$, $IHV_2 = CF(IHV_1, m_2)$, …, $IHV_s = CF(IHV_{s-1}, m_s)$,
- take $h(m) = IHV_s$ as hash value
- advantages:
  - this design makes *streaming* possible
  - hash function analysis becomes compression function analysis
  - analysis easier because domain of *CF* is finite

padding

- *padding*: add dummy bits to satisfy block length requirement
- non-ambiguous padding: add one 1-bit and as many 0-bits as necessary to fill the final block
  - when original message length is a multiple of the block length, apply padding anyway, adding an extra dummy block
  - any other non-ambiguous padding will work as well

Merkle-Damgård strengthening

- let padding leave final 64 bits open
- encode in those 64 bits the original message length
  - that's why messages of length ≥ 2^{64} are not supported
- reasons:
  - needed in the proof of the Merkle-Damgård theorem
  - prevents some attacks such as
    - trivial collisions for random *IV*

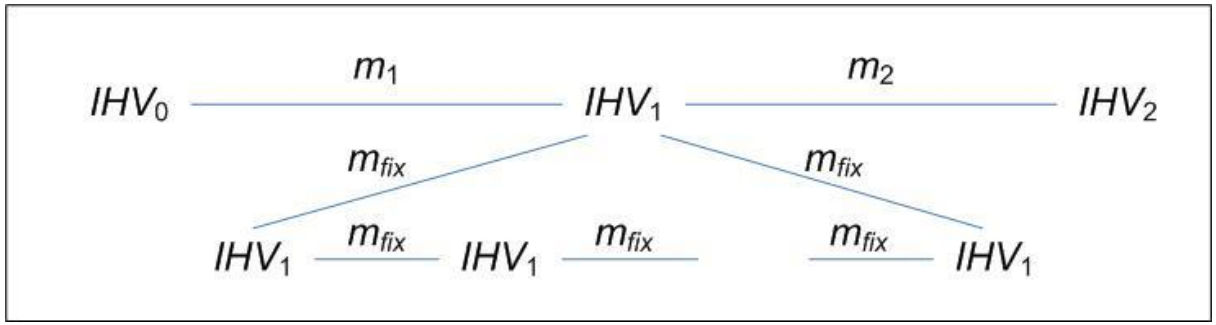


      - now $h(IHV_0, m_1 \,||\, m_2) = h(IHV_1, m_2)$
    - see next slide for more

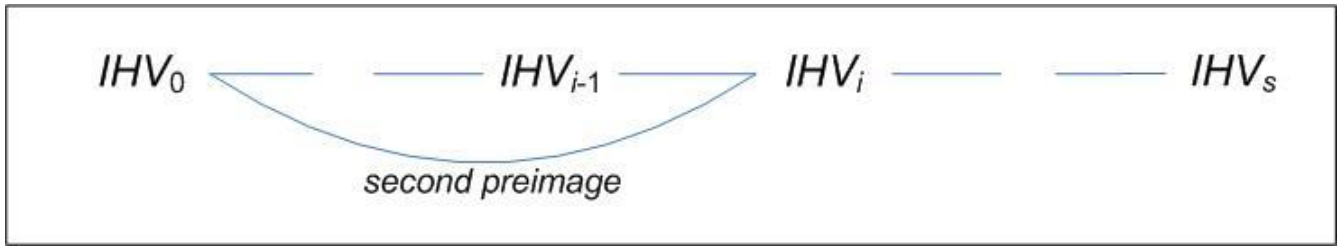
Merkle-Damgård strengthening, cont'd

- fixpoint attack

  fixpoint: *IHV*, *m* such that *CF*(*IHV*,*m*) = *IHV*



- long message attack



compression function collisions

- *collision* for a compression function: m_1, m_2, *IHV* such that $CF(IHV,m_1) = CF(IHV,m_2)$
- *pseudo-collision* for a compression function: m_1, m_2, IHV_1, IHV_2 such that $CF(IHV_1,m_1) = CF(IHV_2,m_2)$
- Theorem (Merkle-Damgård): If the compression function *CF* is pseudo-collision resistant, then a hash function *h* derived by Merkle-Damgård iterated compression is collision resistant.
  - Proof: Suppose $\boldsymbol{h(m_1) = h(m_2)}$, then
    - If $\boldsymbol{m_1, m_2}$ same size: locate the iteration where pseudo-collision occurs
    - Else a pseudo-collision for CF appears in the last blocks (cont. length)
- Note:
  - a method to find pseudo-collisions does not lead to a method to find collisions for the hash function
  - a method to find collisions for the compression function is almost a method to find collisions for the hash function, we 'only' have a wrong *IHV*

Sponges

Given:

- *permutation*: $f : \{0,1\}^b \rightarrow \{0,1\}^b$

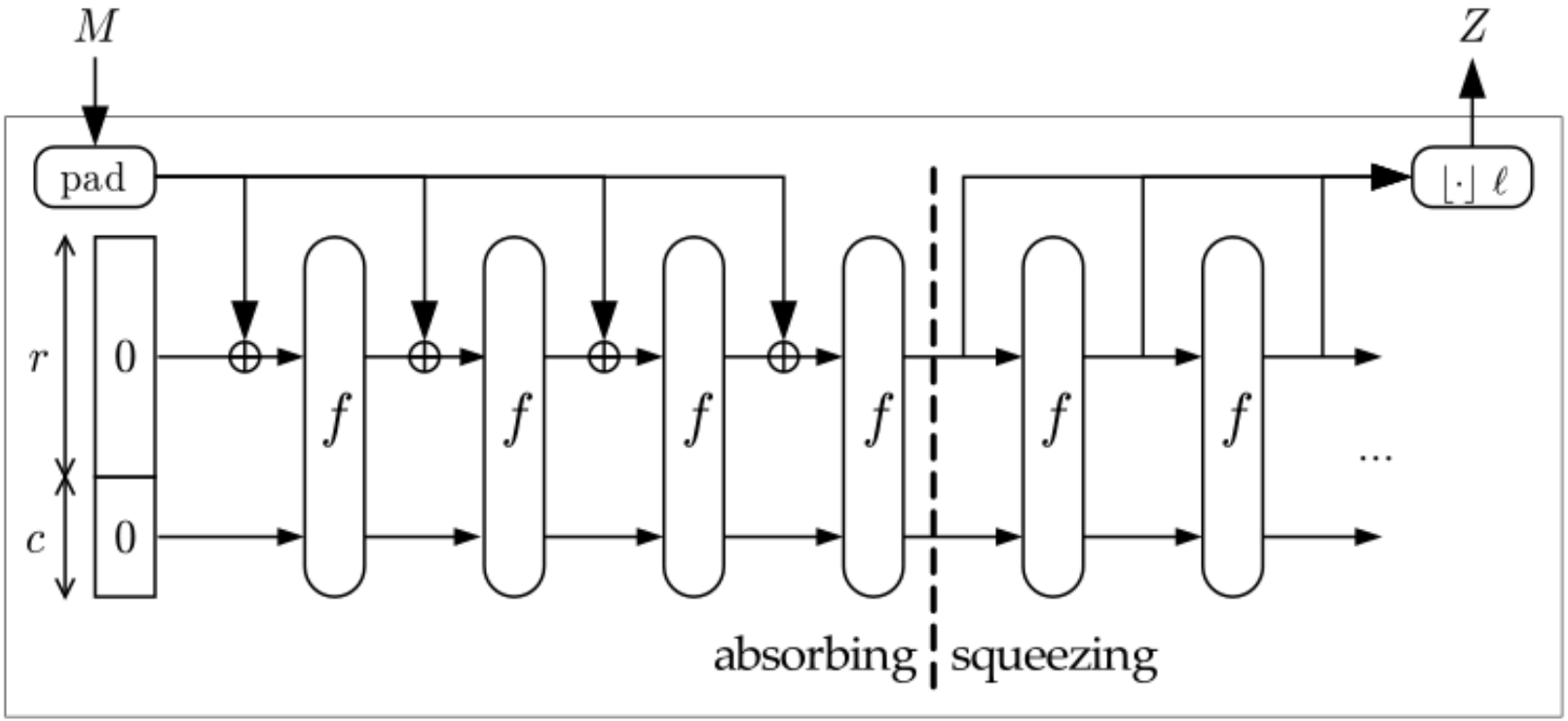
Goal:

