Hash Functions: Past, Present and Future

Bart Preneel

Katholieke Universiteit Leuven

`bartDOTpreneel(AT)esatDOTkuleuvenDOTbe`

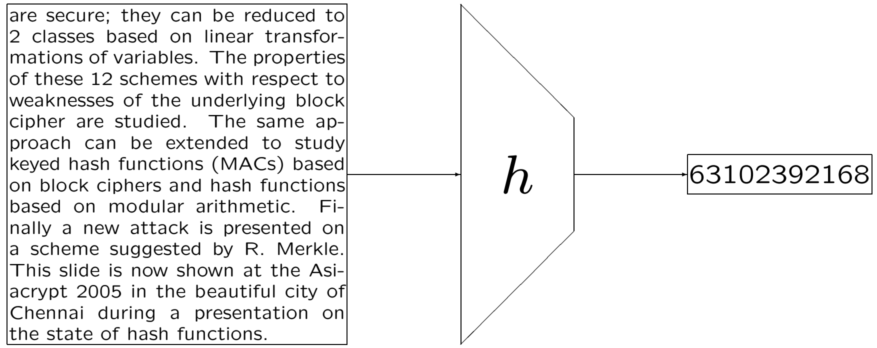
`http://homes.esat.kuleuven.be/~preneel`

5 December 2005

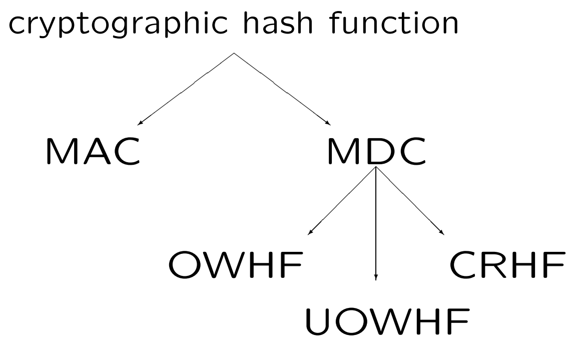
Outline

- Definitions
- Applications
- General Attacks
- Constructions
- Custom Designed Hash Functions
- Hash Functions Based on Block Ciphers
- Hash Functions Based on Algebraic Operations
- Pseudo-randomness
- Conclusions

Hash functions (1)

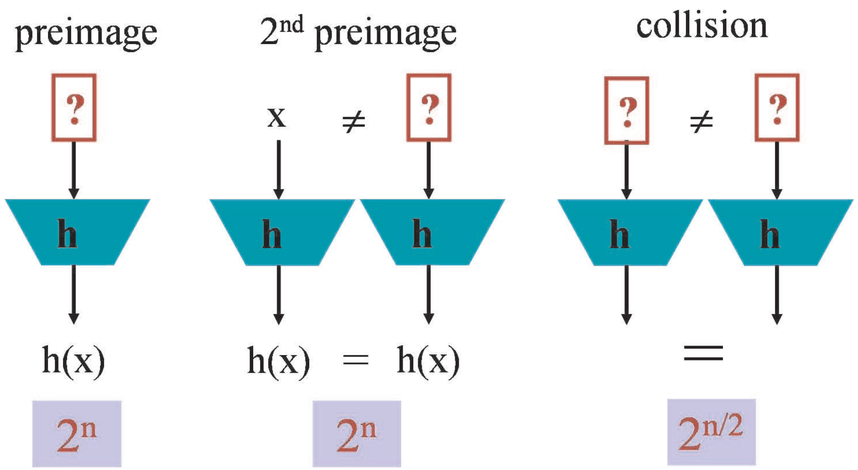


Hash functions (2)



This talk: only MDCs (Manipulation Detection Codes), which are often called 'hash functions'

Informal definitions (1)



Informal definitions (2)

- 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 (3)

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

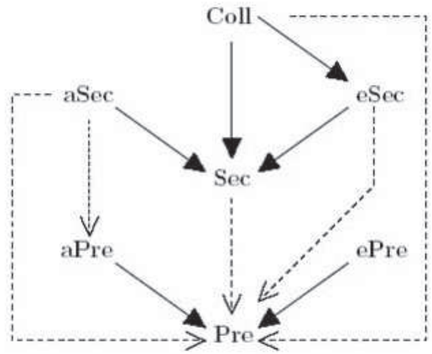


Figure 1: Summary of the relationships among seven notions of hash-function security. Solid arrows represent conventional implications, dotted arrows represent provisional implications (their strength depends on the relative size of the domain and range), and the lack of an arrow represents a separation.

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,. . .
$\leadsto$ 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 $h_K(x') = h_K(x)$

corresponds to eSec

only suitable if signer is trusted not to cheat!

