Design Principles for Hash Functions Revisited

Bart Preneel

Katholieke Universiteit Leuven, COSIC

`bartDOTpreneel(AT)esatDOTkuleuvenDOTbe`

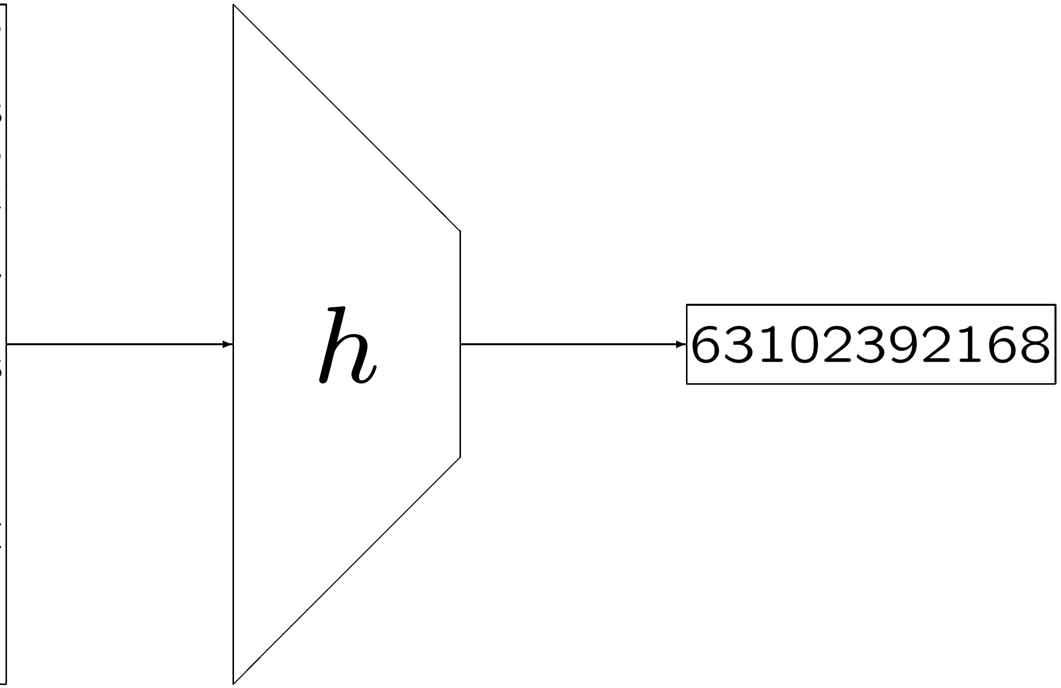
`http://homes.esat.kuleuven.be/~preneel`

October 2005

Outline

- Definitions
- Generic Attacks
- Constructions
- A Comment on HMAC
- Conclusions

are secure; they can be reduced to 2 classes based on linear transformations of variables. The properties of these 12 schemes with respect to weaknesses of the underlying block cipher are studied. The same approach can be extended to study keyed hash functions (MACs) based on block ciphers and hash functions based on modular arithmetic. Finally a new attack is presented on a scheme suggested by R. Merkle. This slide is now shown at the VI Spanish meeting on Information Security and Cryptology in a presentation on the state of hash functions.



Informal definitions (1)

- no secret parameters
- x arbitrary length $\Rightarrow$ fixed length n
- computation "easy"

One Way Hash Function (OWHF):

- preimage resistant: ! $h(x) \; \not\Rightarrow \; x'$ with $h(x) = h(x')$
- 2nd preimage resistant:
  ! $x, h(x) \not\Rightarrow x'(\neq x)$ with $h(x') = h(x)$

Collision Resistant Hash Function (CRHF) = OWHF +

- collision resistant:
  $\not\Rightarrow$ x, $x'(x' \neq x)$ with $h(x) = h(x')$.

Informal definitions (2)

preimage resistant $\not\Rightarrow$ 2nd preimage resistant

- take a preimage resistant hash function; add an input bit b and replace one input bit by the sum modulo 2 of this input bit and b

2nd preimage resistant $\not\Rightarrow$ preimage resistant

- if h is OWHF, $\overline{h}$ is 2nd preimage resistant but not preimage resistant

$$\overline{h}(X) = \begin{cases} 0\|X & \text{if } |X| \leq n \\ 1\|h(X) & \text{otherwise.} \end{cases}$$

collision resistant $\Rightarrow$ 2nd preimage resistant

[Simon 98] one cannot derive collision resistance from ‘general’ preimage resistance

Formal definitions: (2nd) preimage resistance

Notation: $\Sigma = \{0, 1\}$, $l(n) > n$

A **one-way** hash function H is a function with domain $D = \Sigma^{l(n)}$ and range $R = \Sigma^n$ that satisfies the following conditions:

- preimage resistance: let x be selected uniformly in D and let M be an adversary that on input $h(x)$ uses time $\leq t$ and outputs $M(h(x)) \in D$. For each adversary M,

  $$\Pr_{x \in D} \{h(M(h(x))) = h(x)\} < \epsilon \,.$$

  Here the probability is also taken over the random choices of M.

- 2nd preimage resistance: let x be selected uniformly in $\Sigma^{l(n)}$ and let M' be an adversary that on input x uses time $\leq t$ and outputs $x' \in D$ with $x' \neq x$. For each adversary M',

  $$\Pr_{x \in D} \left\{M'(x) = h(x)\right\} < \epsilon \,.$$

  Here the probability is taken over the random choices of M'.

Formal definitions: collision resistance

A **collision-resistant** hash function $\mathcal{H}$ is a function family with domain $D = \Sigma^{l(n)}$ and range $R = \Sigma^n$ that satisfies the following conditions:

- (the functions h_S are preimage resistant and second preimage resistant)
- collision resistance: let F be a collision string finder that on input $S \in \Sigma^s$ uses time $\leq t$ and outputs either "?" or a pair $x, x' \in \Sigma^{l(n)}$ with $x' \neq x$ such that $h_S(x') = h_S(x)$. For each F,

  $$\Pr_S \{F(\mathcal{H}) \neq \text{"?"}\} < \epsilon\,.$$

  Here the probability is also taken over the random choices of F.

