ECRYPT II

The cryptographic hash function crisis and the SHA-3 competition

Bart Preneel
Katholieke Universiteit Leuven - COSIC
Firstname.lastname@esat.kuleuven.be

Cryptography ≠ security

- crypto is only a tiny piece of the security puzzle
  - but an important one
- most systems break elsewhere
  - incorrect requirements or specifications
  - implementation errors
  - application level
  - social engineering

Information processing

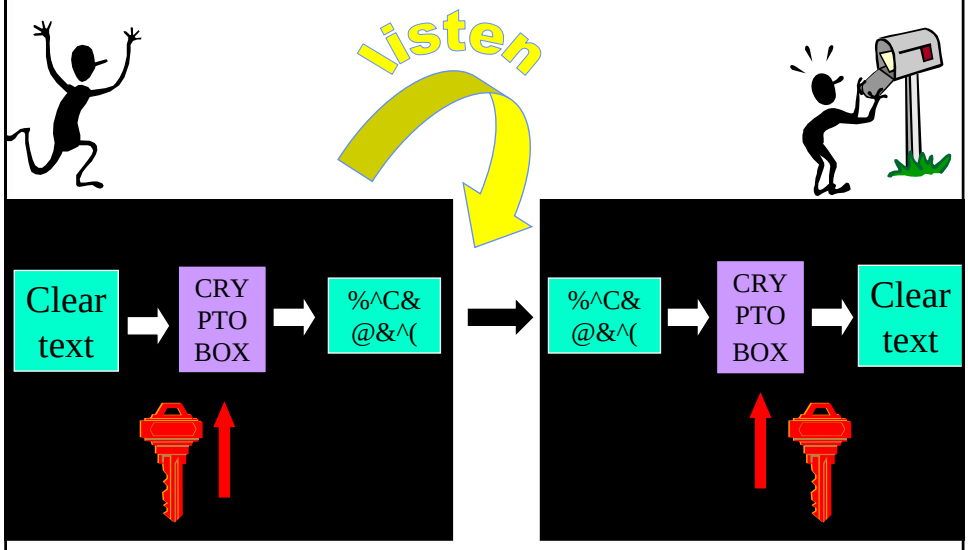
Everything is always connected everywhere

Continuum between software and hardware

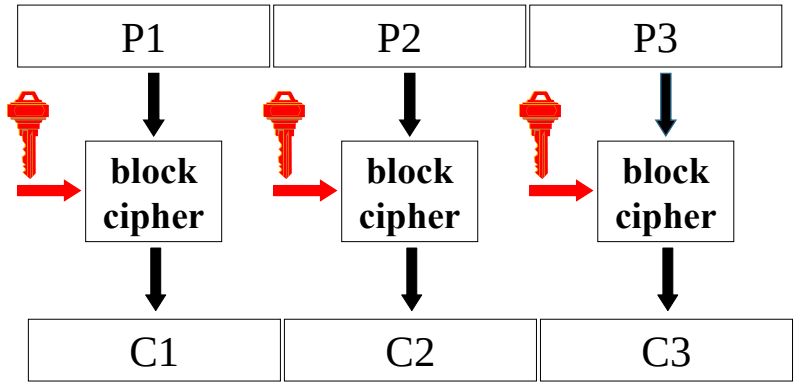
ASIC (microcode) – FPGA – fully programmable processor

Cryptography everywhere

Cryptology: basic principles



Block ciphers



- process blocks of: 64…128 bits
- memoryless
- repeat simple operation (round) many times

Block ciphers

64-bit block

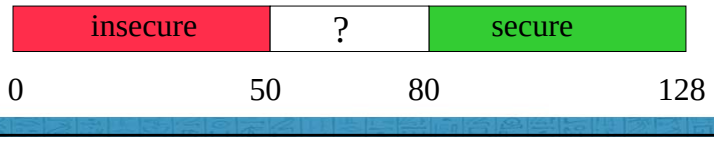
DES (56 bits)
3-DES (112-168)
IDEA (128)
KASUMI (128 in 3G, 64 in 2G)

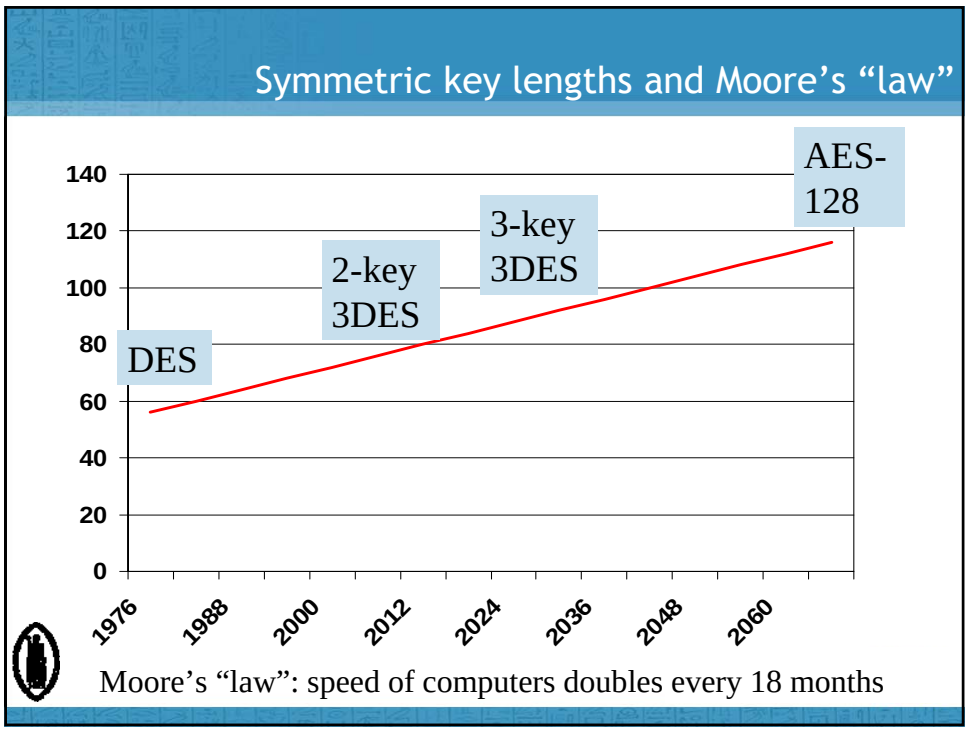
128-bit block

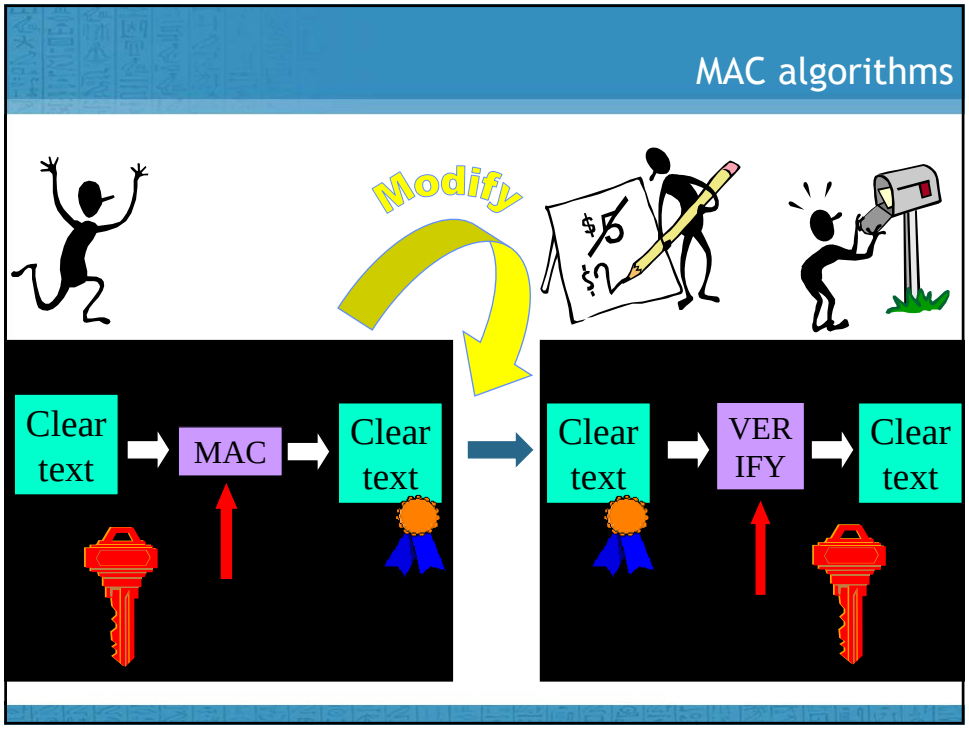
AES (128-192-256)
RC6

56 bits: 4 seconds with $5M
80 bits: 2 year with $5M
128 bits: 256 billion years with $5B