- *Hash function:* h: $\{0,1\}^* \rightarrow \{0,1\}^n$
  (actually h: $\{0,1\}^* \rightarrow \{0,1\}^*$)

- (Already includes CF design, more later)

Sponges

- Used and introduced in SHA3 aka Keccak
  - Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche



sponge

Intermezzo: Random oracles

- Models the perfect hash function
- Truely random function without any structure
- Best attacks: Generic attacks (No structure available!)

Issue:

- No way to build a RO with polynomial-size description

Mind Model:

- Lazy-sampling
  - Imagine a black box implementing the function
  - For every new query, a random response is sampled
  - For old queries, former response is used

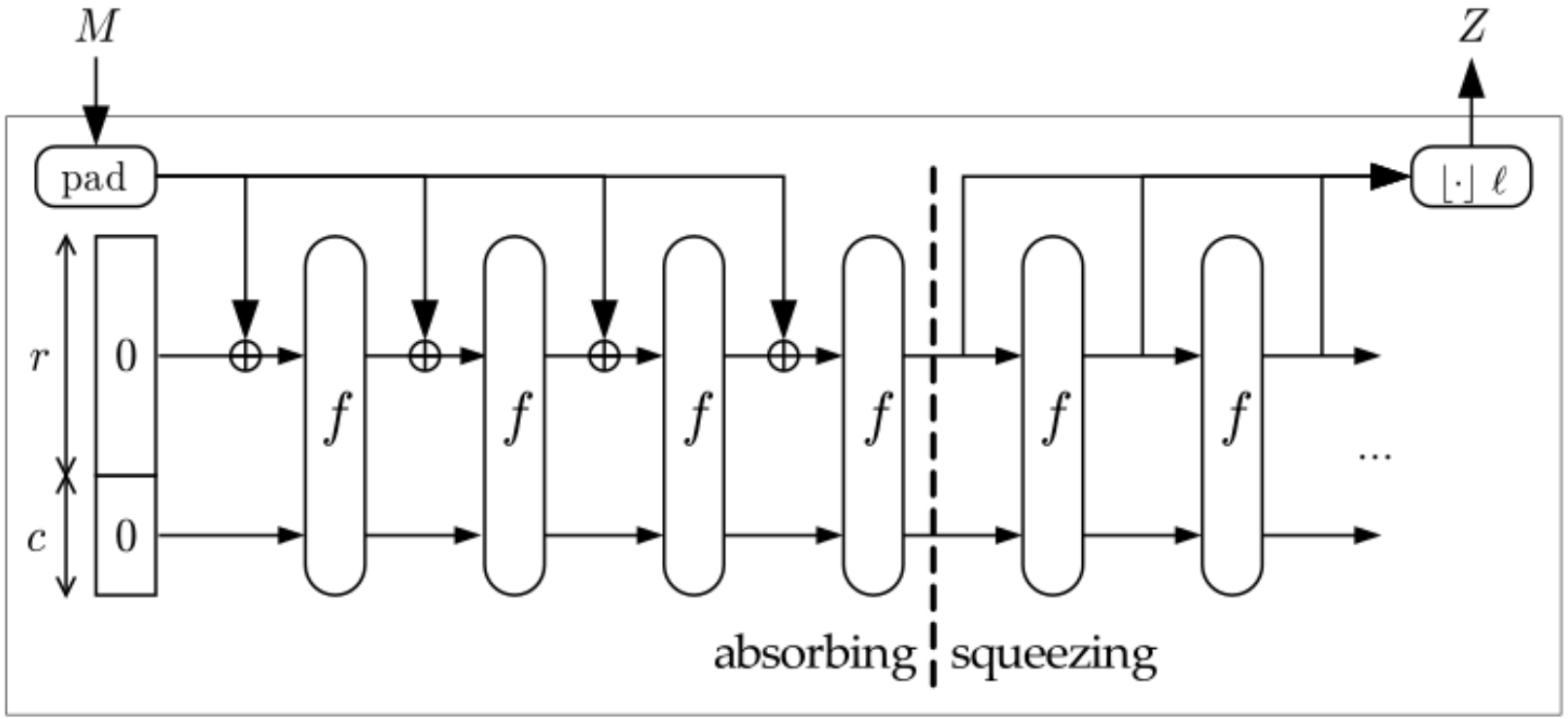
Sponge security

- Theorem (Indifferentiability from a random oracle):
  If f is a random permutation, the expected complexity for differentiating a sponge from a random oracle is $\sqrt{\pi}\,\mathbf{2}^{c/2}$.