Further generalization: Rogaway-Shrimpton, FSE 2004

Consider a family of hash functions.
For (2nd) preimage resistance, one can choose the challenge (x) and/or the key that selects the function.
This gives three flavours:

- random challenge, random key (Pre and Sec)
- random key, fixed challenge (ePre and eSec – everywhere)
- fixed key, random challenge (aPre and aSec – always)

Complex relationship (see figure on next slide).

Relation between definitions: Rogaway-Shrimpton

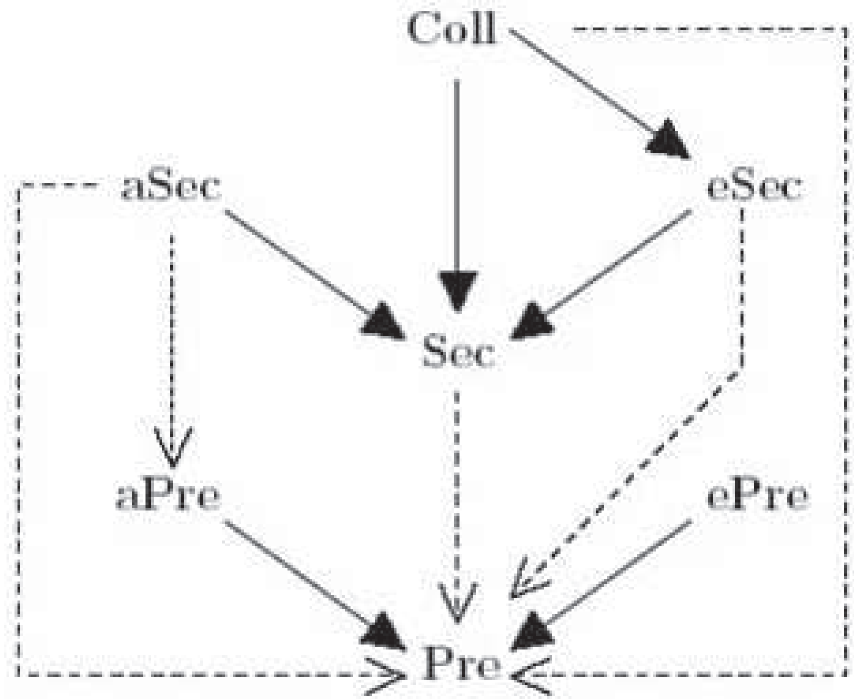


Figure 1: Summary of the relationships among seven notions of hash-function security. Solid arrows represent conventional implications, dotted arrows represent provisional implications (their strength depends on the relative size of the domain and range), and the lack of an arrow represents a separation.

Applications

- digital signatures: OWHF/CRHF, ‘destroy algebraic structure’
- information authentication: protect authenticity of hash result
- (redundancy: hash result appended to data before encryption)
- protection of passwords: preimage resistant
- confirmation of knowledge/commitment: OWHF/CRHF
- pseudo-random string generation/key derivation
- micropayments (e.g., micromint)
- construction of MACs, stream ciphers, block ciphers

collision resistance is not always necessary
but other properties may be needed: pseudo-randomness if keyed,
near-collision resistance, partial preimage resistance,. . .
$\rightsquigarrow$ how to formalize?

Related definitions: UOWH

UOWH or Universal One-Way Hash Function
(TCR: target collision resistant hash functions or eSec)

- generate message x (+ some state)
- choose a random key K
- target collision finder algorithm:
  given x, K, $h()$ (+state), find $x' \neq x$ such that $h_K(x') = h_K(x)$

Generic Attacks (1)

depend only on size of hash result; not on details of the algorithm

guess (2nd) preimage: Pr. success $= \dfrac{(\#\text{trials}) \cdot (\#\text{targets})}{2^n}$

small if $n \geq 80 \ldots 128$
avoid simultaneous attack on all targets:
parameterize ('tweak') hash function

collision: birthday attack (or square root attack) [Yuval'79]

- r variations on genuine message
- r variations on fraudulent message
- probability of a match: 63% for $r = \sqrt{2^n} = 2^{n/2}$

infeasible in 2005 if $n \geq 160$, but this is not sufficient for long-term security.

Generic Attacks (2): time-memory trade-off

multiple preimages of the same function [Hellman80]:

- $O(2^n)$ precomputation, $O(2^{2n/3})$ storage
- single inversion in time $O(2^{2n/3})$

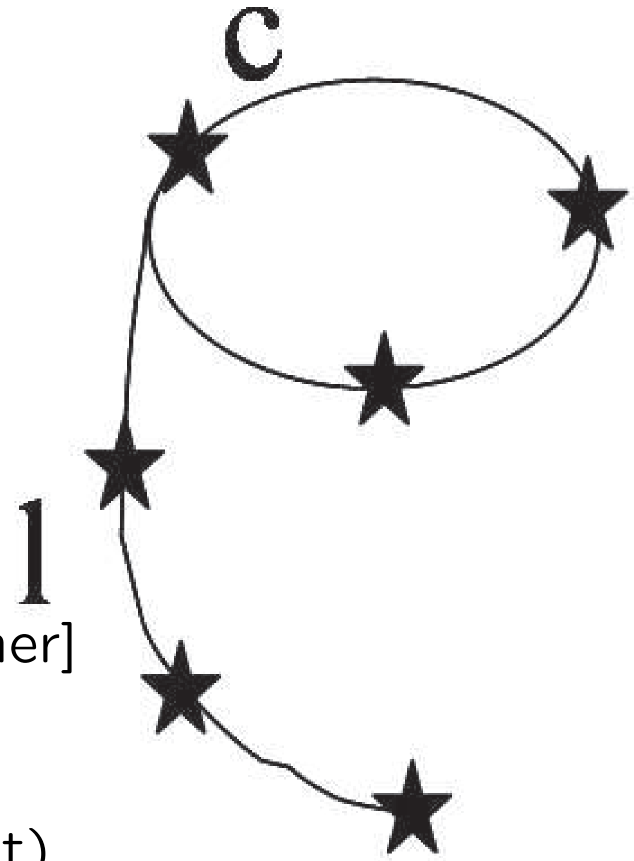
[Wiener02] If $\Theta(2^{3n/5})$ preimages are searched, the full cost per preimage decreases from $\Theta(2^n)$ to $\Theta(2^{2n/5})$.