Symmetric key lengths



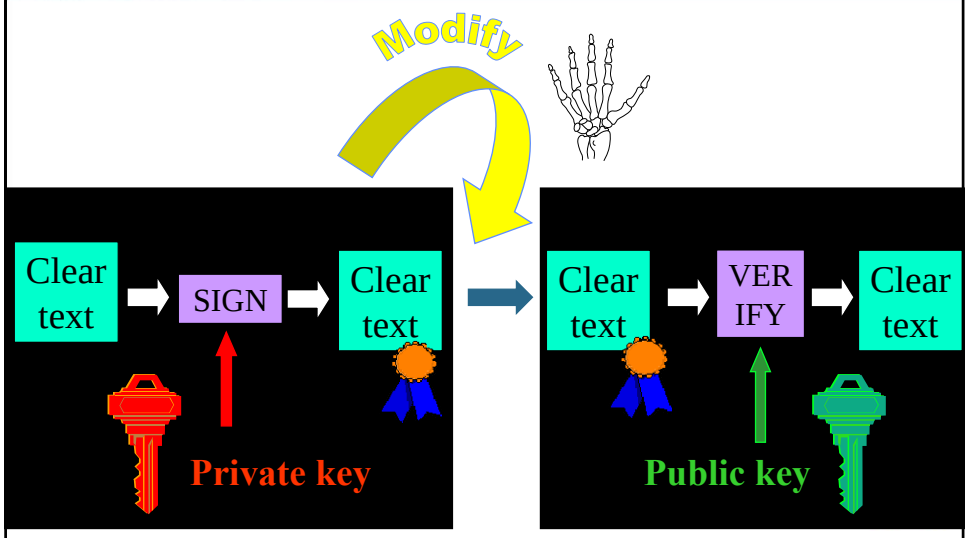

Symmetric key lengths and Moore's "law"
140
120
100
80
60
40
20
0
1976
1988
2000
2012
2024
2036
2048
2060
DES
2-key 3DES
3-key 3DES
AES-128
Moore's "law": speed of computers doubles every 18 months


MAC algorithms
Modify
Clear text
MAC
Clear text
Clear text
VER IFY
Clear text

MAC algorithms

- example schemes: CBC-MAC, HMAC
- result is 4-20 bytes
- same speed as block cipher/hash function
- requires shared secret to verify

Public key cryptology: digital signature



Digital signatures

- example schemes: RSA, DSA, ECDSA
- result is 40-256 bytes
- much slower than a MAC algorithm
- requires no shared secret to verify
- but how do I sign a document that is 1 Mbyte?

Hash functions

X.509 Annex D
MDC-2
MD2, MD4, MD5
SHA-1

→

RIPEMD-160
SHA-256
SHA-512

→

SHA-3

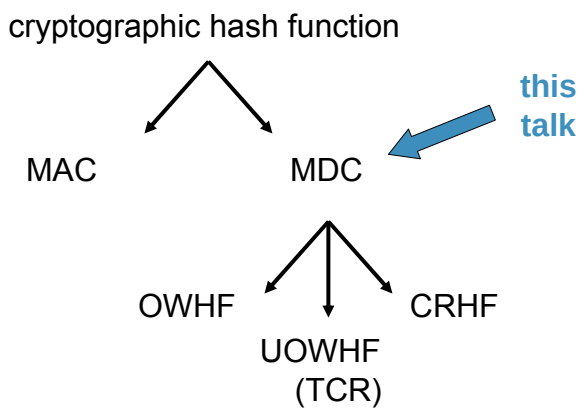
This is an input to a crypto-graphic hash function. The input is a very long string, that is reduced by the hash function to a string of fixed length. There are additional security conditions: it should be very hard to find an input hashing to a given value (a preimage) or to find two colliding inputs (a collision).

→ **h** → *1A3FD4128A198FB3CA345932*

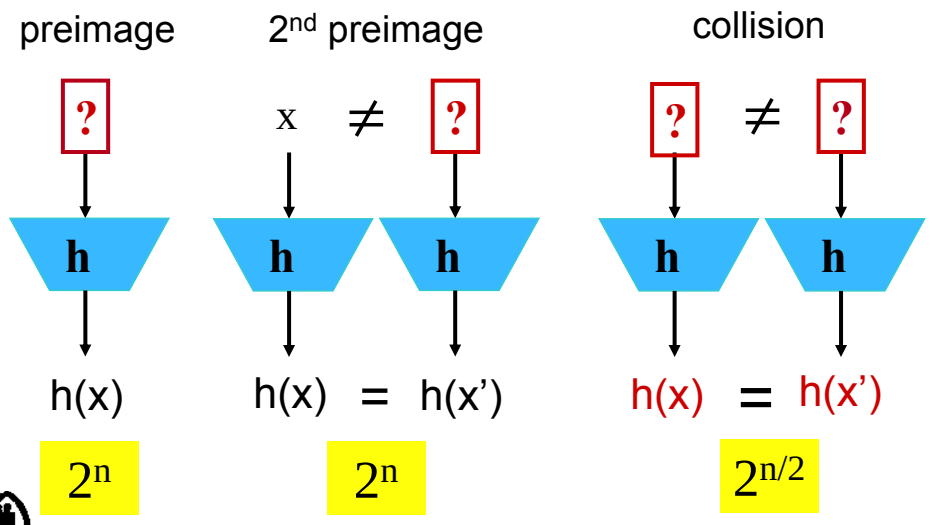
Agenda

- Definitions
- Iterations (modes)
- Compression functions
- SHA-{0,1,2}
- SHA-3 Bits and bytes

Hash function flavors



Security requirements (n-bit result)



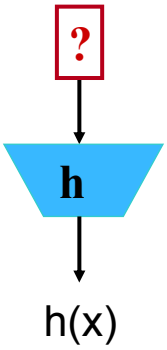
Informal definitions (1)

- no secret parameters
- input string *x* of arbitrary length ⇒ output h(x) of fixed bitlength *n*
- computation "easy"

- One Way Hash Function (OWHF)
  - preimage resistance
  - 2nd preimage resistance
- Collision Resistant Hash Function (CRHF): OWHF +
  - collision resistant

Preimage resistance

preimage

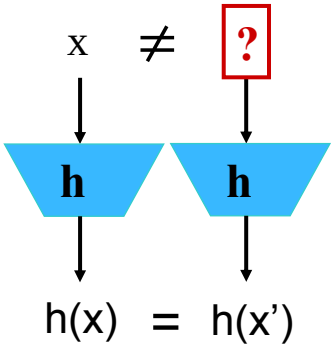


2^n

- in a password file, one does not store
  - (username, password)
- but
  - (username,hash(password))
- this is sufficient to verify a password
- an attacker with access to the password file has to find a preimage

Second preimage resistance

2nd preimage

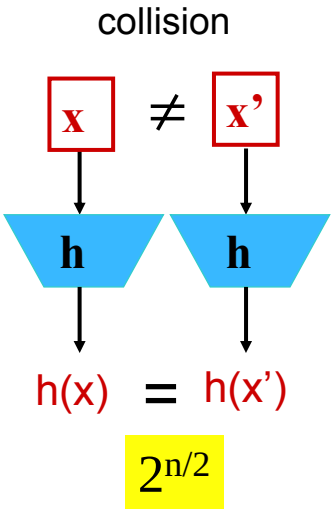


2^n

- transmit x over a fast but insecure channel
- transmit h(x) over a slow but authenticated channel (e.g., read it over the phone)
- an attacker has access to x but he can only fool the recipient if he finds a second preimage of x
- another example:
  - compute a hash of the files on a USB stick before you lend it to your friend
  - you can write down the hash value

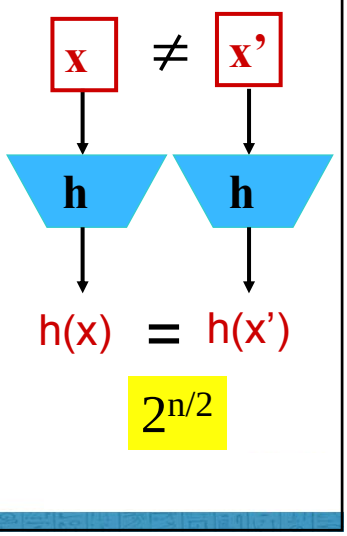
Collision resistance (1/2)

- hacker Alice prepares two versions of a software driver for the O/S company Bob
  - x is correct code
  - x' contains a backdoor that gives Alice access to the machine
- Alice submits x for inspection to Bob
- if Bob is satisfied, he digitally signs h(x) with his private key
- Alice now distributes x' to users of the O/S; these users verify the signature with Bob's public key
- this signature works for x and for x', since h(x) = h(x')!