- Note:
  - Neat way to simplify security arguments
  - Implies bounds for all attacks that use less than $\sqrt{\pi}\,\mathbf{2}^{c/2}$ queries
  - Bounds are those of generic attacks against a random oracle

Sponges

- Used and introduced in SHA3 aka Keccak
  - Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche

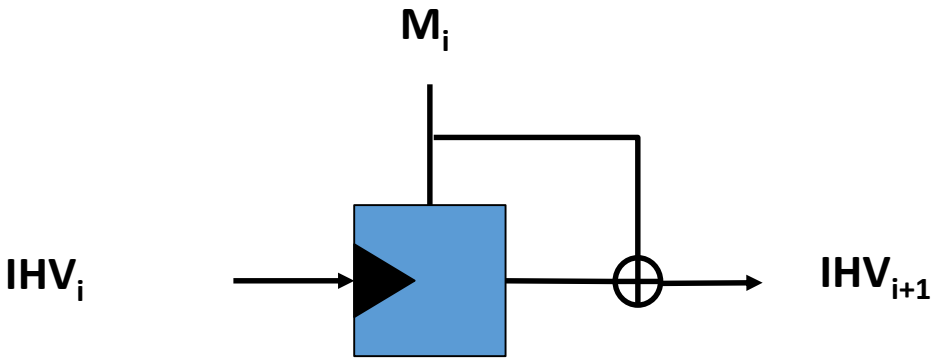


sponge

Compression Function Design

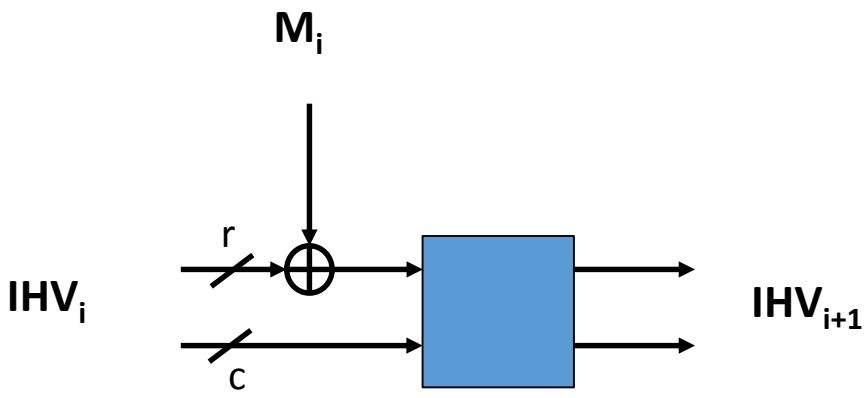
Block-Cipher-based designs

- Traditional approach
- Many possible modes
  - see Preneel, Govaerts, Vandewalle. Hash functions based on block ciphers: a synthetic approach. CRYPTO'93
  - security: Black, Rogaway, Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. CRYPTO'02
- Most popular: Matyas-Meyer-Oseas



Permutation-based designs

- Less frequent use
- Keccak compression function:



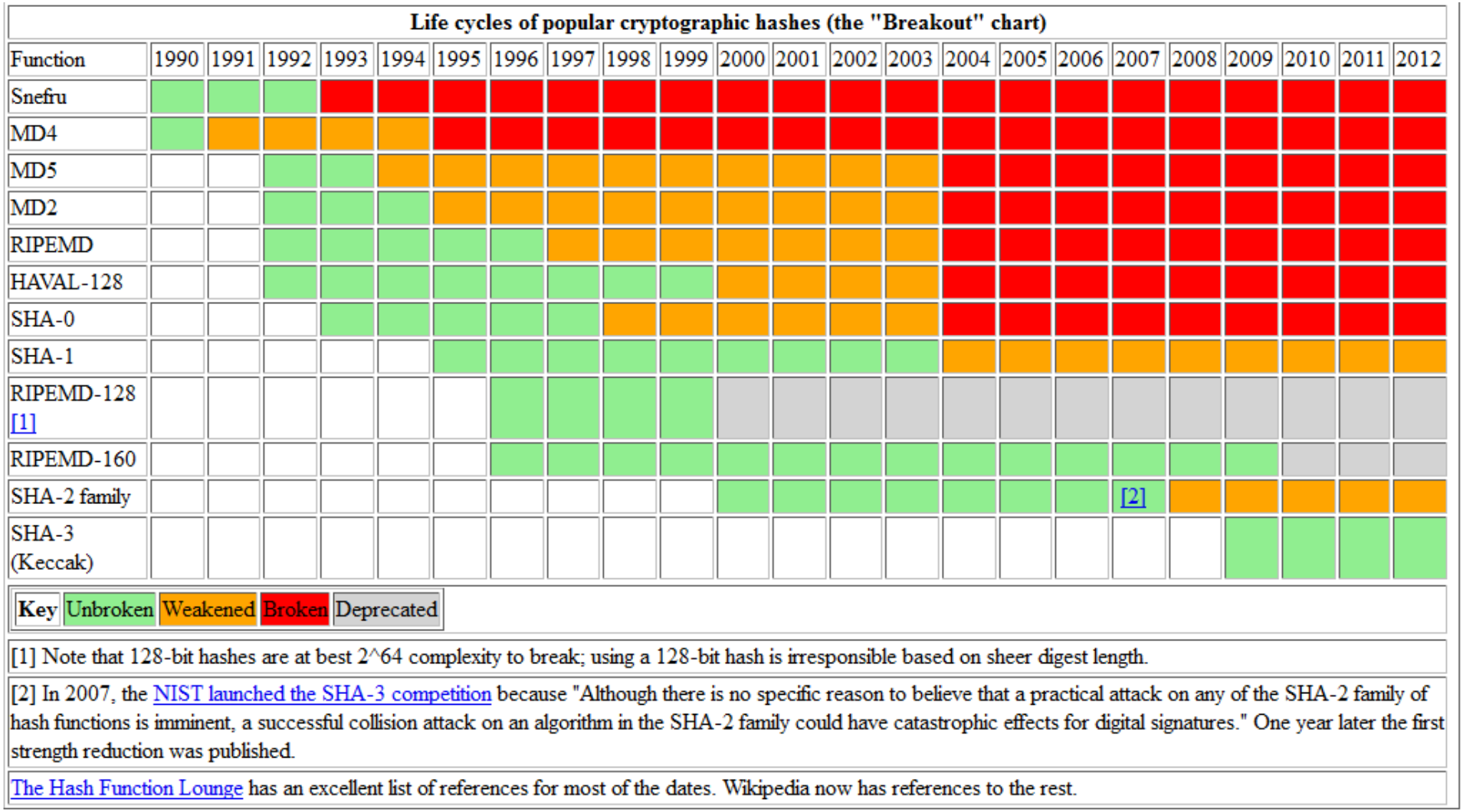
- Important: NEVER hand out last c bits of IHV!