Full cost: product of number of components with the duration of their use (motivation: hardware = ALUs, memory chips, wires, switching elements)

Generic Attacks (3): the birthday attack

Efficient implementations of the birthday attack

- very little memory: cycle finding algorit
- full parallelism



Distinguished point: $l = c = (\pi/8) \cdot 2^{n/2}$
$\Theta(e2^{n/2} + e2^{d+1})$ steps
$\Theta(n2^{n/2-d})$ memory
with e the cost of evaluating the function f

Full cost [Wiener02]: $\Theta\left(en2^{n/2}\right)$

In practice 10 million $ [van Oorschot-Wiener]

- $n = 128$: 5 hours in 2004
- $n = 160$: 37 years (with 2004 equipment)

Generic Attacks (4)

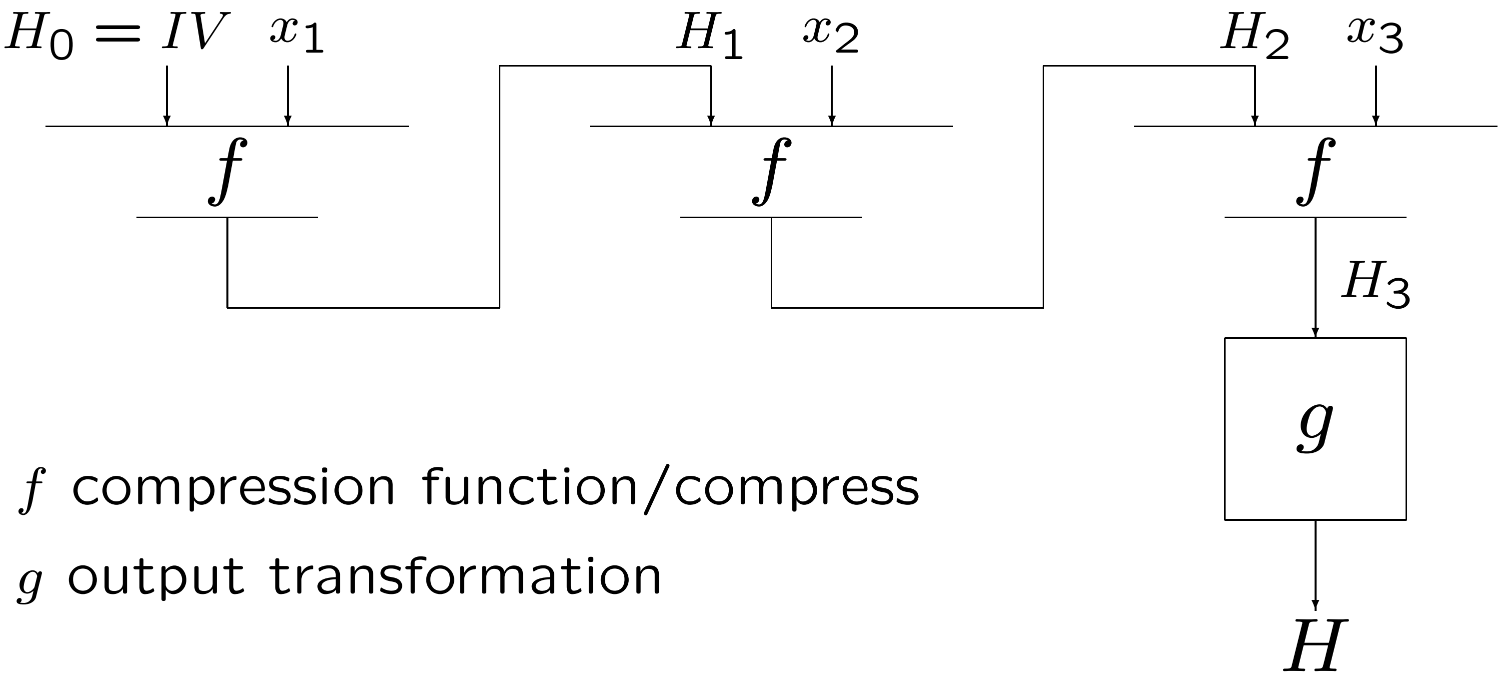
attacker	investment	tool	hash result: 2nd preimage 2006	2nd preimage 2015	collision 2006	collision 2015
Pedestrian Hacker	$400	FPGA	74	80	115	127
Small Business	$10,000	FPGA	79	85	125	137
Corporate Department	$300K	ASIC	90	96	147	159
Big Company	$10M	ASIC	95	101	158	169
Intelligence Agency	$300M	ASIC	100	106	162	174

Size of hash result to withstand a brute force 2nd preimage and collision attack during 1 year. For the 2nd preimage attack, it is assumed that 65,536 messages are attacked in parallel. Inspired by Blaze et al., 1996.

FPGA = Field Programmable Gate Array;

ASIC = Application Specific Integrated Circuit

Construction (1): iterated hash function



f compression function/compress

g output transformation

unambiguous padding of input to multiple of block length
divide input into blocks x_1, x_2,, . . . , x_t

Construction (2): relation between security f-h

iterating a compression function can make it less secure:

- trivial 2nd preimage/collision:
  replace IV by H_1 and delete the first message block x_1
- 2nd preimage attack for a message with t blocks:
  increases success probability with a factor of t
- fixed points: $f(H_{i-1}, x_i) = H_{i-1}$ can lead to trivial 2nd preimages or collisions

one possible solution: Merkle-Damgård strengthening

- fix IV and append input length in padding

cf. [Merkle, Crypto 89] and [Damgård, Crypto 89]

Construction (3): relation between security f-h

[Damgård-Merkle 89]
Let f be a collision resistant function mapping l to n bits (with $l > n$).

- If the padding contains the length of the input string, and if f is preimage resistant, the iterated hash function h based on f will be a CRHF.
- If an unambiguous padding rule is used, the following construction will yield a CRHF ($l - n > 1$):
  $H_1 = f(H_0 \parallel 0 \parallel x_1)$ and $H_i = f(H_{i-1} \parallel 1 \parallel x_i)\ \ i = 2, 3, \ldots t$.