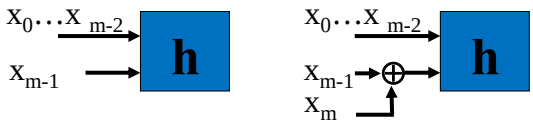
Collision resistance (2/2)

- in many cryptographic protocols, Alice wants to commit to a value x without revealing it
- Alice picks a secret random string r and sends y = h(x || r) to Bob
- in a later phase of the protocol, Alice reveals x and r to Bob and he checks that y is correct
- if Alice can find a collision, that is (x,r) and (x',r') with x' ≠ x she can cheat
- if Bob can find a preimage, he can learn x and cheat



Relation between definitions (informal!)

- 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, $\underline{h}$ is 2nd preimage resistant but not preimage resistant: $\underline{h}(x) = \begin{cases} 0 \,||\, x & \text{if } |x| \le n \\ 1 \,||\, h(X) & \text{otherwise} \end{cases}$
- collision resistant $\Rightarrow$ 2nd preimage resistant

Brute force (2nd) preimage

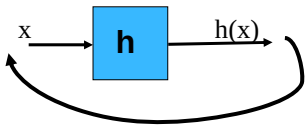
- **multiple target second preimage (1 out of many):** if one can attack 2^t simultaneous targets, the effort to find a single preimage is 2^{n-t}
- **multiple target second preimage (many out of many):**
  - time-memory trade-off with $\Theta(2^n)$ precomputation and storage $\Theta(2^{2n/3})$ time per (2nd) preimage: $\Theta(2^{2n/3})$ [Hellman'80]
  - full cost per (2nd) preimage from $\Theta(2^n)$ to $\Theta(2^{2n/5})$ [Wiener'02] (if $\Theta(2^{3n/5})$ targets are attacked)
- answer: randomize hash function with a parameter S (salt, key, spice,…)

The birthday paradox

- given a set with S elements
- choose r elements at random (with replacements) with r « S
- the probability p that there are at least 2 equal elements (a collision) $\cong 1 - \exp(-r(r-1)/2S)$
- more precisely, it can be shown that
  - $p \geq 1 - \exp(-r(r-1)/2S)$
  - if $r < \sqrt{2S}$ then $p \geq 0.6\, r\,(r-1)/2S$

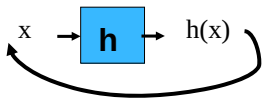
Brute force collision search

- Consider the functional graph of h

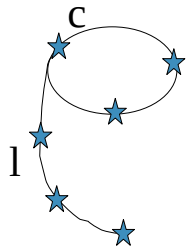


Brute force collision search

- low memory and parallel implementation of the birthday attack [Pollard'78][Quisquater'89][Wiener-van Oorschot'94]



- distinguished point (d bits)
  - $\Theta(e2^{n/2} + e\ 2^{d+1})$ steps with e the cost of one function evaluation
  - $\Theta(n2^{n/2-d})$ memory
  - full cost: $\Theta(e\ n2^{n/2})$ [Wiener'02]



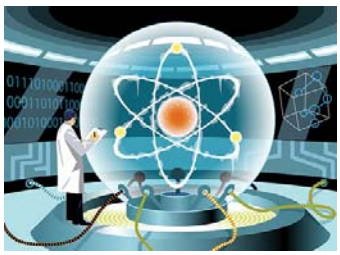
$l = c = (\pi/8)\ 2^{n/2}$

Brute force attacks in practice

- (2nd) preimage search
  - n = 128: 23 B$ for 1 year if one can attack $\mathbf{2^{40}}$ targets in parallel
- parallel collision search with low memory
  - n = 128: 1 M$ for 8 hours (or 1 year on 100K PCs)
  - n = 160: 90 M$ for 1 year
  - need 256-bit result for long term security (30 years or more)

Quantum computers

- in principle exponential parallelism
- inverting a one-way function: 2^n reduced to $2^{n/2}$ [Grover'96]
- collision search:
  - $2^{n/3}$ computation + hardware [Brassard-Hoyer-Tapp'98]
  - [Bernstein'09] classical collision search requires $2^{n/4}$ computation and hardware (= standard cost of $2^{n/2}$)



Collision resistance

- **hard to achieve in practice**
  - many attacks
  - requires double output length $2^{n/2}$ versus 2^n
- **hard to achieve in theory**
  - [Simon'98] one cannot derive collision resistance from "general" preimage resistance (there exists no black box reduction)
- **hard to bypass:**
  - UOWHF (TCR, eSec) randomize hash function after choosing the message [Naor-Yung'89]
    - how to enforce this in practice?
  - randomized hashing: RMX mode [Halevi-Krawczyk'05]
    $H(r \| x_1 \oplus r \| x_2 \oplus r \| \ldots \| x_t \oplus r)$
    - needs e-SPR (not met by MD5)
    - issues with insider attacks (i.e. attacks by the signer)

Formalizing the definitions is tricky

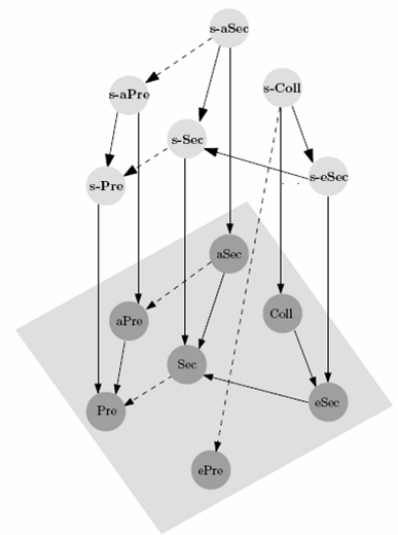
- for collision resistance: formalization requires a family of functions indexed by a parameter S
  - alternatively, one can formalize human ignorance [Stinson'06], [Rogaway'06]
- for (2nd) preimage resistance, one can choose the challenge (x) and/or the key (S) that selects the function. This gives three flavors [Rogaway-Shrimpton'04]:
  - random challenge, random key (Pre and Sec)
  - random key, fixed challenge (ePre and eSec - everywhere) (eSec=UOWHF)
  - fixed key, random challenge (aPre and aSec - always)
- can an attacker use S' ≠ S?
- complex relationship (see figure on next slide)

Relation between properties

[Rogaway-Shrimpton'04]

[Stinson'06]

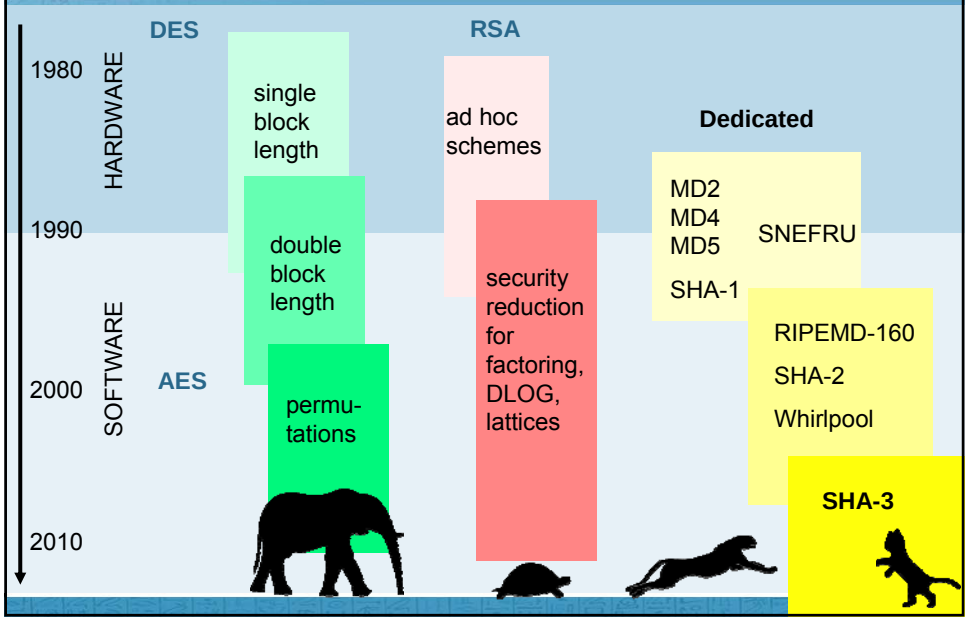
[Reyhanitabar-Susilo-Mu'10]