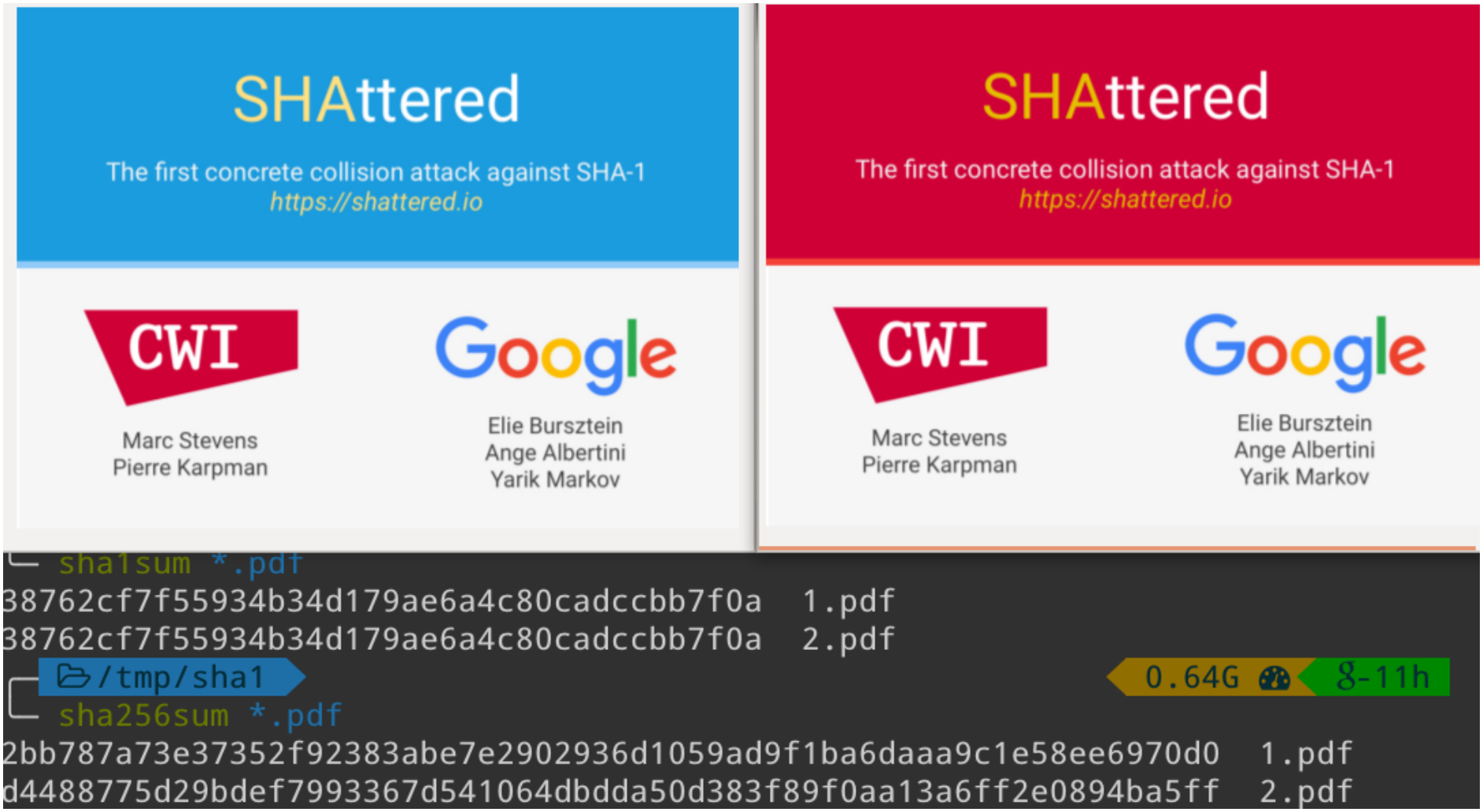
Common security assessment

- Provable security analyzed in idealized models:
  - „Black-box models“ (e.g. ideal cipher, random oracle, or random permutation model)
- Proofs assuming underlying building block behaves like such an idealized building block
- Building blocks then get cryptanalyzed, searching for “non-random behavior”

Provable hash functions

- people don’t like that one can’t prove much about hash functions
- reduction to established ‘hard problem’ such as factoring is seen as an advantage
- Example: VSH – Very Smooth Hash
  - Contini-Lenstra-Steinfeld 2006
  - collision resistance provable under assumption that a problem directly related to factoring is hard
  - but still far from ideal
    - bad performance compared to SHA-256
    - all kinds of multiplicative relations between hash values exist