Generic Attacks (1)

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

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

$\leadsto n \geq 80 \ldots 128$
avoid simultaneous attack on all targets:
parameterize ('tweak'/'salt'/'spice') 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 $r = \sqrt{2^n} = 2^{n/2}$

$\leadsto n \geq 160 \ldots 256$

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

the average effort to find a (second) preimage for one out of 2^t targets equals 2^{n-t} (and for $t = n/2$ this is $2^{n/2}$);
but if t is large, storage and search costs will be dominant

if one has to find (second) preimages for many targets, one can use a time-memory trade-off [Hellman80]:

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

[Wiener02] If $\Theta(2^{3n/5})$ targets are attacked, the full cost per (2nd) 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

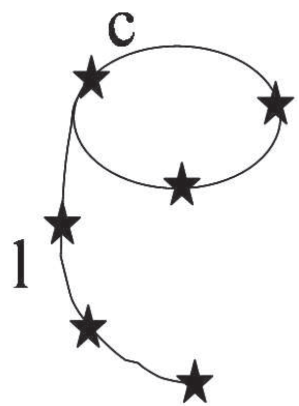
Efficient implementations of the birthday attack

- very little memory: cycle finding algorithms
- 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 [van Oorschot-Wiener]

- $n = 128$: 100 K$ for 1 month
- $n = 160$: 500 M$ for 1 year

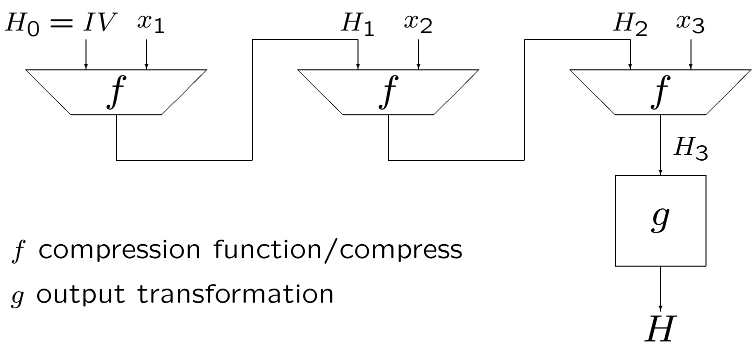


Generic Attacks (4)

attacker	invest-ment	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

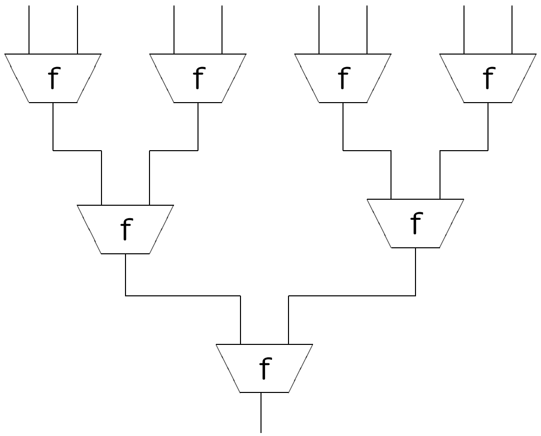
[Lai-Massey 92]
Assume that the padding contains the length of the input string, and that the message X (without padding) contains at least two blocks. Then finding a second preimage for h with a fixed IV requires 2^n operations iff finding a second preimage for f with arbitrarily chosen H_{i-1} requires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

BUT:

- this theorem is not quite right (see below)
- 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)

Advantage of strong compression function f: tree construction.



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}.

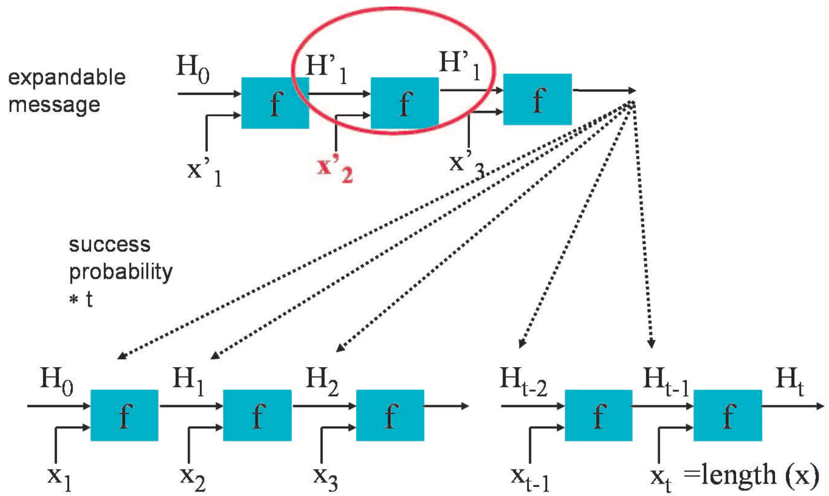
New: if one hashes 2^t message **blocks** with an iterated hash function, the effort to find a second preimage is only

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

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.