Properties in practice

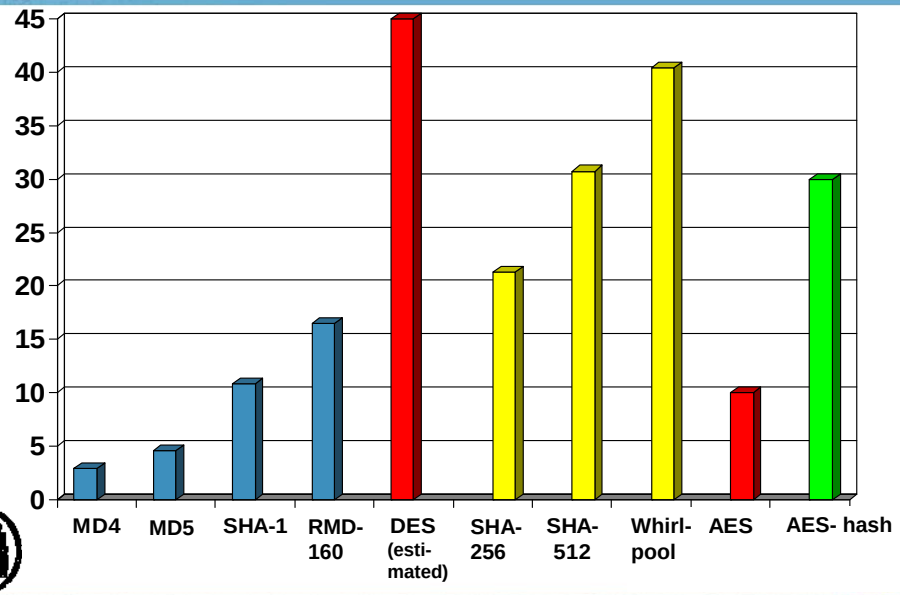
- collision resistance is not always necessary
- other properties are needed:
  - pseudo-randomness if keyed (with secret key)
  - pseudo-random oracle property
  - near-collision resistance
  - partial preimage resistance
  - multiplication freeness
- how to formalize these requirements and the relation between them?

Hash function history 101



Performance of hash functions - Bernstein

(cycles/byte) AMD Intel Pentium D 2992 MHz (f64)



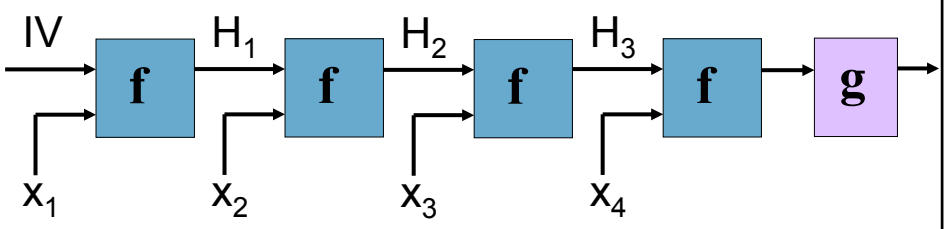
Applications

- protection of passwords
- data authentication
- digital signatures
- confirmation of knowledge/commitment
- micropayments

- pseudo-random string generation/key derivation
- construction of MAC algorithms, stream ciphers, block ciphers,…

Iteration
(mode of compression function)

Hash function: iterated structure

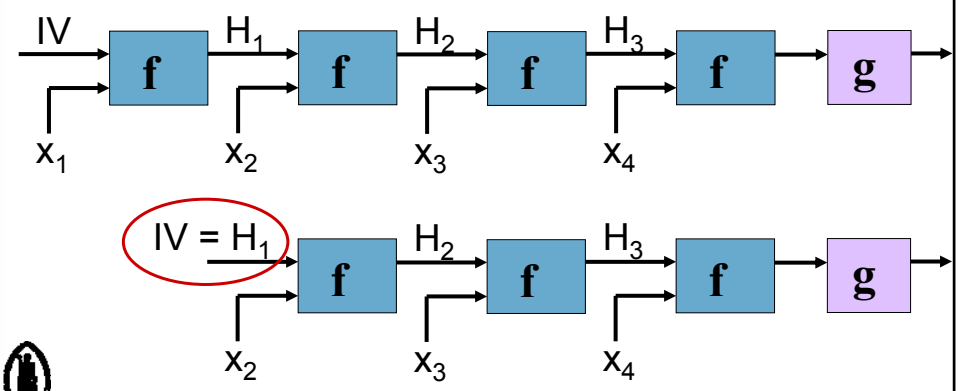


split messages into blocks of fixed length and hash them block by block with a compression function f

efficient and elegant
but …

Security relation between f and h

- iterating f can degrade its security
  - trivial example: 2nd preimage



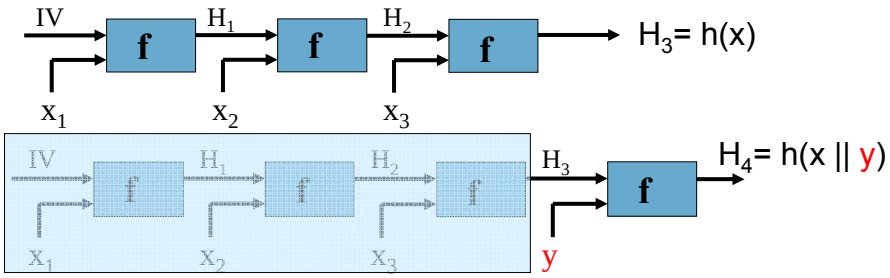
Security relation between f and h (2)

- solution: Merkle-Damgård (MD) strengthening
  - fix IV, use unambiguous padding and insert length at the end
- f is collision resistant $\Rightarrow$ h is collision resistant
  [Merkle'89-Damgård'89]
- f is ideally 2nd preimage resistant $\Leftrightarrow$ h is ideally 2nd preimage resistant [Lai-Massey'92] ?

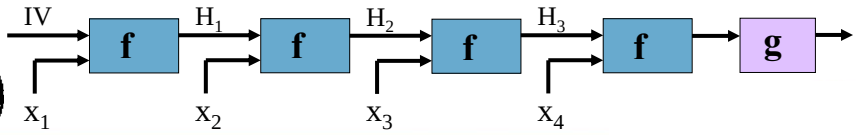
- few hash functions have a strong compression function
- very few hash functions treat x_i and H_{i-1} in the same way
- composition results for preimage resistance tricky

Security relation between f and h (3)

length extension: if one knows h(x), easy to compute h(x || y) without knowing x



Solution: output transformation

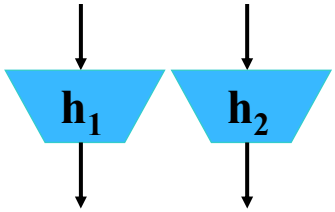


Some attacks on MD: 1999-2005

- multi-collision attack and impact on concatenation [Joux'04]
  - the concatenation of 2 iterated hash functions ($g(x)= h_1(x) \| h_2(x)$) is as most as strong as the strongest of the two (even if both are independent)
- long message 2nd preimage attack [Dean-Felten-Hu'99], [Kelsey-Schneier'05]
  - if one hashes **2^t message blocks** with an iterated hash function, the effort to find a second preimage is only $\mathbf{2^{n-t+1} + t\, 2^{n/2+1}}$
  - appending the length does not help here!