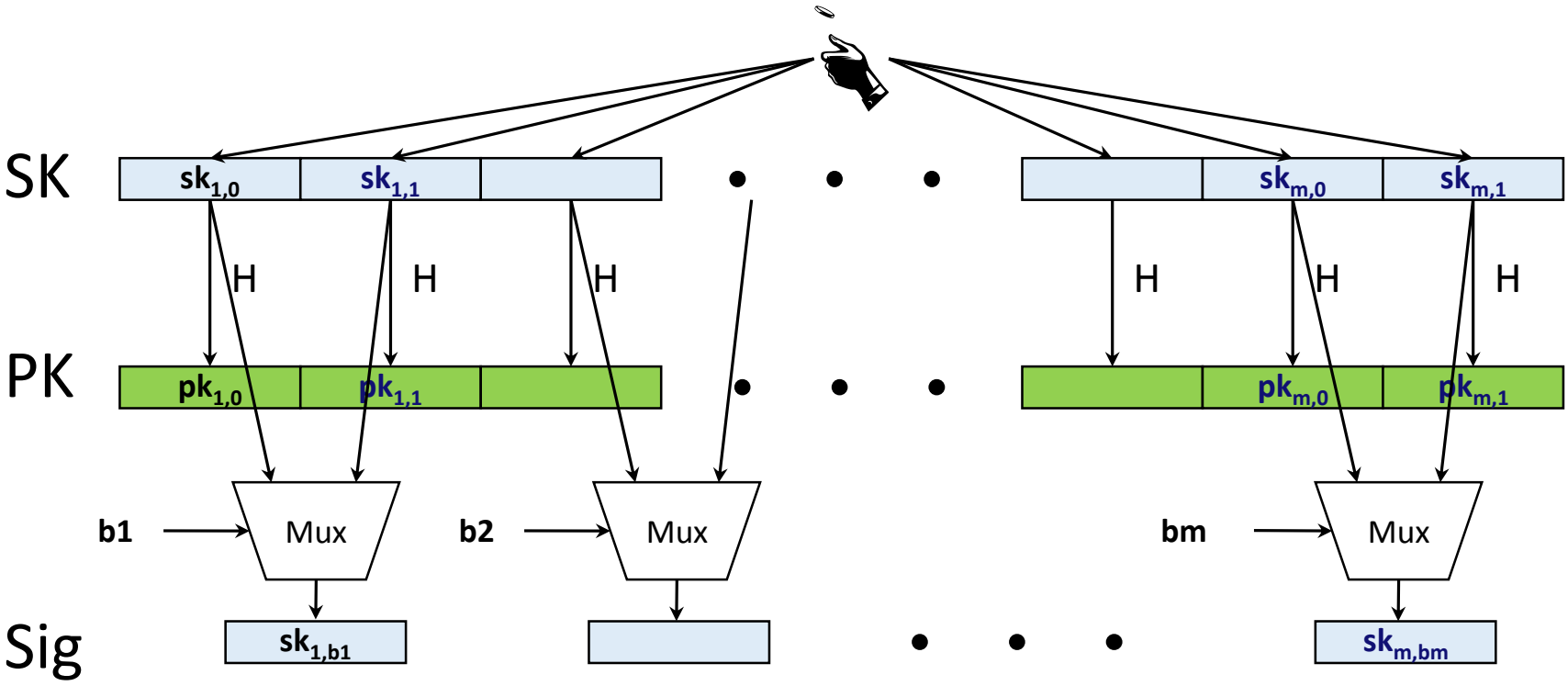

Life cycles of popular cryptographic hashes (the "Breakout" chart)
Function
1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
Snefru
MD4
MD5
MD2
RIPEMD
HAVAL-128
SHA-0
SHA-1
RIPEMD-128 [1]
RIPEMD-160
SHA-2 family
[2]
SHA-3 (Keccak)
Key
Unbroken
Weakened
Broken
Deprecated
[1] Note that 128-bit hashes are at best 2^64 complexity to break; using a 128-bit hash is irresponsible based on sheer digest length.
[2] In 2007, the NIST launched the SHA-3 competition because "Although there is no specific reason to believe that a practical attack on any of the SHA-2 family of hash functions is imminent, a successful collision attack on an algorithm in the SHA-2 family could have catastrophic effects for digital signatures." One year later the first strength reduction was published.
The Hash Function Lounge has an excellent list of references for most of the dates. Wikipedia now has references to the rest.


SHAttered
The first concrete collision attack against SHA-1
https://shattered.io
CWI
Marc Stevens
Pierre Karpman
Google
Elie Bursztein
Ange Albertini
Yarik Markov
SHAttered
The first concrete collision attack against SHA-1
https://shattered.io
CWI
Marc Stevens
Pierre Karpman
Google
Elie Bursztein
Ange Albertini
Yarik Markov
sha1sum *.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a 1.pdf
38762cf7f55934b34d179ae6a4c80cadccbb7f0a 2.pdf
/tmp/sha1
0.64G
8-11h
sha256sum *.pdf
2bb787a73e37352f92383abe7e2902936d1059ad9f1ba6daaa9c1e58ee6970d0 1.pdf
d4488775d29bdef7993367d541064dbdda50d383f89f0aa13a6ff2e0894ba5ff 2.pdf

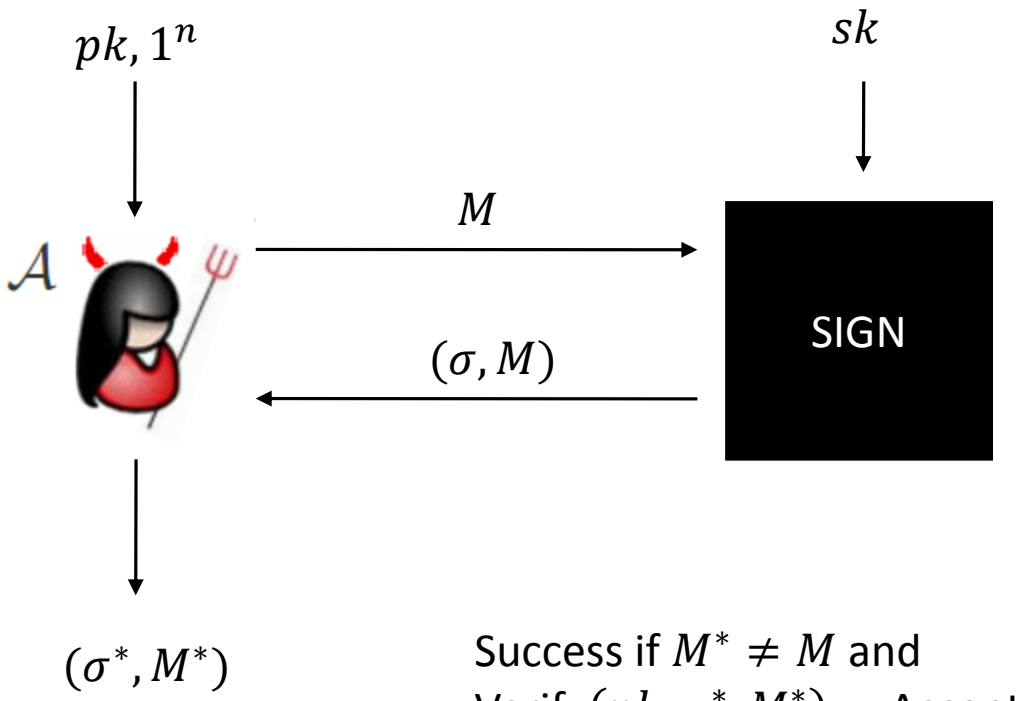
Hash-based signatures

Lamport-Diffie OTS [Lam79]

Message M = b1,…,bm, OWF H * = n bit



EU-CMA for OTS



Success if $M^* \neq M$ and
Verify$(pk, \sigma^*, M^*) =$ Accept

Security

Theorem:

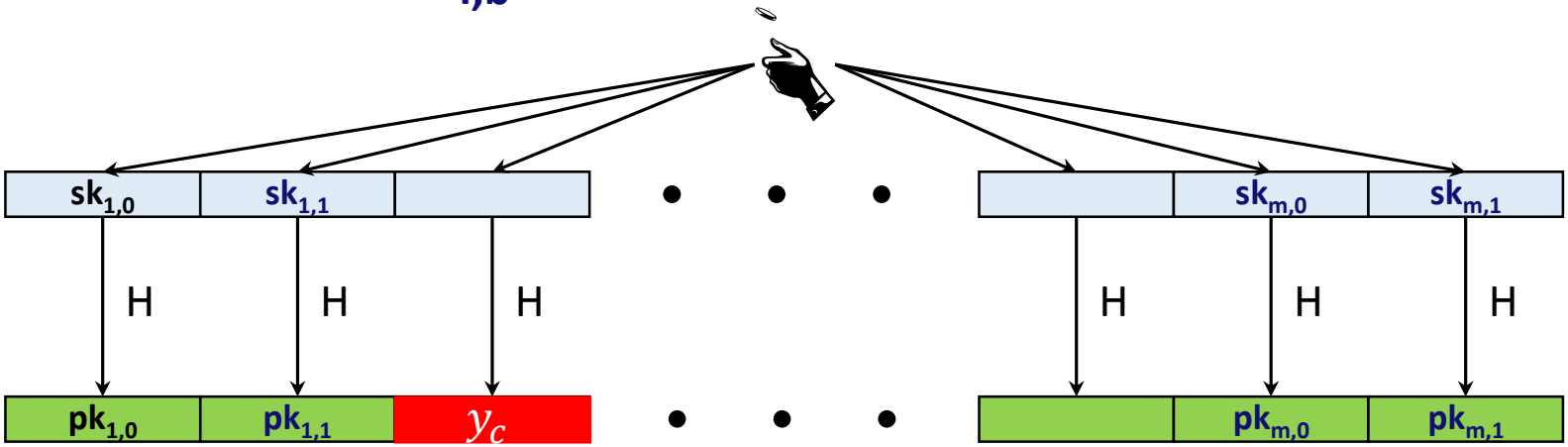
If H is one-way then LD-OTS is one-time eu-cma-secure.