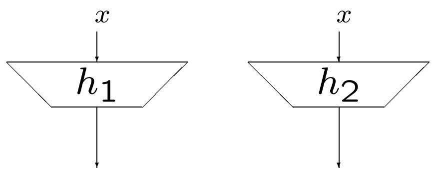
Defeating Merkle-Damgård for (2nd) preimages



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

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

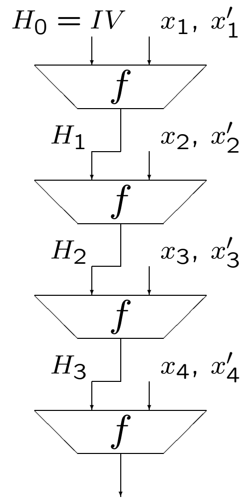
If either of the functions is weak, the attacks may work better

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



Defeating commitment protocol: herding

protocol: publish $h(x)$, reveal x at later date

herding attack [Kelsey,Kohno05]
find second preimage $x' = z||y||x$ with z and y selected in 2020

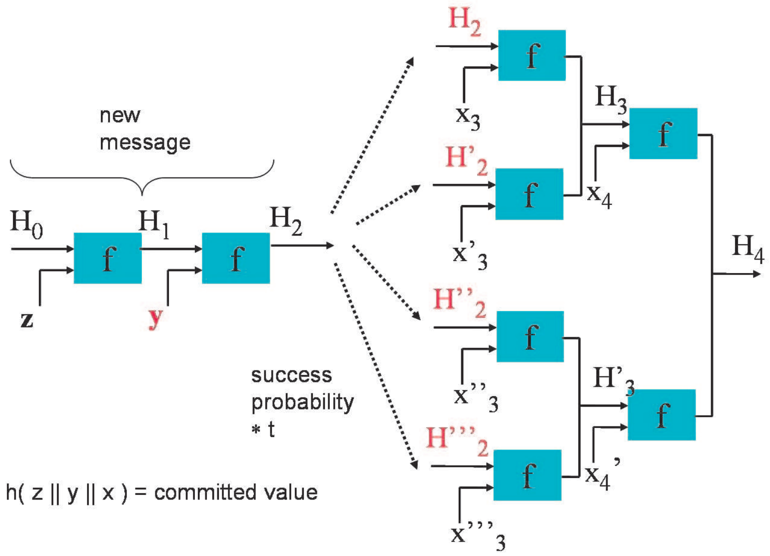
approach: generate collision tree of 2^t values H_{j-1} and x_j hashing to the same value (cost $(2 \cdot 2^{t/2} \cdot 2^{n/2})$)

z = result of all India cricket games between 2010 and 2020

try random strings y until $h(z||y) = H_{j-1}$ for some j (cost 2^{n-t})
then $h(z||y||x_j) = h(x)$

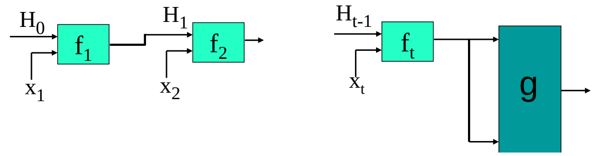
Example: $n = 128$, $t = 42$:
precomputation 2^{86}, inversion 2^{86}, storage about 100 Terabyte

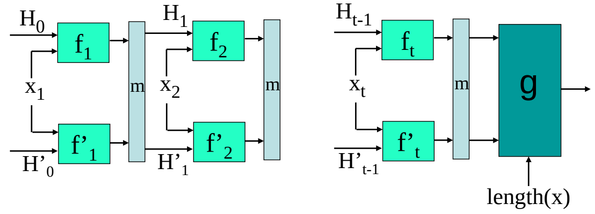
Defeating commitment protocol: herding (2)



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 $f \rightsquigarrow f_i$ (Biham) or dithering (Rivest)
- multi-collisions, herding: avoid breakdown at $2^{n/2}$ with larger internal memory (e.g., RIPEMD, [Lucks05])
- 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?





Construction (7): UOWH

[Naor-Yung 89]
Composition lemma for UOWH

[Bellare-Rogaway 97]

- XOR linear scheme
- basic tree hash
- exor tree hash

efficiency improvements
[Shoup 00], [Sarkar04], [Lee, Chang, Lee, Sung, Nandi 04]

easier to design

Custom Designed Hash Functions (1)

shortlist:

- MD4-family: MD4, extended MD4, MD5, SHA, SHA-1, RIPEMD-160, SHA-xxx
- MD2 (8 to 8-bit table)
- Snefru (8 to 32-bit tables, 8 passes)
- N-hash (FEAL-based)
- FFT-hash III (FFT transform)
- Subhash (hardware)
- Tiger (64-bit architecture)
- Panama (VLIW processor) – broken [2001]
- Whirlpool
- FORK-256
- DHA-256
- . . . and many broken proposals . . .