How (NOT) to strengthen a hash function? [Joux'04]

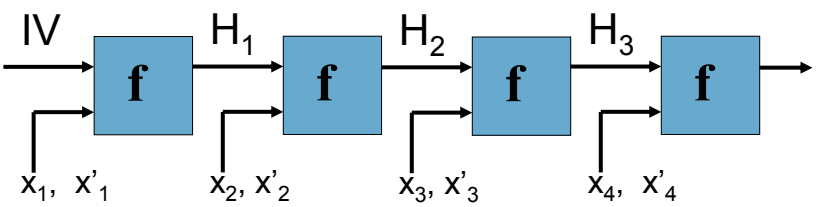
- answer: concatenation
- h_1 (n1-bit result) and h_2 (n2-bit result)



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

- intuition: the strength of g against collision/(2nd) preimage attacks is the product of the strength of h_1 and h_2
  - if both are "independent"
- but….

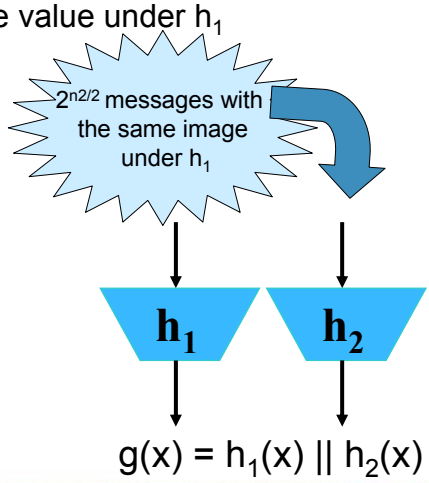
Multi-collisions [Joux '04]



- for IV: 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 $h(x_1||x_2||x_3||x_4) = h(x'_1||x_2||x_3||x_4) = h(x'_1||x'_2||x_3||x_4) = \ldots$
  $= h(x'_1||x'_2||x'_3||x'_4)$ **a 16-fold collision**

How (NOT) to strengthen a hash function? [Joux'04]

- h_1 (n1-bit result) and h_2 (n2-bit result)
- find a $2^{n2/2}$-fold multi-collision for h_1, that is, a huge set of messages that map to the same value under h_1
- by the birthday paradox, with high probability two of the values in this set will collide under h_2
- cost
  - computation $n2 \cdot 2^{n1/2} + 2^{n2/2}$
  - memory $2^{n2/2}$



Formal results [Joux '04]

consider h_1 (n1-bit result) and h_2 (n2-bit result), with $n2 \geq n1$.

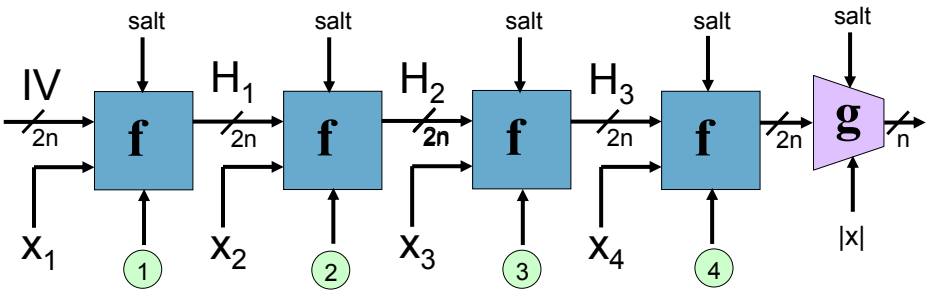
concatenation of 2 iterated hash functions ($g(x)= h_1(x) \| h_2(x)$) is as most as strong as the strongest of the two (even if both are independent)

- cost of collision attack against g at most
  $n2 \cdot 2^{n1/2} + 2^{n2/2} << 2^{(n1 + n2)/2}$
- cost of (2nd) preimage attack against g at most
  $n2 \cdot 2^{n1/2} + 2^{n1} + 2^{n2} << 2^{n1 + n2}$
- if either of the functions is weak, the attacks may work better

Summary

Improving MD iteration

salt + output transformation + counter + wide pipe



security reductions well understood
many more results on property preservation

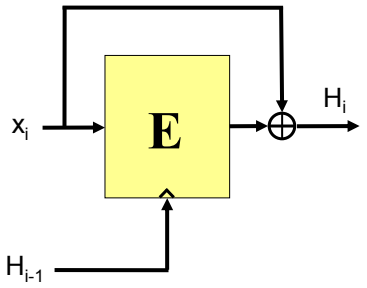
Improving MD iteration

- degradation with use: salting (family of functions, randomization)
- extension attack + PRO preservation: strong output transformation g (which includes total length and salt)
- long message 2nd preimage: preclude fix points
  - counter $f \rightarrow f_i$ [Biham-Dunkelman]
- multi-collisions, herding: avoid breakdown at $2^{n/2}$ with larger internal memory: known as wide pipe
  - e.g., extended MD4, RIPEMD, [Lucks'05]

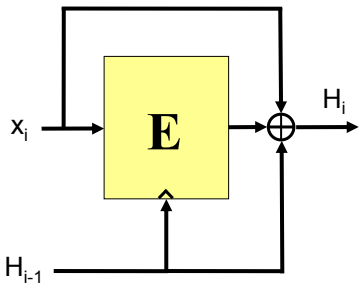
Compression functions

Block cipher (E_K) based

Davies-Meyer



Miyaguchi-Preneel



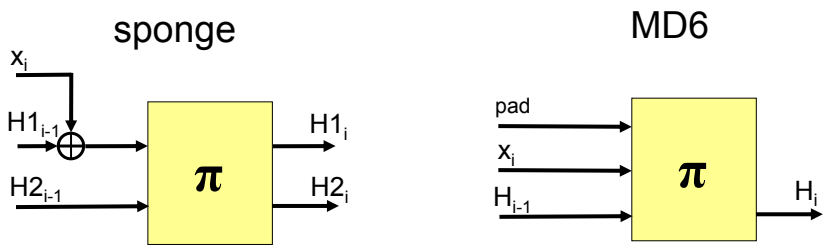
- output length = block length
- 12 secure compression functions (in ideal cipher model)
- requires 1 key schedule per encryption

Block cipher (E_K) based

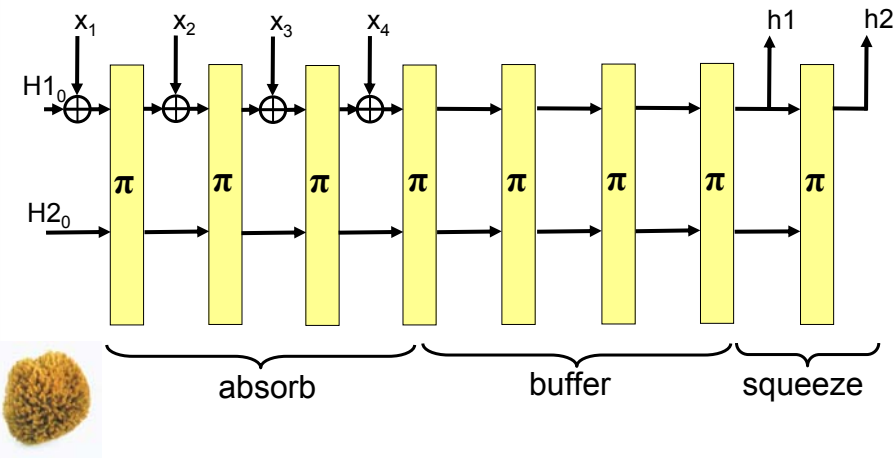
- which assumptions are needed on the block cipher E to prove MD iterated Davies-Meyer secure?
  - standard model: no security results (PRF/PRP is not sufficient)
  - ideal cipher model: ok to prove collision resistance and (second) preimage resistance
    - can this be relaxed?
  - not PRO preserving (length extension problem)
  - PRA preserving

Permutation (π) based

Large permutation



Permutation (π) based: sponge

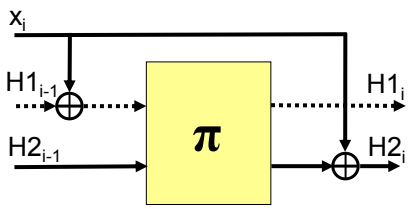


Examples: Panama, RadioGatun, Grindahl, Keccak (no buffer)