Reduction

Input: y_c, k

Set $H \leftarrow h_k$

Replace random $\mathbf{pk_{i,b}}$



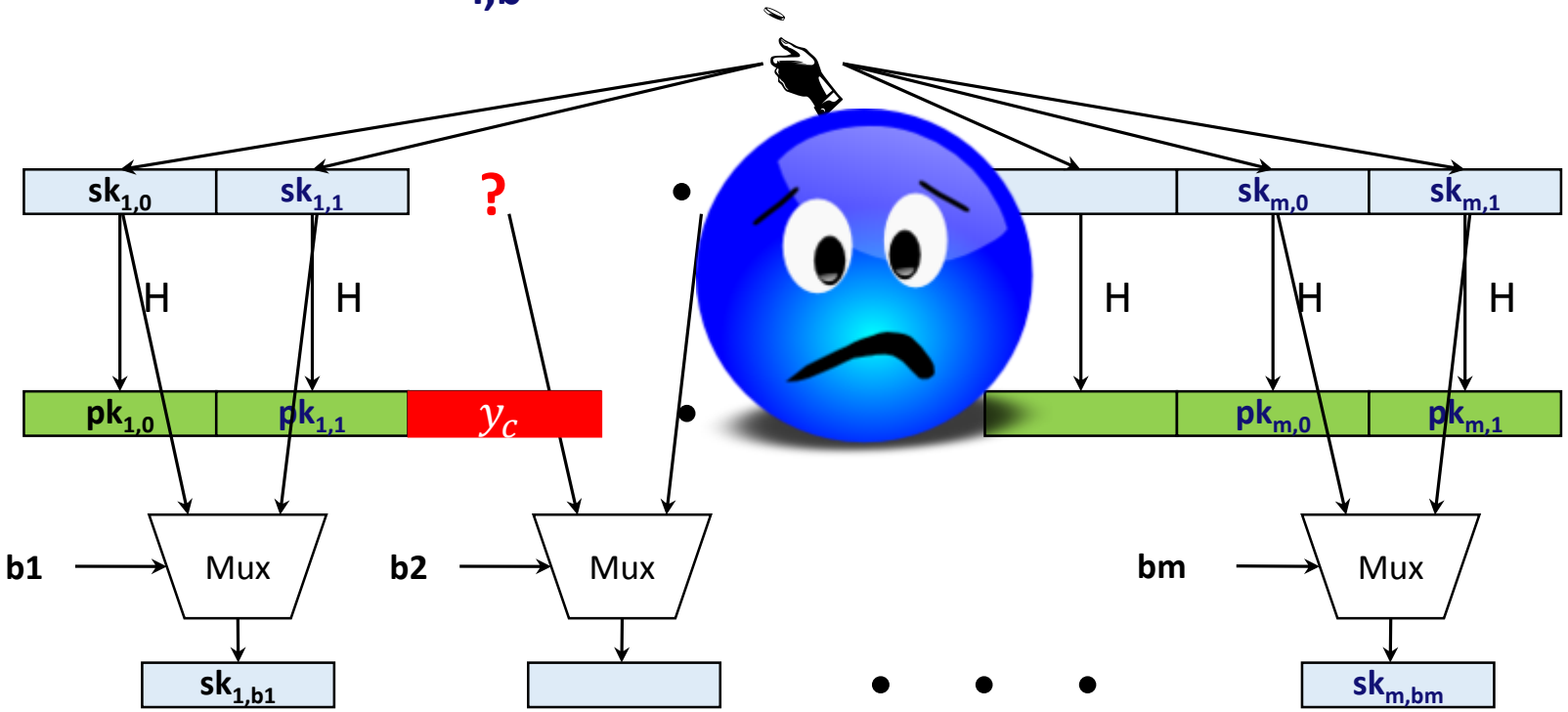
Reduction

Input: y_c, k

Set $H \leftarrow h_k$

Replace random $\mathbf{pk_{i,b}}$

Adv. Message: M = b1,…,bm
If bi = b return fail
else return Sign(M)