Construction (4): relation between security f-h

[Lai-Massey 92]
Assume that the padding contains the length of the input string, and that the message X (without padding) contains at least two blocks. Then finding a second preimage for h with a fixed IV requires 2^n operations iff finding a second preimage for f with arbitrarily chosen H_{i-1} requires 2^n operations.

BUT:

- very few hash functions have a strong compression function
- very few hash functions are designed based on a strong compression function in the sense that they treat x_i and H_{i-1} in the *same way*.

Construction (5)

Observation: attacks on f do not necessarily mean attacks on h

- (2nd) preimage for f with chosen H_{i-1}
  (2nd) preimage for h
- pseudo-preimage: (2nd) preimage for f with random H_{i-1}
  preimage for h if IV can be changed
  (2nd) preimage for h in time $2^{(n+s)/2}$ if pseudo-preimage in 2^s
- pseudo-collision: collision for compress with $H_{i-1} \neq H'_{i-1}$
  certificational weakness only
- collision for f with random H_{i-1}
  collision for h if IV can be changed
- collision for f with chosen H_{i-1}
  collision for h

Defeating Merkle-Damgård for (2nd) preimages

[Dean-Felten-Hu'99] and [Kelsey-Schneier, Eurocrypt05]

Known since Merkle: if one hashes 2^t **messages**, the average effort to find a second preimage for one of them is 2^{n-t}.

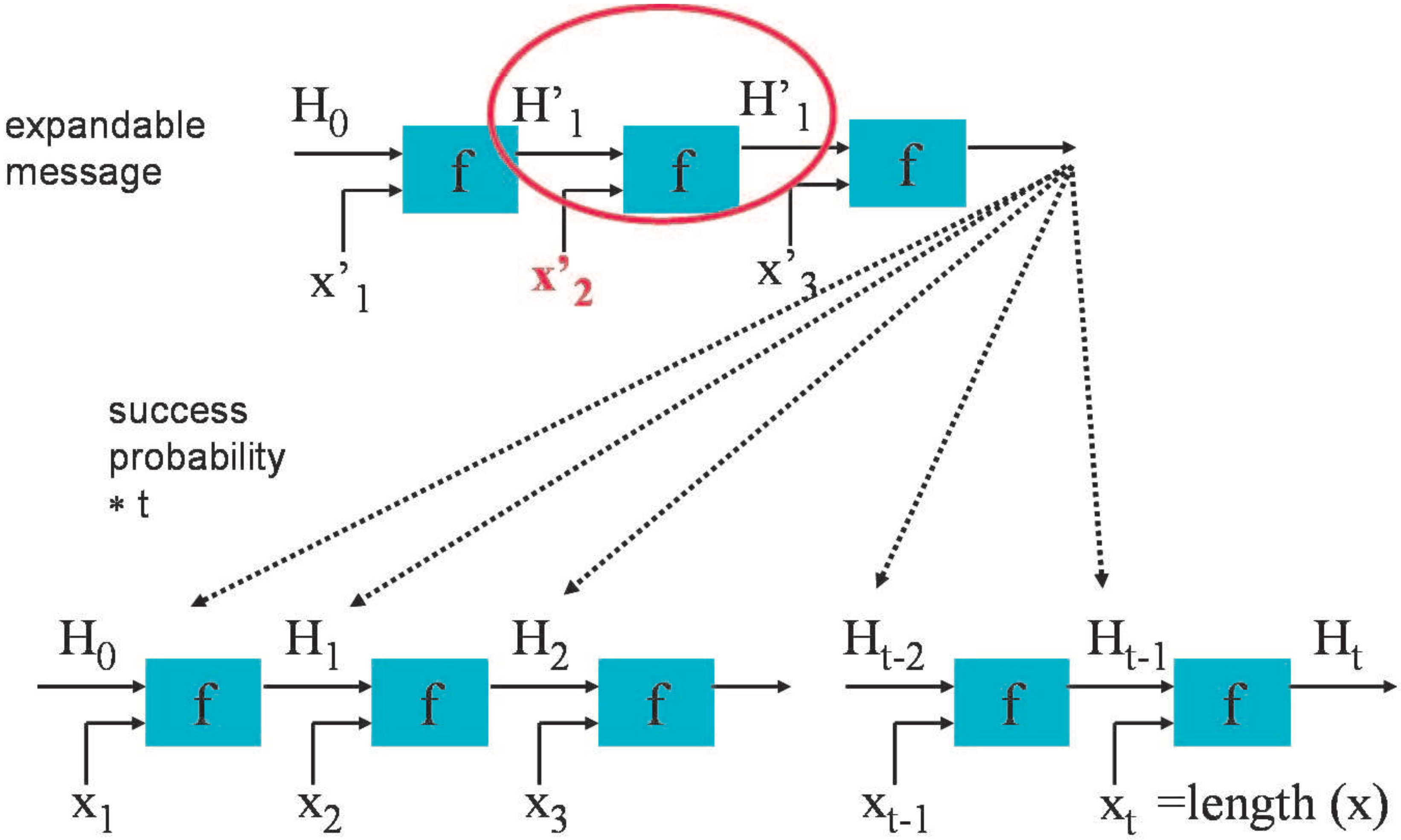
New: if one hashes 2^t message **blocks** with an iterated hash function, the effort to find a second preimage is only

$$t2^{n/2+1} + 2^{n-t+1}$$

Idea: use fixed points to match the correct length
Finding fixed points can be easy (e.g., Davies-Meyer).
But still very long messages

Conclusion: appending the length does not work for 2nd preimage attacks.