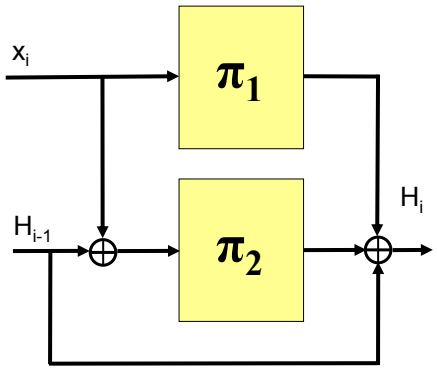
Permutation (π) based

small permutation

JH



Grøstl



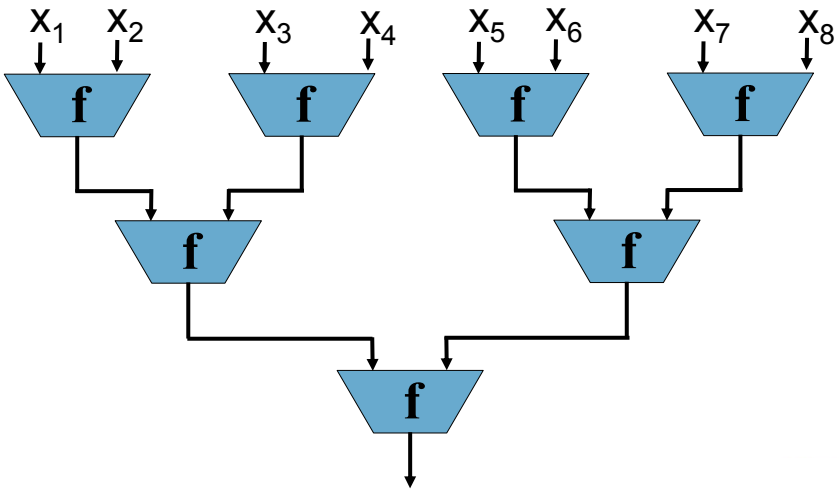
Iteration modes

- security of simple modes well understood
- powerful tools available

- analysis of slightly more complex schemes very difficult
- which properties are meaningful?
- which properties are preserved?
- MD versus sponge is still open debate

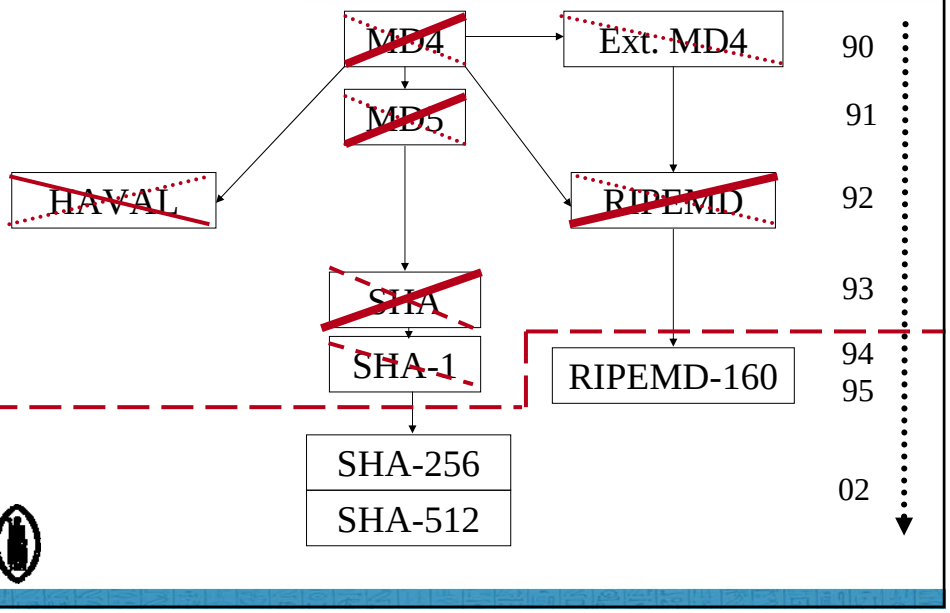
Tree structure: parallelism

[Damgård'89], [Pal-Sarkar'03]



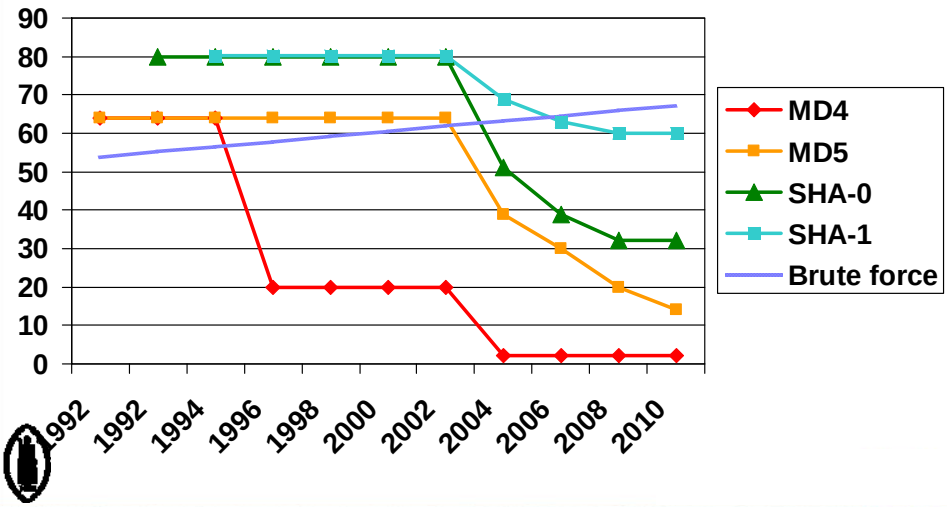
SHA-{0,1,2}

MDx-type hash function history



The complexity of collision attacks

brute force: 1 million PCs (1 year) or US$ 100,000 hardware (4 days)

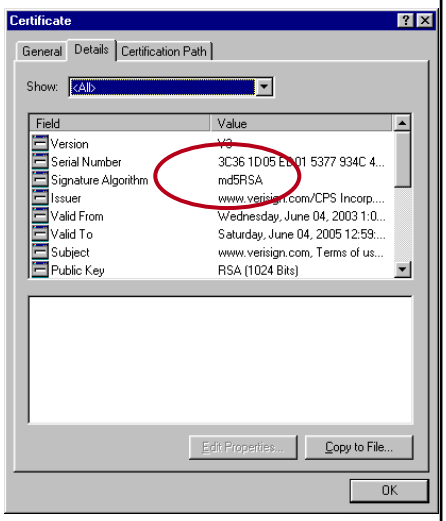


MD5 [Rivest'91]

- 4 rounds (64 steps)
- pseudo-collisions [denBoer-Bosselaers'93]
- collisions for compression function [Dobbertin'96]
- collisions for hash function
  - [Wang+'04] – 15 minutes
  - …
  - [Stevens+'09] – milliseconds
  - brute force (2^{64}): 1M$ 8 hours in 2010
- 2nd preimage in 2^{123} [Sasaki-Aoki'09]

MD5

- advice (RIPE since '92, RSA since '96): stop using MD5
- largely ignored by industry until 2009 (click on a cert...)