MD4

designed by Rivest in 1990
3 rounds

- collisions for 2 rounds [Merkle90, denBoerBosselaers91]
- near collision [Vaudenay94]
- collisions for full MD4 in 2^{20} steps [Dobbertin96]
- (second) preimage for 2 rounds [Dobbertin97]
- collisions for full MD4 by hand [Wang+04]
- practical preimage attack for 1 in 2^{56} messages [Wang+05]

abandoned since 1993

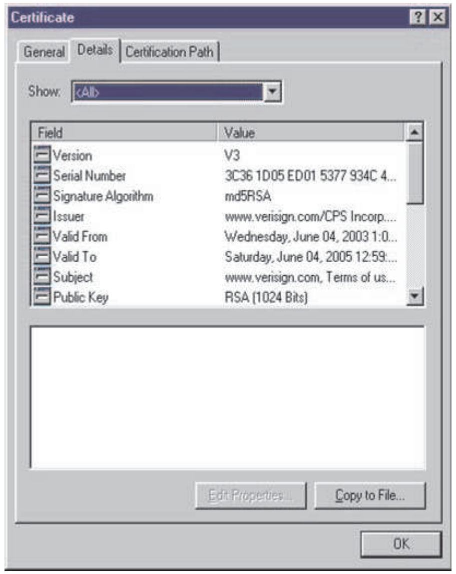
MD5

designed by Rivest in 1991
4 rounds

- collisions for compression function f [denBoer-Bosselaers93] – ΔIV
- real collisions for compression function f [Dobbertin96] - wrong IV
- **real collisions in 2^{39} steps [Wang+04] 15 minutes!!**

Collisions for MD5

- Advice (RIPE since 1992, RSA since 1996): stop using MD5
- largely ignored by industry (click on any cert . . .)
- collisions for MD5 are within range of a brute force attack anyway (2^{64})
- attack is being improved



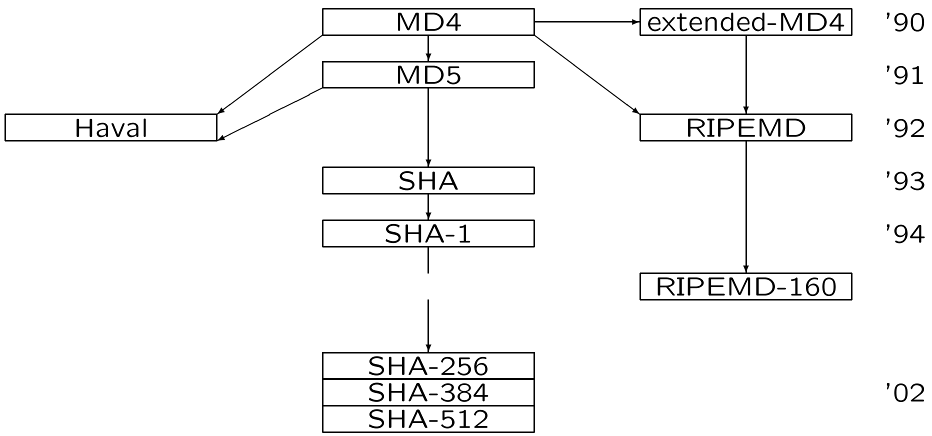
SHA-1

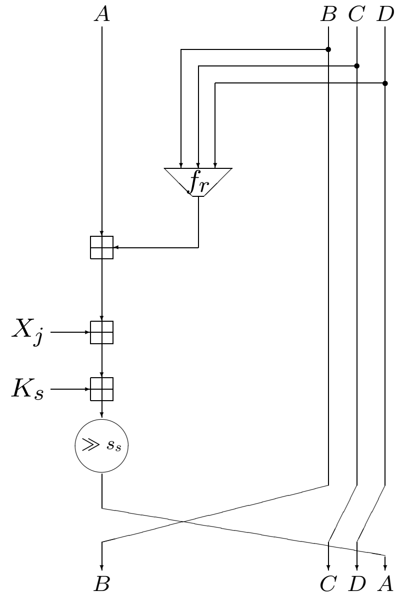
SHA designed by NIST (NSA) in 1993
5 'rounds'
redesign after 2 years (95) to SHA-1

- Collisions for SHA(-0) in 2^{51} [Joux+04]
- Collisions for SHA(-0) in 2^{39} [Wang+05]