Reduction

Input: y_c, k

Set $H \leftarrow h_k$

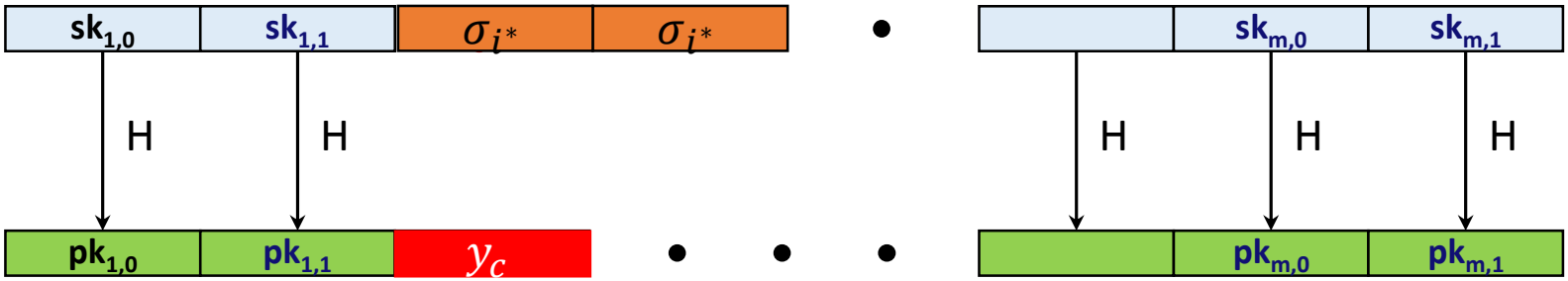
Choose random $\mathbf{pk_{i,b}}$

Forgery: M* = b1*,…,bm*,

$$\sigma = \sigma_1, \ldots, \sigma_m$$

If bi ≠ b return fail

Else return σ_{i^*}



Reduction - Analysis

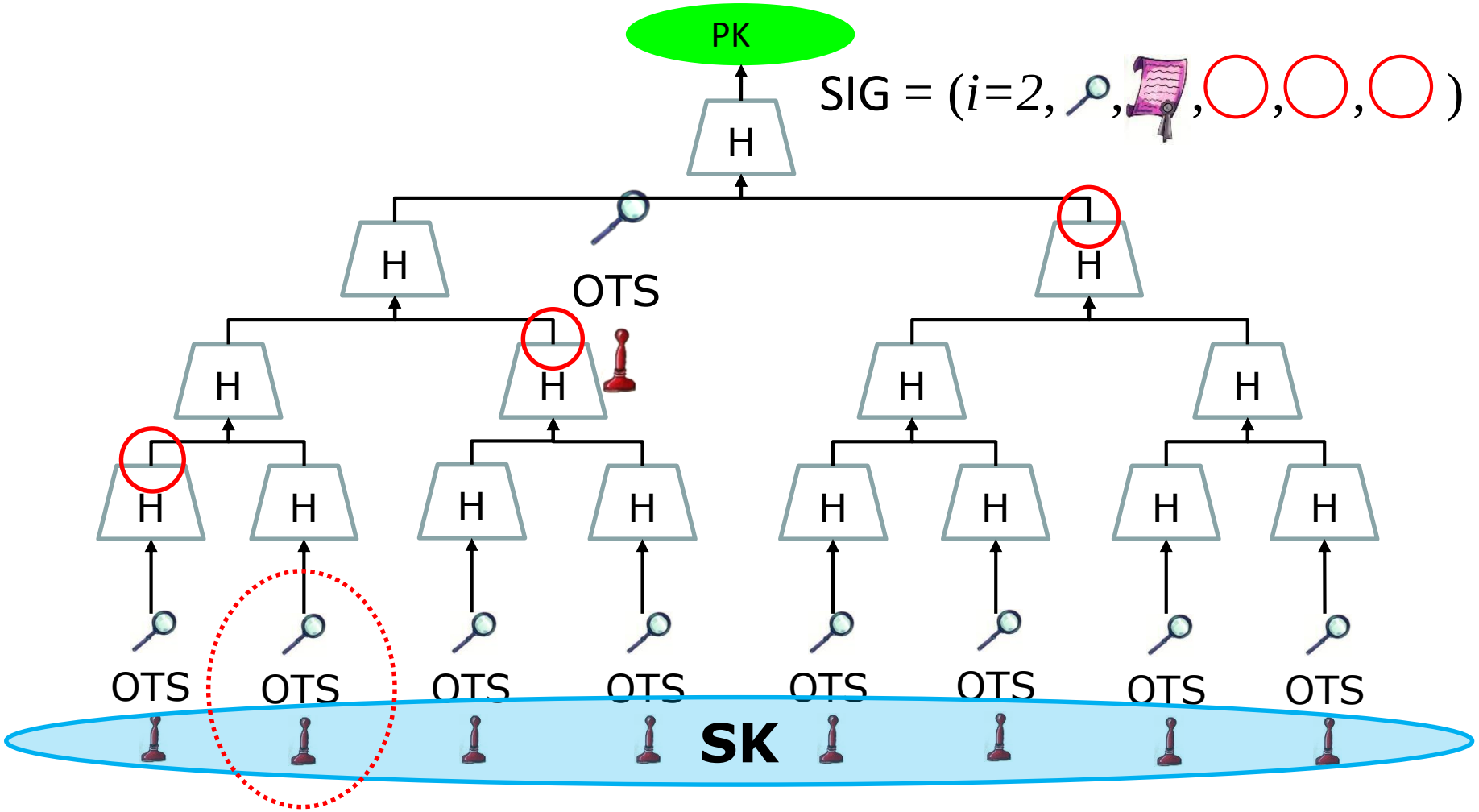
Abort in two cases:

1. $b_i = b$
   probability ½ : b is a random bit
2. $b_i \neq b$

   probability 1 - 1/m: At least one bit has to flip as $M^* \neq M$

Reduction succeeds with A‘s success probability times 1/2m.

Merkle's Hash-based Signatures



Security

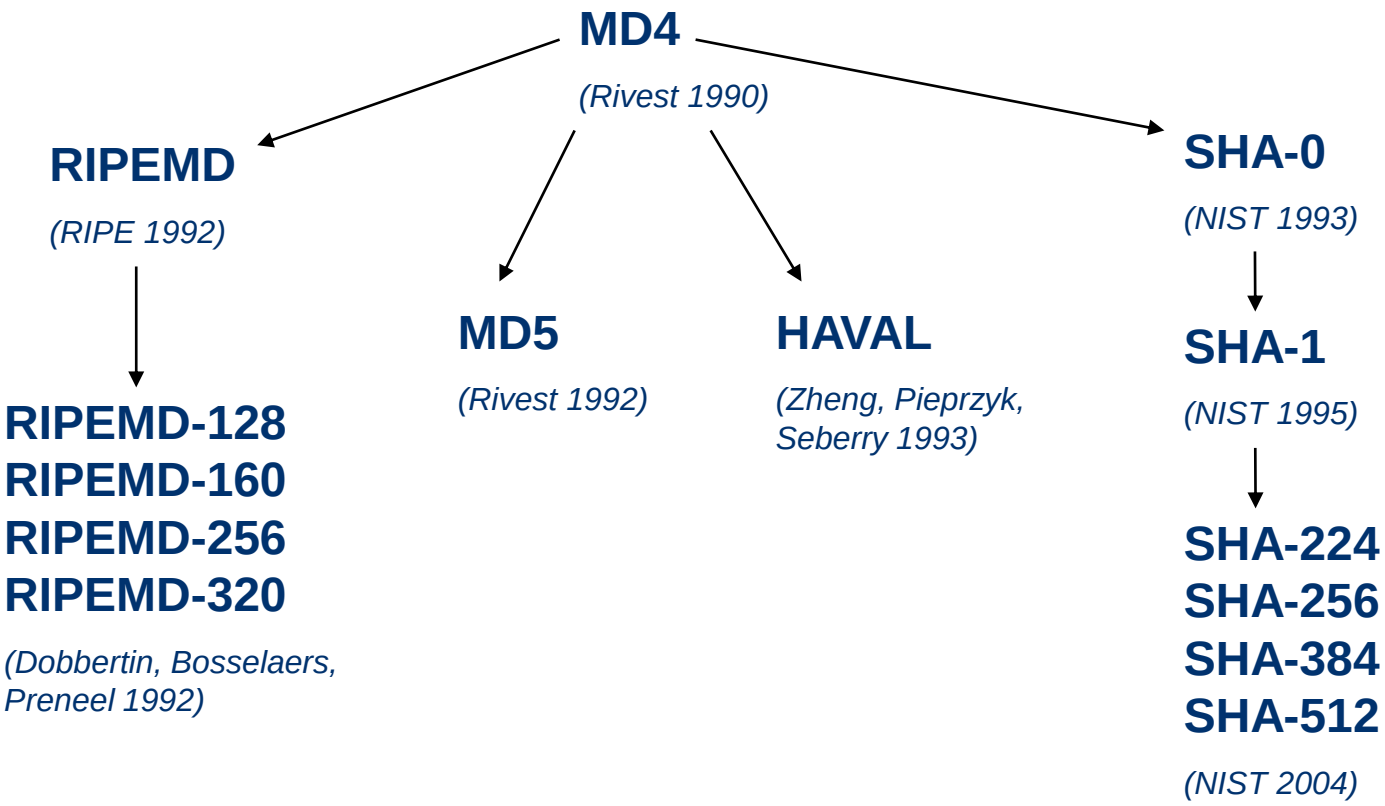
Theorem:

MSS is eu-cma-secure if OTS is a one-time eu-cma secure signature scheme and H is a random element from a family of collision resistant hash functions.

Questions?