SHA-1 [NIST'95]

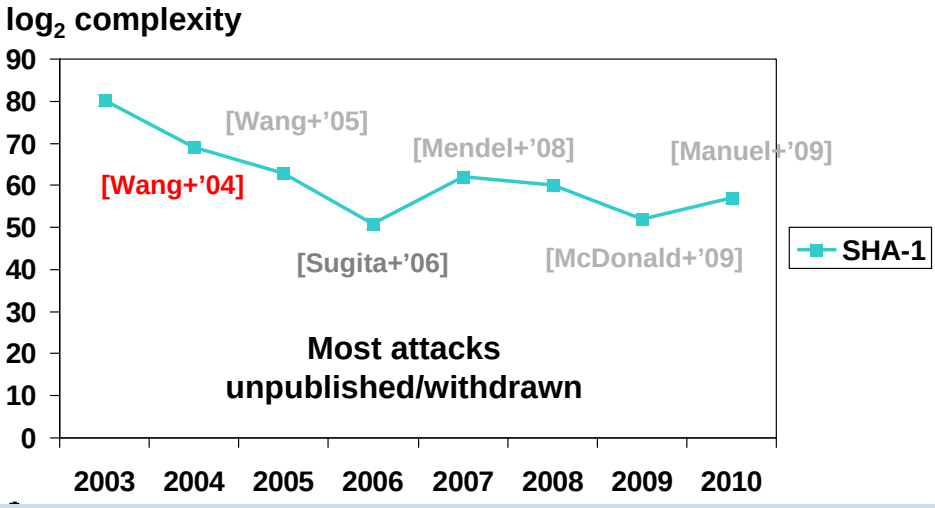
- fix to SHA-0
- add rotation to message expansion: quasicyclic code, $d_{min} = 25$
  $w_j \leftarrow (w_{j-3} \oplus w_{j-8} \oplus w_{j-14} \oplus w_{j-16}) >>> 1 \quad j > 15$

collisions

- 53 steps [Oswald-Rijmen'04 and Biham-Chen'04]
- 58 steps [Wang+'05]
- 64 steps in 2^{35} – highly structured [De Cannière-Rechberger'06-'07]:
- 70 steps in 2^{44} – highly structured [De Cannière-Rechberger'06-'07]:
- 70 steps 2^{39} (4 days on a PC) [Joux-Peyrin'07]
- 2^{69} [Wang+'05]
- 2^{63} ? [Wang+'05 - unpublished]
- 2^{51} ? [Sugita+'06]
- 2^{62} ? [Mendel+'08 - unpublished]
- 2^{52} ?? [McDonald+'09 - unpublished]

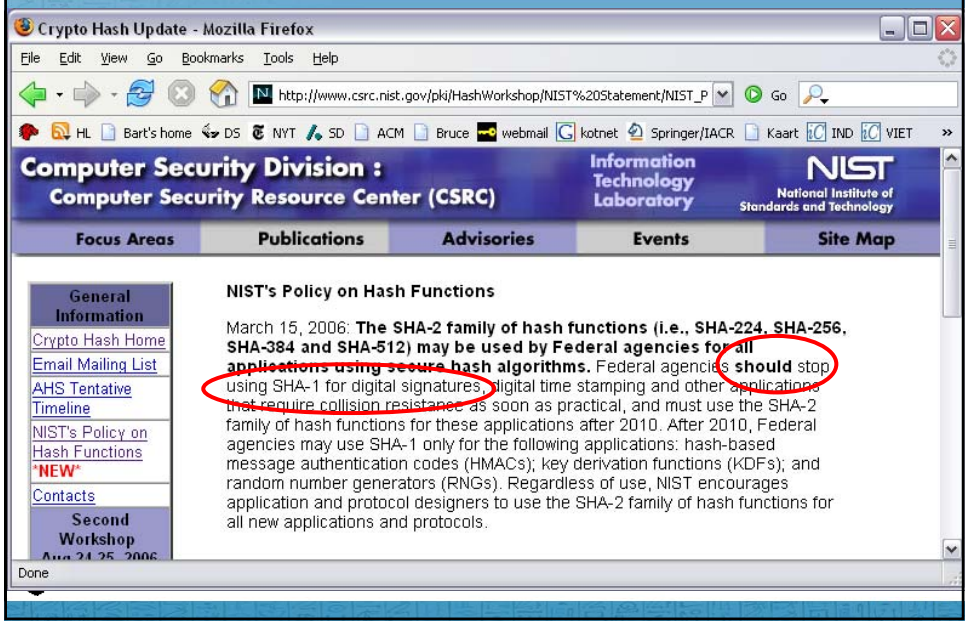
preimages for 48/80 steps in $2^{160-\varepsilon}$ [Aoki-Sasaki'09]

SHA-1



prediction: collision for SHA-1 in the next 12-18 months

NIST and SHA-1



Impact of collisions

- collisions for MD5, SHA-0, SHA-1
  - 2 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

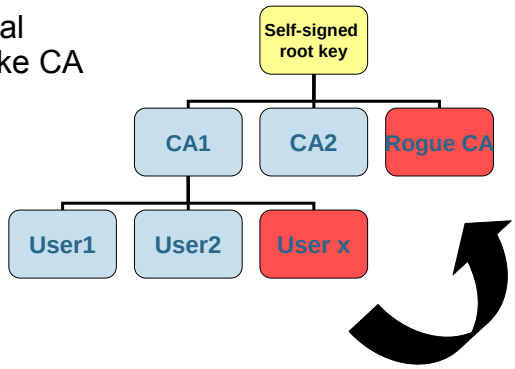
- what is achievable for MD5?
  - 2 colliding executables/postscript/gif/…[Lucks-Daum'05]
  - 2 colliding RSA public keys – thus with colliding X.509 certificates [Lenstra+'04]
  - chosen prefix attack: different IDs, same certificate [Stevens+'07]
  - **2 arbitrary colliding files (no constraints) in 8 hours for 1 M$**

Rogue CA attack

[Sotirov-Stevens-Appelbaum-Lenstra-Molnar-Osvik-de Weger '08]

- request user cert; by special collision this results in a fake CA cert (need to predict serial number + validity period)

impact: **rogue CA** that can issue certs that are trusted by all browsers



- 6 CAs have issued certificates signed with MD5 in 2008:
  - Rapid SSL, Free SSL (free trial certificates offered by RapidSSL), TC TrustCenter AG, RSA Data Security, Verisign.co.jp

Impact of MD5 collisions

- digital signatures: only an issue if for non-repudiation
- 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 2005 (cf. traffic tickets in Australia)

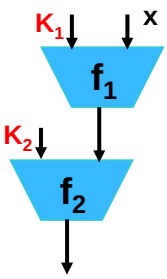
And (2nd) preimages?

- security degrades with number of applications
- for large messages even with the number of blocks (cf. supra)
- specific results:
  - MD2: 2^{73} [Knudsen+09]
  - MD4: 2^{102} [Leurent'08]
  - MD5: 2^{123} [Sasaki-Aoki'09]
  - SHA-0: 52 of 80 steps in $2^{156.6}$ [Aoki-Sasaki'09]
  - SHA-1: 48 of 80 steps in $2^{159.3}$ [Aoki-Sasaki'09]

HMAC

- HMAC keys through the IV (plaintext)
  - collisions for MD5 invalidate current security proof of HMAC-MD5

	Rounds in f2	Rounds in f1	Data complexity
MD4	48	48	2^{72} CP + 2^{77} time
MD5	64	33 of 64	$2^{126.1}$ CP
MD5	64	64	2^{51} CP & 2^{100} time (RK)
SHA-0	80	80	2^{109} CP
SHA-1	80	53 of 80	$2^{98.5}$ CP



Upgrades

- RIPEMD-160 is good replacement for SHA-1
- upgrading algorithms is always hard
- TLS uses MD5 || SHA-1 to protect algorithm negotiation (up to v1.1)
- **upgrading negotiation algorithm is even harder: need to upgrade TLS 1.1 to TLS 1.2**

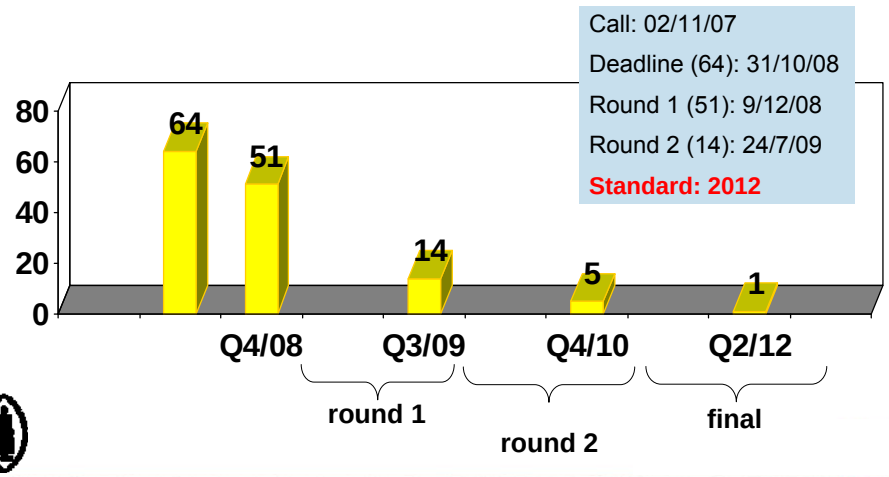
SHA-2 [NIST'02]