- Collisions for SHA-1 in 2^{63} [Wang+05]

The MDx-family: pedigree





Step for MD4

→ updates one word of chaining variable

→ based on
- Boolean function f_r
- message word X_j
- additive constant K_s
- rotation amounts s_s

→ operations on 32-bit words
- addition mod 2^{32}
- fixed rotations ($\ggg 11, \ggg 7$)
- bitwise AND, XOR (f_r)

MDx-family: properties

Algorithm	n	rounds	steps	word	block	endianness
MD4	128	3	48	32	512	Little
ext-MD4	256	2×3	96	32	512	Little
MD5	128	4	64	32	512	Little
SHA-1 (SHA)	160	4	$80 + 64$	32	512	Big
RIPEMD	128	2×3	96	32	512	Little
RIPEMD-128	128	2×4	128	32	512	Little
RIPEMD-160	160	2×5	160	32	512	Little
SHA-256	256	–	$64 + 64$	32	512	Big
SHA-384	384	–	$80 + 64$	64	1024	Big
SHA-512	512	–	$80 + 64$	64	1024	Big
HAVAL	128- -256	3,4,5	96, 128, 160	32	1024	Little

MDx-family: history of attacks

- collisions $MD4_{12}$ [Merkle '90]
- collisions $MD4_{23}$ [den Boer-Bosselaers '91]
- pseudo-collisions MD5 [den Boer-Bosselaers '93]
- collisions $MD4_{12}$, near collisions MD4 [Vaudenay '94]
- unidentified problem with SHA [NSA '94]
- [Dobbertin '95-'97]:
  - collisions $RIPEMD_{23}$, $RIPEMD_{12}$
  - collisions MD4 (even with structure)
  - collisions for ext-MD4 compress with random IV
  - collisions for MD5 compress with random IV
  - preimage for $MD4_{12}$
- collisions for SHA in 2^{61} [Chabaud-Joux '98]
- collisions for 2 rounds of RIPEMD [Debaert-Gilbert '01]

MD4-family: history of attacks (2)

- Collision attacks on reduced 2-round versions of HAVAL [Kasselman-Penzhorn 00] [Park-Sung-Chee-Lim 02] [Her-Sakurai-Kim 03]
- Saarinen 2003: slide attacks on SHA and MD5
- simplified SHA-xxx ($+ \rightarrow \oplus$, symmetric constants) has symmetry properties [Gilbert-Handschuh 03]
- Collisions for Haval [Biryukov, Van Rompay, Preneel 02]
- Collisions for SHA(-0) in 2^{50} [Joux+ 04]
- Collisions for MD4 (by hand), MD5, and RIPEMD [Wang-Feng-Lai-Yu 04]
- Attack on 53 out of 80 rounds of SHA-1 [Biham-Chen 04]
- Attack on 53 out of 80 rounds of SHA-1 [Rijmen-Oswald 04]
- 2^{39} attack on SHA(-0) [Wang-Yu-Yin 05]
- 2^{63} attack on SHA-1 [Wang-Yin-Yu 05]
- variant of second preimage attack on MD4

MD4-family: SHA-1 & RIPEMD-160

common features:

- 160-bit result
- extra rotate on one of the message words ('MSB problem')
- both in ISO/IEC 10118-3:1998 (also RIPEMD-128)

RIPEMD-160:

- two independent parallel halves, which are made as different as possible (order of message words, Boolean functions, constants, rotations)
- 5 rounds

MD4-family: SHA-1 & RIPEMD-160

SHA-1: (FIPS 180)

- no repetition of message words, but encoding ($j \geq 16$):

$$X[j] := (X[j-3] \oplus X[j-8] \oplus X[j-14] \oplus X[j-16]) <<< 1;$$

$=$ systematic linear code [$n = 2560, k = 512, d < 86$]

compared to SHA:

- bitwise shortened cyclic code [$n = 80, k = 16, d = 23$]

$$X[j] := X[j-2] \oplus X[j-3] \oplus X[j-7] \oplus X[j-16];$$

MD4-family: SHA-xxx

SHA-224, SHA-256, SHA-384, SHA-512

- message processing:

  $$X[j] := \sigma_1(X[j-2]) \oplus X[j-7] \oplus \sigma_0(X[j-15]) \oplus X[j-16];$$

  with σ_i = sum of 2 rotated and 1 shifted value of the same variable
- more complex round functions: each step has multiplexer, majority, Σ-function (sum of 3 rotated versions of input)
- 64 different constants
- SHA-384, SHA-512: 64-bit words
- SHA-384 is obtained by truncating result of SHA-512

Whirlpool [Rijmen-Barreto00]

- based on a Rijndael-like block cipher with a 512-bit block and a 512-bit key (state: 8×8 matrix)

  $$E_K(X) \oplus X \oplus K$$

- key schedule (message input): same rounds as block cipher with constant key
- S-box is *not* inverse, but built of four 4-bit S-boxes
- best known attack: 6 rounds out of 10

Attack ideas by Wang et al.