Basic Building Blocks

the MD4 family of hash functions



design of MD4 family compression functions

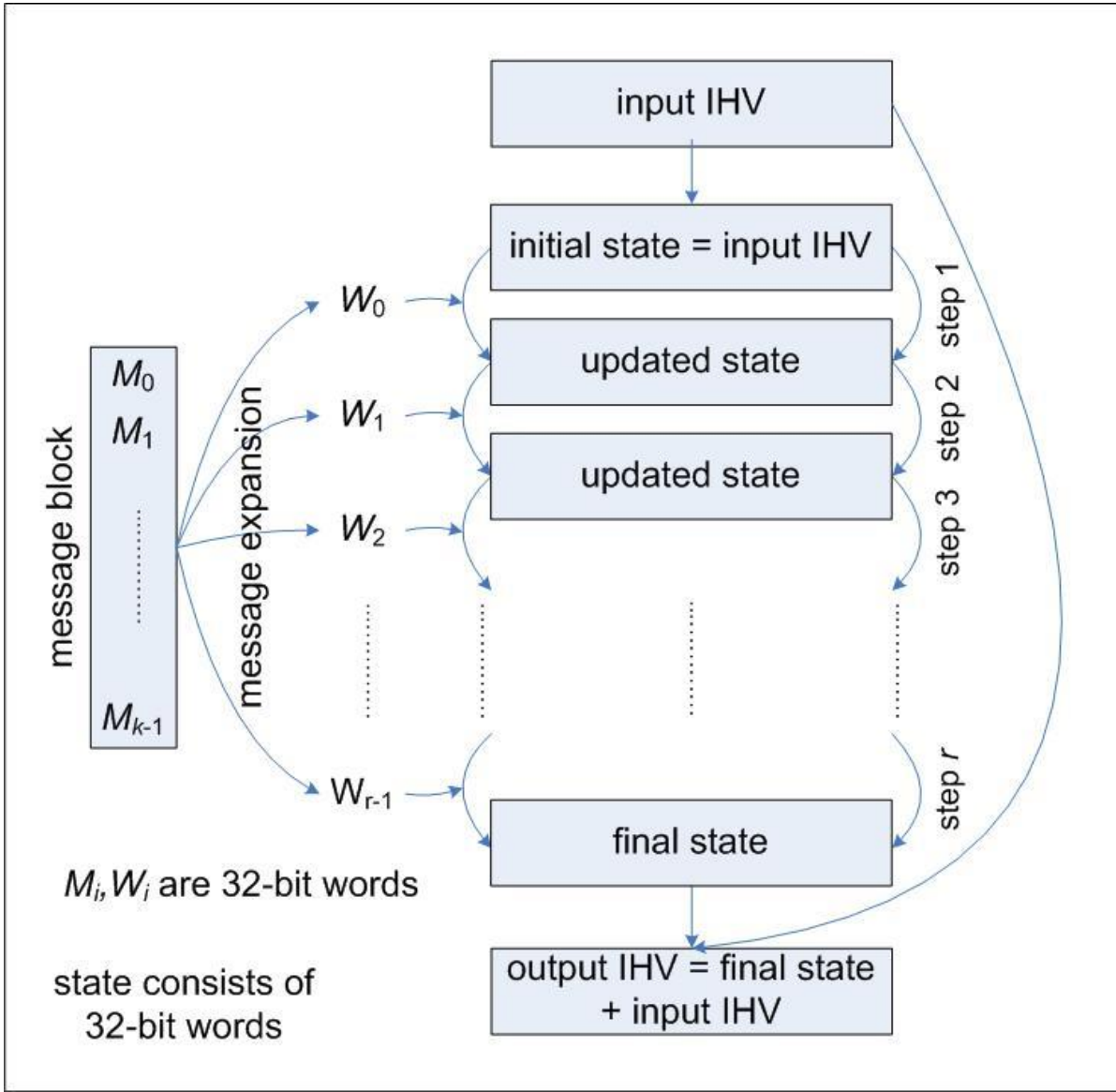
message block
  split into words

message expansion
  input words for each step

IHV → initial state

each step updates state with an input word

final state 'added' to *IHV*
  (feed-forward)



design details

- MD4, MD5, SHA-0, SHA-1 details:
  - 512-bit message block split into 16 32-bit words
  - state consists of 4 (MD4, MD5) or 5 (SHA-0, SHA-1) 32-bit words
  - MD4: 3 rounds of 16 steps each, so 48 steps, 48 input words
  - MD5: 4 rounds of 16 steps each, so 64 steps, 64 input words
  - SHA-0, SHA-1: 4 rounds of 20 steps each, so 80 steps, 80 input words
  - message expansion and step operations use only very easy to implement operations:
    - bitwise Boolean operations
    - bit shifts and bit rotations
    - addition modulo 2^{32}
  - proper mixing believed to be cryptographically strong

message expansion

- MD4, MD5 use *roundwise permutation*, for MD5:
  - $W_0 = M_0$, $W_1 = M_1$, …, $W_{15} = M_{15}$,
  - $W_{16} = M_1$, $W_{17} = M_6$, …, $W_{31} = M_{12}$, (jump 5 mod 16)
  - $W_{32} = M_5$, $W_{33} = M_8$, …, $W_{47} = M_2$, (jump 3 mod 16)
  - $W_{48} = M_0$, $W_{49} = M_7$, …, $W_{63} = M_9$ (jump 7 mod 16)
- SHA-0, SHA-1 use *recursivity*
  - $W_0 = M_0$, $W_1 = M_1$, …, $W_{15} = M_{15}$,
  - SHA-0: $W_i = W_{i-3}$ XOR W_{i-8} XOR W_{i-14} XOR W_{i-16} for i = 16, …, 79
  - problem: k^{th} bit influenced only by k^{th} bits of preceding words, so not much diffusion
  - SHA-1: $W_i = (W_{i-3}$ XOR W_{i-8} XOR W_{i-14} XOR $W_{i-16})<<<1$
    (additional rotation by 1 bit,
    this is the *only* difference between SHA-0 and SHA-1)

Example: step operations in MD5

- in each step only one state word is updated
- the other state words are *rotated* by 1
- state update:

  $A' = B + ((A + f_i(B,C,D) + W_i + K_i) <<< s_i)$

  K_i, s_i step dependent constants,

  $+$ is addition mod 2^{32},

  f_i round dependend boolean functions:

  $f_i(x,y,z) = xy \; OR \; (\neg x)z$ for $i = 1, \ldots, 16$,

  $f_i(x,y,z) = xz \; OR \; y(\neg z)$ for $i = 17, \ldots, 32$,

  $f_i(x,y,z) = x \; XOR \; y \; XOR \; z$ for $i = 33, \ldots, 48$,

  $f_i(x,y,z) = y \; XOR \; (y \; OR \; (\neg z))$ for $i = 49, \ldots, 64$,

  these functions are nonlinear, balanced, and have an *avalanche effect*

Step operations in MD5