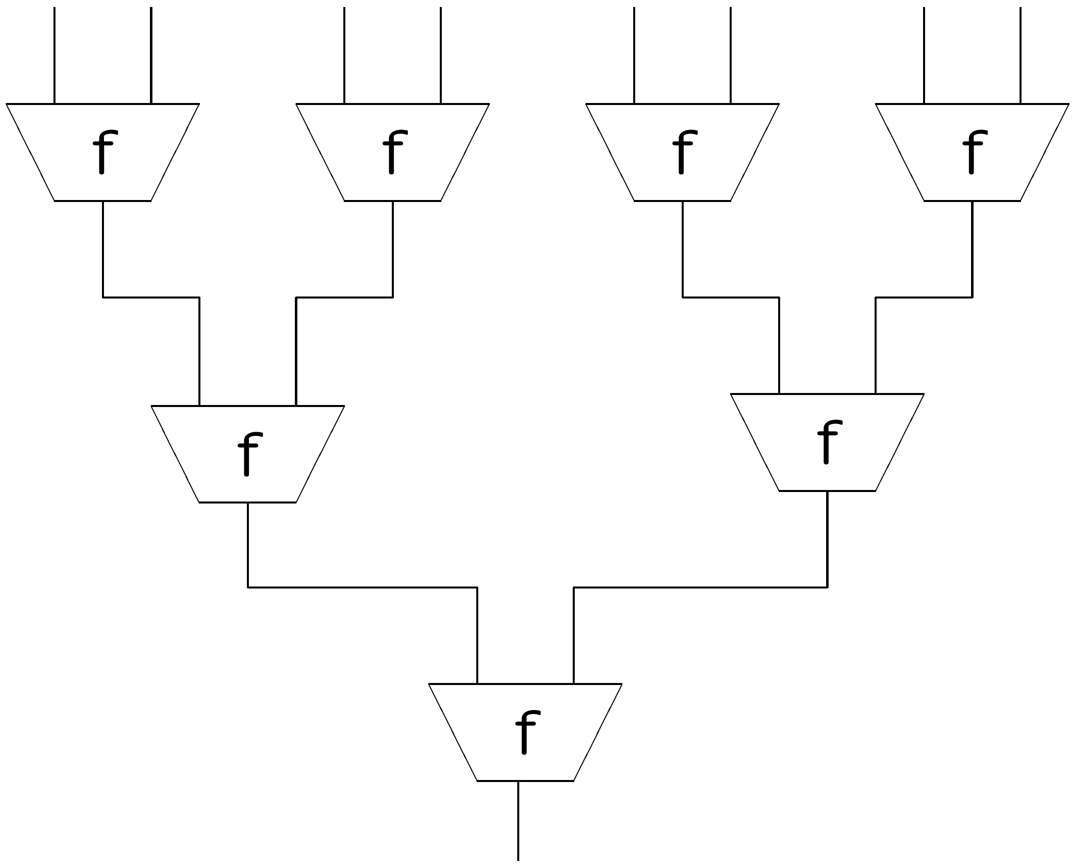
Defeating Merkle-Damgård for (2nd) preimages



$$h(x'_1 \mid\mid x'_2 \mid\mid x'_2 \mid\mid x'_2 \mid\mid x'_2 \mid\mid x'_3 \mid\mid \ldots \mid\mid x_{t-1} \mid\mid x_t) = h(x_1 \mid\mid x_2 \mid\mid x_3 \mid\mid \ldots \mid\mid x_{t-1} \mid\mid x_t)$$

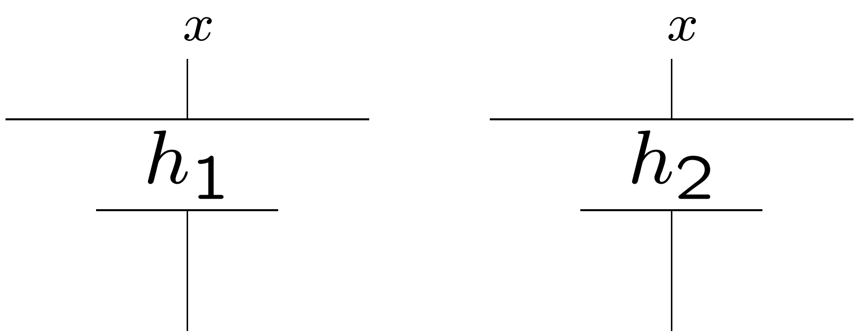
Tree Construction

Advantage of strong compression function f: tree construction.



How (not) to strengthen a hash function?

Answer concatenation:
Consider h_1 (n_1-bit result) and h_2 (n_2-bit result), with $n_1 \geq n_2$.



$$g(x) = h_1(x) || h_2(x)$$

Intuition: the strength of $g(x)$ is the product of the strength of the two hash functions (if both are “independent”).
But . . .

Multicollisions [Joux, Crypto 2004]

Consider h_1 (n_1-bit result) and h_2 (n_2-bit result), with $n_1 \geq n_2$.
The concatenation of two iterated hash functions ($g(x) = h_1(x)||h_2(x)$) is only as strong as the strongest of the two hash functions (even if both are independent).

- Cost of collision attack against g

$$\leq \; n_1 \cdot 2^{n_2/2} + 2^{n_1/2} \ll 2^{(n_1+n_2)/2}$$

- Cost of (2nd) preimage attack against g

$$\leq \; n_1 \cdot 2^{n_2/2} + 2^{n_1} + 2^{n_2} \ll 2^{n_1+n_2}$$

If either of the functions is weak, the attacks may work better

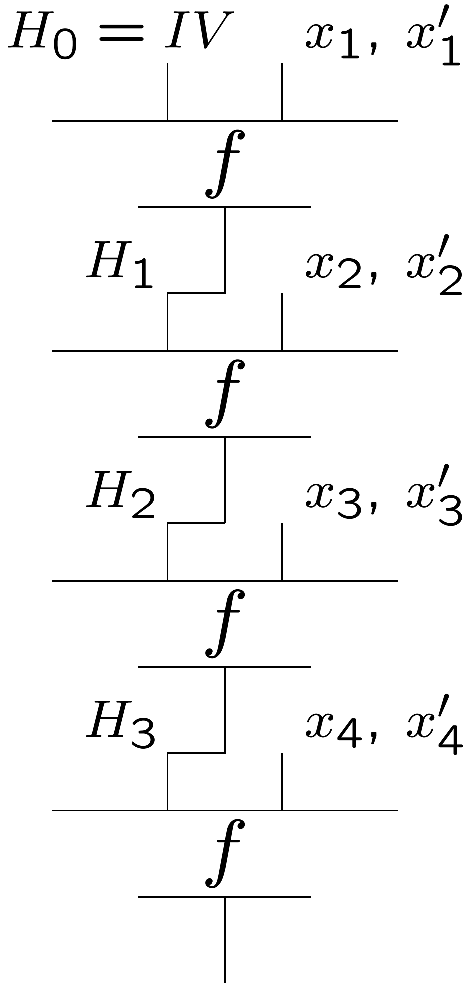
Main observation: finding multiple collisions for an iterated hash function is not much harder than finding a single collision.