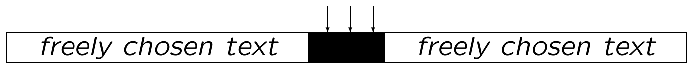
Very clever combination of new and known techniques:

- differential attack but modular differences (mod 2^{32}) rather than $\oplus$ [Berson'92]
- find new differentials with control of carry bits
- message modification: characteristic satisfied in first steps [Biham92,Rijmen-Preneel93]
- advanced message modification: characteristic satisfied further below
- multi-block technique [Preneel92]

Impact of recent attacks

collisions for MD5, SHA(-0), SHA-1

- two messages differ in a few bits in 1 to 3 512-bit input blocks
- limited control over message bits in these blocks
- but arbitrary choice of bits before and after them



what is achievable today?

- 2 colliding executables
- 2 colliding postscript documents [Lucks-Daum 05] or pictures
- 2 colliding RSA public keys thus with colliding X.509 certificates [Lenstra-Wang-de Weger 04]
- **2 arbitrary colliding files (no constraints) for 100 K$**

Impact of recent attacks

collisions:

- none for signatures computed before attacks were public (1 August 2004)
- none for certificates if public keys are generated at random in a controlled environment
- **substantial** for signatures after 1 August 2004 (cf. traffic tickets in Australia)

second preimages:

- security degrades with number of applications
- general attacks based on fixed points [Kelsey, Schneier 05]
- specific attacks exist for MD2/MD4
- for MD5/SHA-1: not a threat for current applications

Practical solutions

- RIPEMD-160 seems more secure than SHA-1 **;-)**
- message precoding [Szydlo-Yin 05]
- small patches to SHA-1 [Jutla-Patthak 05]
- use more recent standards (but 40-80 cycles/byte)
- use older schemes: Tiger, Snefru with more rounds
- start from scratch: new NIST competition

Outline

- *Definitions*
- *Applications*
- *General Attacks*
- *Constructions*
- *Custom Designed Hash Functions*
- Hash Functions Based on Block Ciphers
- Hash Functions Based on Algebraic Operations
- Pseudo-randomness
- Conclusions

Based on Block Ciphers (1)

Why:

- trust
- reduce design, evaluation, and implementation effort
- compact implementation

Why not:

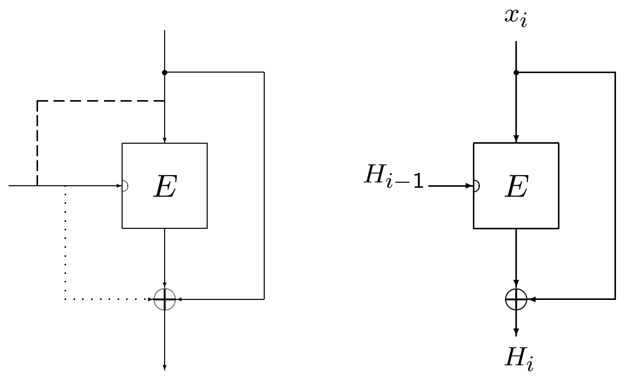
- slow (key schedule)
- export restrictions
- weaknesses which are not relevant to encryption

rate = # blocks hashed per encryption

Based on Block Ciphers (2)

single block length hash functions:

- 12 'secure' schemes of rate 1; one in ISO/IEC 10118-1
- collision $2^{n/2}$, (2nd) preimage 2^n



security proof: [Winternitz '82] [Black, Rogaway, Shrimpton '02]

Based on Block Ciphers (3)

double block length hash functions with rate 1:

$$\begin{aligned} H_i^1 &= E_{A_i^1}(B_i^1) \oplus C_i^1 \\ H_i^2 &= E_{A_i^2}(B_i^2) \oplus C_i^2 \end{aligned}$$

- A_i^1, B_i^1, C_i^1 binary linear combinations of H_{i-1}^1, H_{i-1}^2, x_i^1, and x_i^2
- A_i^2, B_i^2, C_i^2 are binary linear combinations of H_{i-1}^1, H_{i-1}^2, x_i^1, x_i^2, and H_i^1.

goal: collision 2^m, (2nd) preimage 2^{2m}
BUT:

- [Hohl et al. 94]: compression function has at most security level of single length hash function
- [Knudsen-Preneel-Lai 96]: collisions in time $2^{3m/4}$ or $2^{m/2}$

Based on Block Ciphers (4)

rate < 1:

- [Brachtl et al. (IBM) 89] MDC-2: rate 1/2
  ISO/IEC 10118-2
- [Brachtl et al. (IBM) 89] MDC-4: rate 1/4

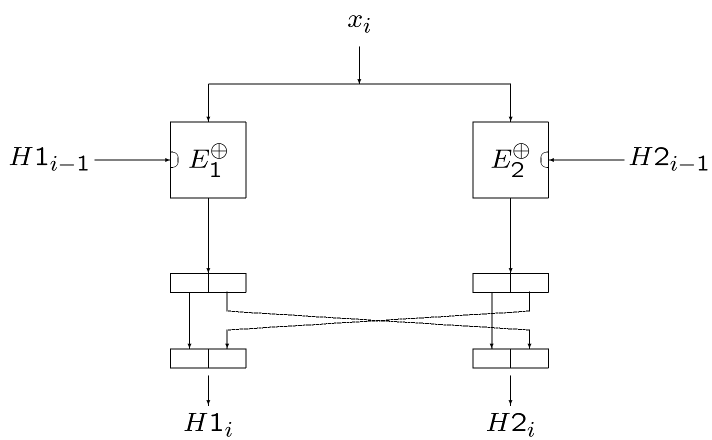
security for DES:

	rate	collision	preimage	coll (f)	preimage (f)
MDC-2	1/2	2^{55}	2^{83}	2^{28}	2^{54}
MDC-4	1/4	2^{56}	2^{109}	2^{41}	2^{90}

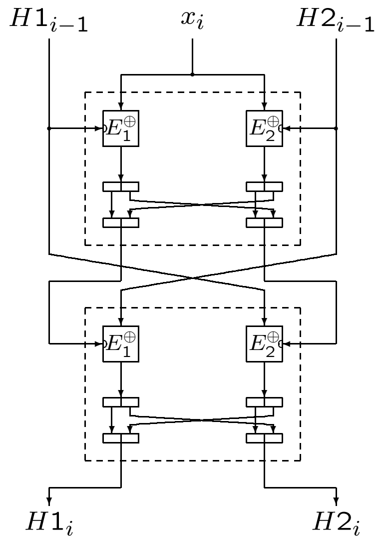
problem: proof of security?

Based on Block Ciphers (5): MDC-2

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



Based on Block Ciphers (6): MDC-4



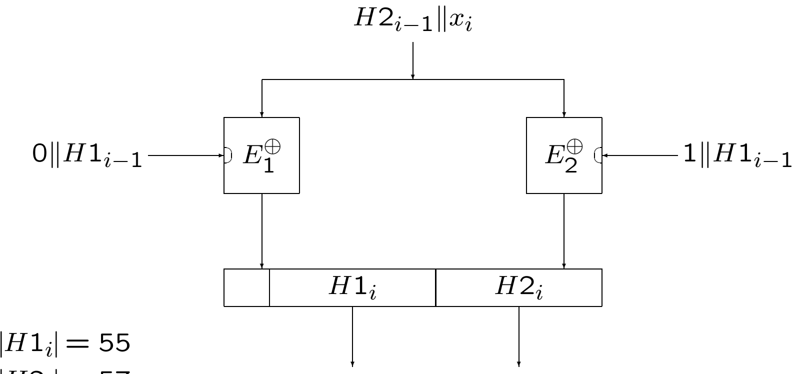
Based on Block Ciphers (7)

double block length hash functions:
(with collision resistant compression function)

- [Merkle '89]
  – rate between 1/18. . . 1/4, inconvenient block sizes
  – security proof (2^{56}) based on black box model of DES
- [Knudsen-Preneel '96-'97]:
  – rate 1/3 with 9 parallel encryptions
  – security proof (2^{72}) based on black box model of DES
  – assumption in security proof needs small correction
- [Nandi-Lee-Sakurai-Lee '05]
  – security proof collisions $2^{2n/3}$ for rate 1/3
  – but near-preimages and near-collisions [Knudsen-Muller '05]
- [Aiello-Haber-Venkatesan '98]:
  – very fast because of modified key schedule
  – security proof for several assumptions on DES
- research topic – also double key length [Lai-Massey 92] . . .