Multicollisions by Joux

for H_0, collision for block 1: x_1, x'_1
for H_1, collision for block 2: x_2, x'_2
for H_2, collision for block 3: x_3, x'_3
for H_3, collision for block 4: x_4, x'_4

now we have a 16-fold multicollision for h

$$h(x_1||x_2||x_3||x_4)$$
$$= h(x'_1||x_2||x_3||x_4)$$
$$= \dots$$
$$= h(x'_1||x'_2||x'_3||x_4)$$
$$= h(x'_1||x'_2||x'_3||x'_4)$$

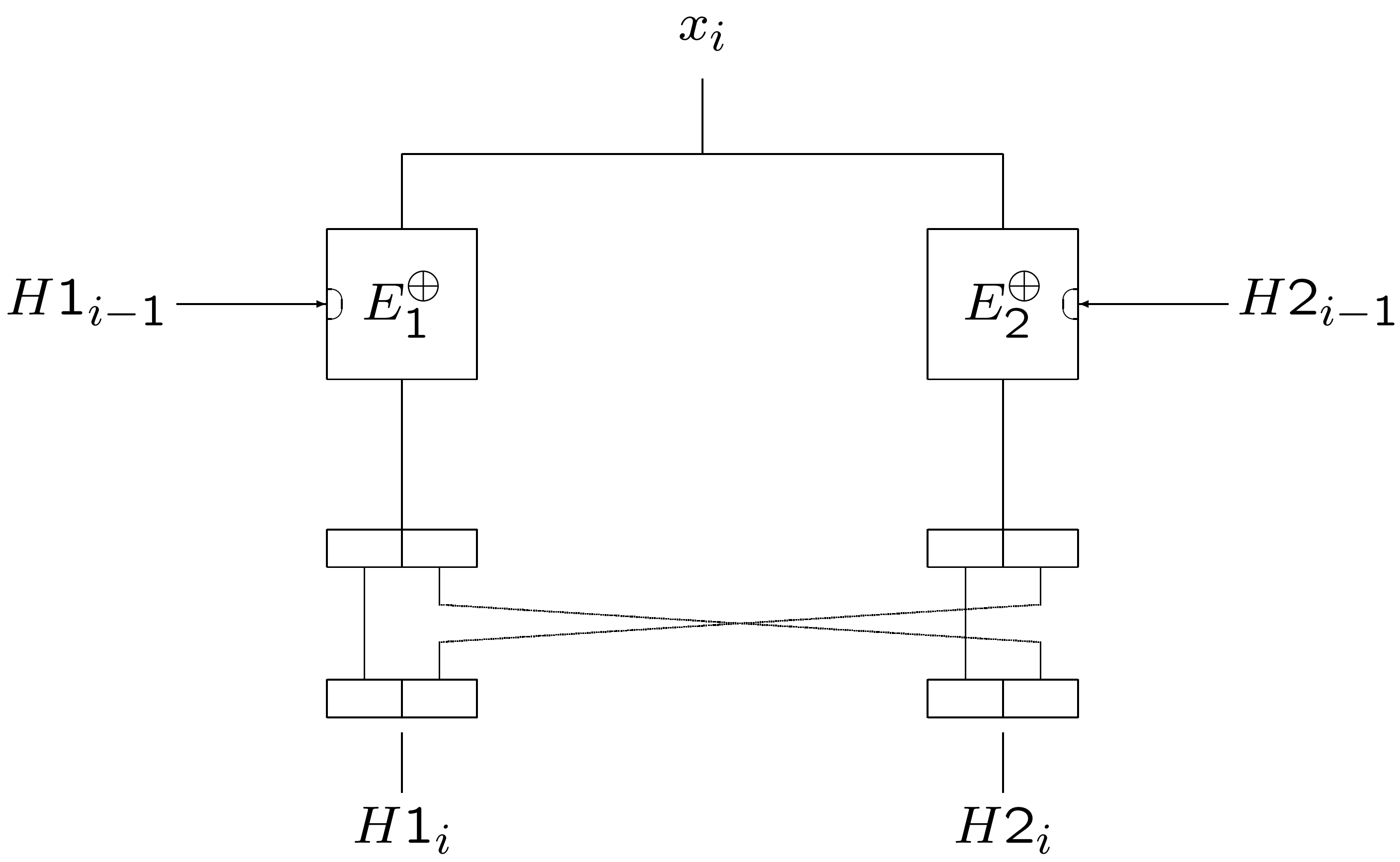


Improving Merkle-Damgård

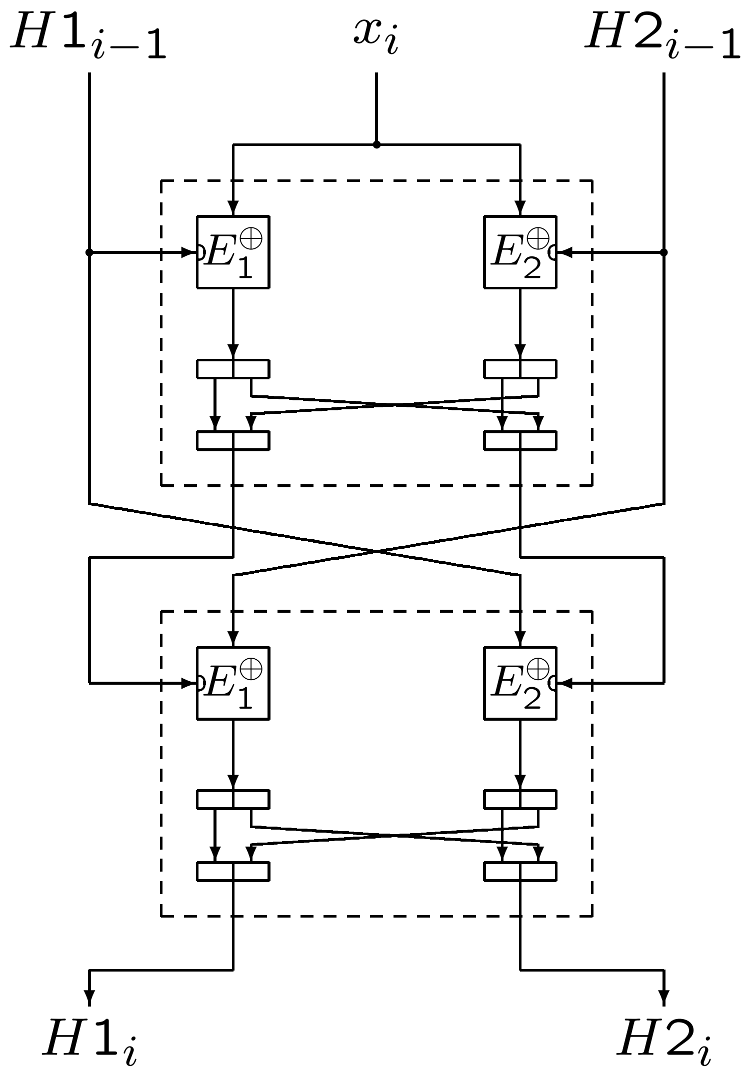
- including salting (family of functions, randomization)