Based on Block Ciphers (8): Merkle

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



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

Based on Algebraic Structures (1)

Why:

- sometimes one can prove security reductions
- compact implementation
- fast (knapsack-type problems)

Why not:

- mathematical structure can be exploited
- slow (modular exponentiation)
- vulnerable to trapdoors

Algebraic Structures (2): modular arithmetic

how to generate the RSA modulus?
answer: secure multi-party computation
[Boneh-Franklin 97], [Frankel-MacKenzie-Yung 98]

schemes with security reduction:

- [Damgård 87]: equivalent to factoring
- [Gibson 91]: discrete logarithm modulo a composite
- [Chaum et al. 91], [Brands], [Bellare et al. 94]
  discrete logarithm in a group of prime order G_p
  - prime p and t random elements α_i from G_p ($\alpha_i \neq 1$).

$$H_{t+1} = \prod_{i=1}^{t} \alpha_i^{\tilde{x}_i} \qquad \text{with } \tilde{x}_i = 1 \| x_i$$

Algebraic Structures (3): modular arithmetic

schemes with security reduction (continue):

- [Contini-Lenstra-Steinfeld 05]: VSH (Very Smooth Hash)
  - based on factoring
  - equivalent to finding modular square roots of smooth numbers
  - needs about $1/\log_2 n$ modular multiplications (mod n) per bit
  - 110/180 cycles/byte (1024/20480-bit modulus) or about 25 times slower than SHA-1
  - reduction not very tight
- [Charles-Boren-Lauter 05]: expander graphs
  - elliptic curve based construction

Algebraic Structures (4): modular arithmetic

schemes without security reduction:

- many broken proposals, including CCITT X.509 Annex D
- most promising: ISO/IEC 10118-4:1998
  MASH-1 (Modular Arithmetic Secure Hash)
  $$H_i = ((x_i \oplus H_{i-1}) \vee A)^2 \pmod{N} \quad \oplus H_{i-1}$$
  $A =$ `0xF00...00`
  x_i: 4 most significant bits in every byte equal to `1111`
  output transformation that reduces output size to at most $n/2$
  MASH-2: replace exponent 2 by $2^8 + 1$

  security for n-bit RSA modulus:
  - best known attacks: preimage in $2^{n/2}$, collision in $2^{n/4}$
  - feedforward of H_{i-1} essential

Algebraic Structures (5): knapsacks and lattices

additive knapsacks:

knapsack problem of dimensions n and $\ell(n)$:
given a set of n l-bit integers $\{a_1, a_2, \ldots, a_n\}$, and an l-bit integer S find a vector X with components x_i equal to 0 or 1 such that

$$\sum_{i=1}^{n} a_i \cdot x_i = S \bmod 2^{\ell(n)}.$$

for hashing, one needs $n > \ell(n)$.

the **good** news:

- [Impagliazzo-Naor 96]: UOWH as secure as knapsack
- [Ajtai 96], [Goldreich+ 96]: one-way and collision-resistant function if approximating the shortest vector in a lattice to polynomial factors is hard
- [Sendrier et al.]: random matrix + structured input: syndrome decoding is hard problem

Algebraic Structures (6): knapsacks and lattices

the **bad** news:

- the knapsack problem seems to be 'too easy' for realistic parameters (1000 vectors of 500 bits).
- LLL for $\ell(n) > 1.0629n$
- [Camion-Patarin 91] and [Patarin 93] for $n \gg \ell(n)$
- [Wagner-02] generalized birthday attack

Algebraic Structures (7): knapsacks and lattices

multiplicative knapsacks: [Tillich-Zémor 94]

matrix product in group $SL_2(F_{2^n})$

$$A = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} X & X+1 \\ 1 & 1 \end{pmatrix}$$

$$\pi\{0,1\} \rightarrow \{A, B\}; 0 \mapsto A,\ 1 \mapsto B$$

$$h(x_1 x_2 \ldots x_n) = \pi(x_1) \cdot \pi(x_2) \ldots \pi(x_n)$$

evaluation:

\+ proof that two colliding messages have 'large' Hamming distance

\+ parallelism

– new attacks using algebraic structure

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

Pseudo-random functions?

joint work with Jongsung Kim and Alex Biryukov

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)	$2^{228.6}$ CP
Haval-4 (128)	128 (full)	102 (red.)	$2^{253.9}$ CP
MD4 (48)	48 (full)	48 (full)	2^{74} CP
MD5 (64)	64 (full)	33 (red.)	$2^{126.1}$ CP
SHA(-0) (80)	80 (full)	80 (full)	2^{109} CP
SHA-1 (80)	80 (full)	43 (red.)	$2^{159.9}$ CP

Concluding Remarks

- we understand very little about the security of hash functions
- designers have been too optimistic (over and over again...)
- block ciphers, MAC algorithms, stream ciphers get faster, but hash functions now 4-5 times slower
- 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,...

Read more?

- ECRYPT hash function workshop `http://www.ecrypt.eu.org` and `http://www.impan.gov.pl/BC/05Hash.html`
- NIST hash function workshop `http://www.csrc.nist.gov/pki/HashWorkshop/index.html`
- My 1993 PhD thesis `http://homes.esat.kuleuven.be/~preneel`
- Overview paper from 1998 (LNCS 1528) `http://www.cosic.esat.kuleuven.be/publications/article-346.pdf`