- add a strong output transformation g (which includes total length and salt)
- preclude fix points: counter f f_i (Biham) or dithering (Rivest)
- multi-collisions: larger internal memory (e.g., Lucks)
- rely on principles of block cipher design, but with larger security margins
- probably not by combining smaller building blocks (à la MDC-2/MDC-4)
- can we build in parallelism and incrementality in an elegant way?

Based on Block Ciphers: MDC-2

$E_K^{\oplus}(X) = E_K(X) \oplus X$

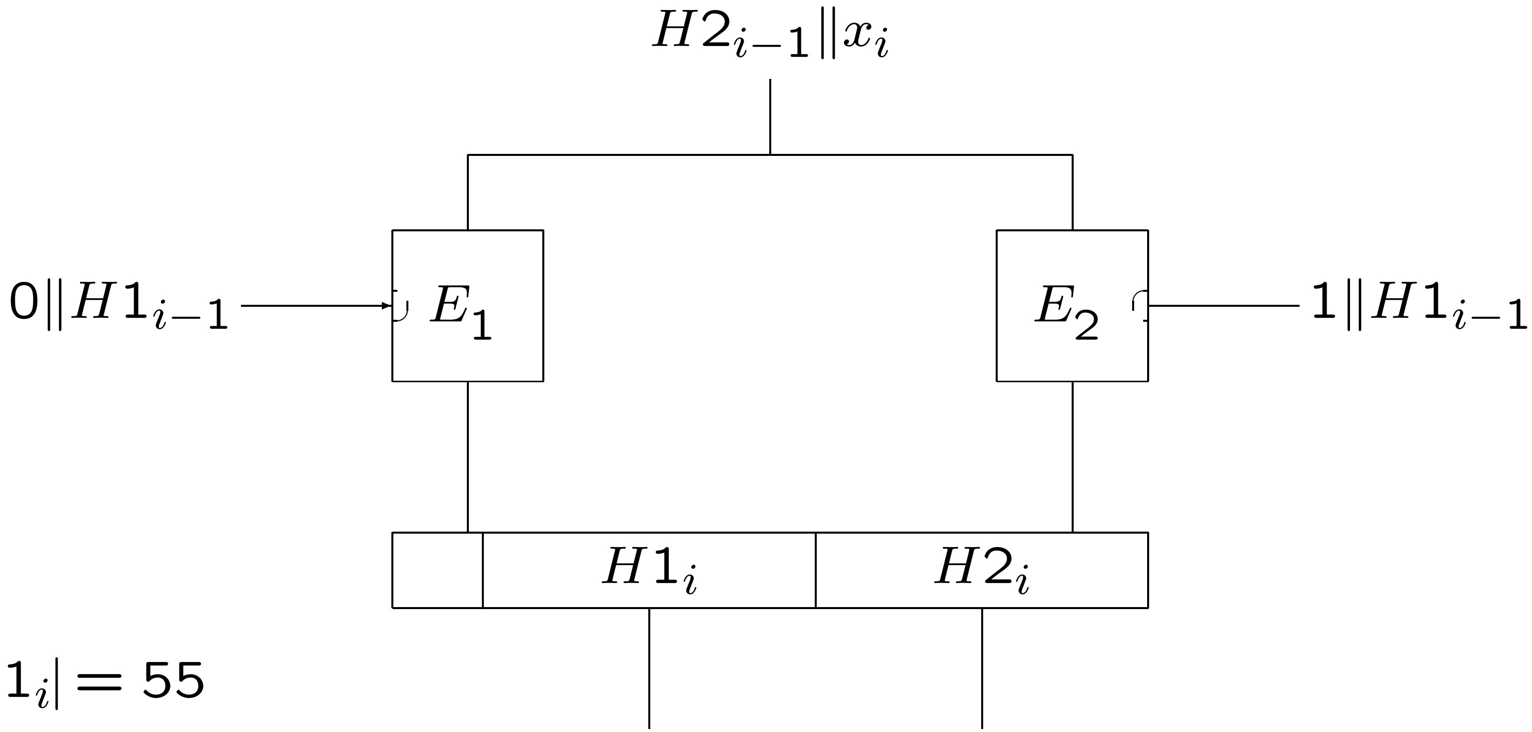


Based on Block Ciphers: MDC-4



Based on Block Ciphers: [Merkle, Crypto '89]

$E_K(X) = E_K(X) \oplus X$



$|H1_i| = 55$
$|H2_i| = 57$
$|x_i| = 7$

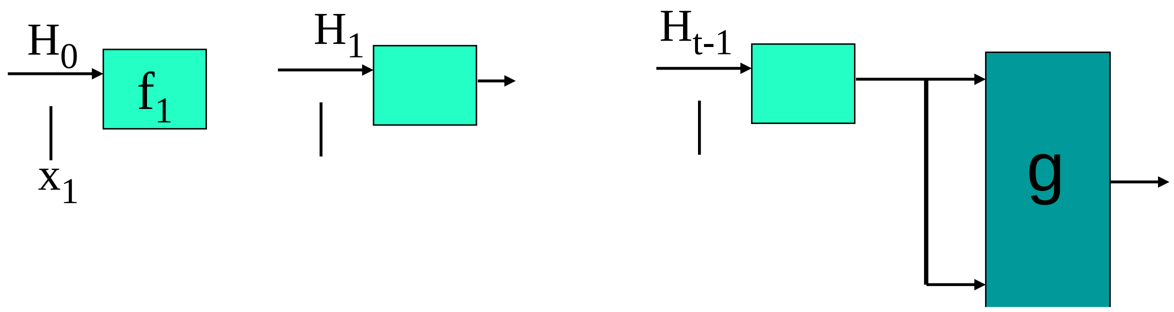
Incremental hashing

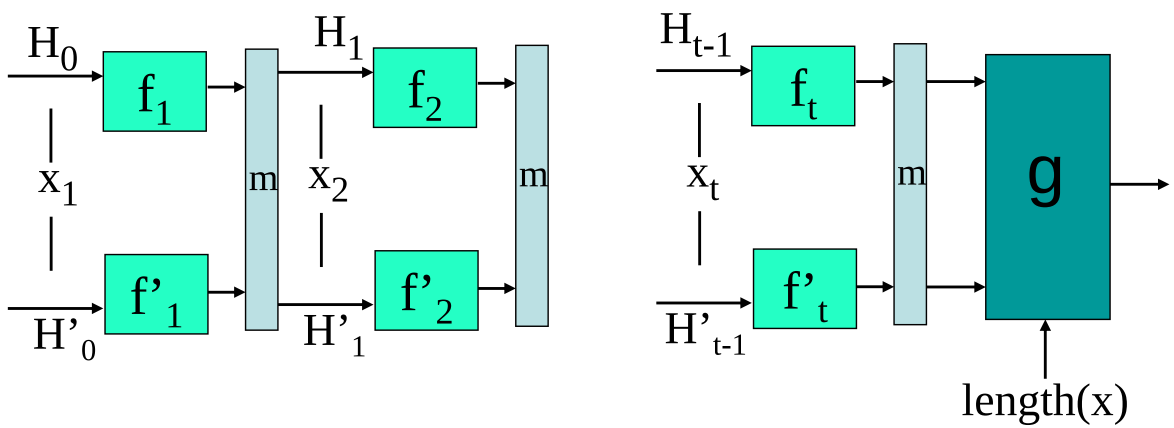
incrementality [Bellare et al. 94]
Given x and $h(x)$, if a small modification is made to x, resulting in x', one can update $h(x)$ in time proportional to the amount of modification between x and x', rather than having to recompute $h(x')$ from scratch.

[Bellare-Micciancio 97]

- hash individual blocks of message
- combine hash values with a group operation, e.g., multiplication in a group of prime order in which the discrete logarithm problem is hard

proof based on 'random oracle' assumption


H0
f1
x1
H1
Ht-1
g


H0
f1
x1
f'1
H'0
m
H1
f2
x2
f'2
H'1
m
Ht-1
ft
xt
f't
H't-1
m
g
length(x)

Pseudo-random functions?

joint work with Jongsung Kim

Key question: where to put the key?

If keyed through message input: block ciphers
best known attack: related-key boomerang distinguisher

hash function	rounds	data complexity
Haval-4 (128)	96 (full)	$2^{11.6}$ RK-CP + 2^{6} RK-ACC
MD4 (48)	48 (full)	2^{6} RK-CP + 2^{6} RK-ACC
MD5 (64)	64 (full)	$2^{13.6}$ RK-CP + $2^{11.6}$ RK-ACC
SHA-1 (80)	59 (red.)	$2^{70.3}$ RK-CP + $2^{68.3}$ RK-ACC

Distinguishers for HMAC

keyed through IV:

$$\textbf{HMAC} \quad h((K \oplus p_2) \parallel h((K \oplus p_1) \parallel x))$$

For short messages with compression function f_K:

$$\textbf{HMAC} \quad f_{K_2}(f_{K_1}(x))$$

hash function	f_{K_2}	f_{K_1}	data complexity
Haval-3 (96)	96 (full)	96 (full)	$2^{245.5}$ CP
Haval-4 (128)	128 (full)	99 (red.)	$2^{257.5}$ CP
MD4 (48)	48 (full)	35 (red.)	$2^{108.5}$ CP
MD5 (64)	64 (full)	22 (red.)	$2^{118.5}$ CP
SHA-1 (80)	80 (full)	23 (red.)	$2^{146.5}$ CP

Concluding Remarks

- we understand very little about the security of hash functions
- designers have been too optimistic (over and over again. . .)
- do we need a 'small' collision resistant compression function?
- how do we design a collision resistant compression function?
- more work should be done on other security properties:
  (2nd) preimage resistance, partial preimage resistance,
  pseudo-randomness, security with iterated applications,. . .

Design Principles for Iterated Hash Functions Revisited

Bart Preneel
Dept. Electrical Engineering ESAT-COSIC,
K.U.Leuven, Belgium
Bart.Preneel(at)esat.kuleuven.be

October 15, 2005

In this talk we review the design principles for iterated hash functions developed in the last two decades. We start by revisiting the definitions and requirements for hash functions; we focus in particular on the relation between the definitions and on the non-standard requirements such as partial preimage resistance, pseudo-randomness, and the issues related to parameterisation.

Next we discuss the relation between the security of the compression functions and that of the iterated hash functions based on them. In particular, we revisit the results of Merkle-Damgård (collision resistance) and Lai-Massey (preimage resistance). We also present the recent attacks by Felten *et al.,* Kelsey *et al.,* and Joux on iterated hash functions. We summarize ways to avoid these weaknesses and conclude by touching on parallelism and incrementality.

As a more specific result, we intend to briefly discuss the conclusions of our work on the applicability of the currently-known cryptanalytic techniques to HMAC constructions based functions of the MD4 and SHA family.

(This talk can be seen in part as an update of Design Principles for Dedicated Hash Functions. Fast Software Encryption 1993, LNCS 809, pp. 71-82).