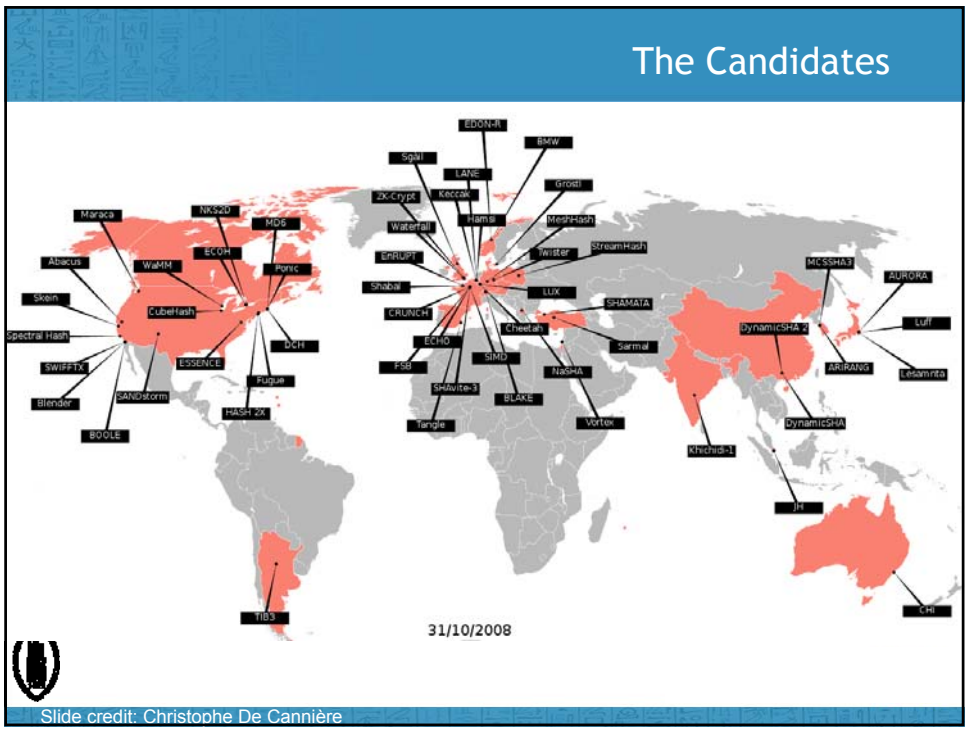
- SHA-224, SHA-256, SHA-384, SHA-512
  - non-linear message expansion
  - more complex operations
  - 64/80 steps
  - SHA-384 and SHA-512: 64-bit architectures
- SHA-256 collisions: 24/64 steps [Sanadhya-Sarkar'08]
- SHA-256 preimages: 43/64 steps [Aoki+'09]
- implementations today faster than anticipated
- adoption
  - industry may migrate to SHA-2 by 2011 or may wait for SHA-3
  - very slow for TLS/IPsec (no pressing need)

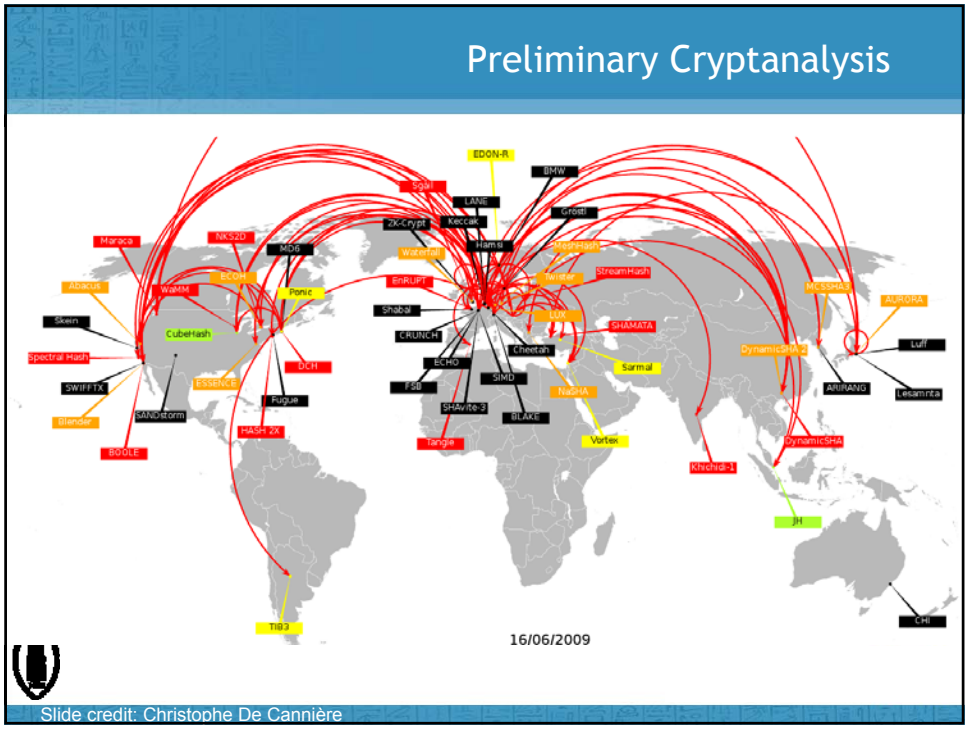
SHA-3
(bits and bytes)

NIST AHS competition (SHA-3)

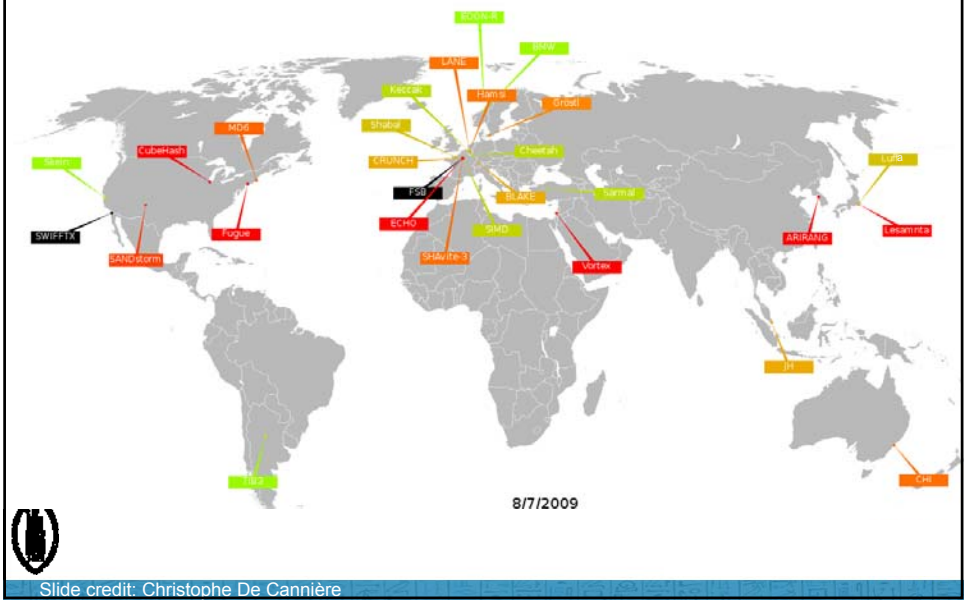
- SHA-3 must support 224, 256, 384, and 512-bit message digests, and must support a maximum message length of at least 2^{64} bits




The Candidates
EDON-R
BMW
Sgail
LANE
Grøstl
2K-Crypt
Keccak
Hamsi
MeshHash
Waterfall
Twister
StreamHash
EnRUPT
Maraca
NK52D
MD6
ECOH
WaMM
Ponic
Abacus
Skein
Shabal
LUX
SHAMATA
CRUNCH
CubeHash
Spectral Hash
Cheetah
Sarmal
ECHO
DCH
SIMD
NaSHA
FSB
SWIFFTX
ESSENCE
Fugue
SHAvite-3
BLAKE
Blender
SANDstorm
HASH 2X
Tangle
Vortex
BOOLE
MCSSHA3
AURORA
DynamicSHA 2
Luffa
ARIRANG
Lesamnta
DynamicSHA
Khichidi-1
JH
TIB3
CHI
31/10/2008
Slide credit: Christophe De Cannière

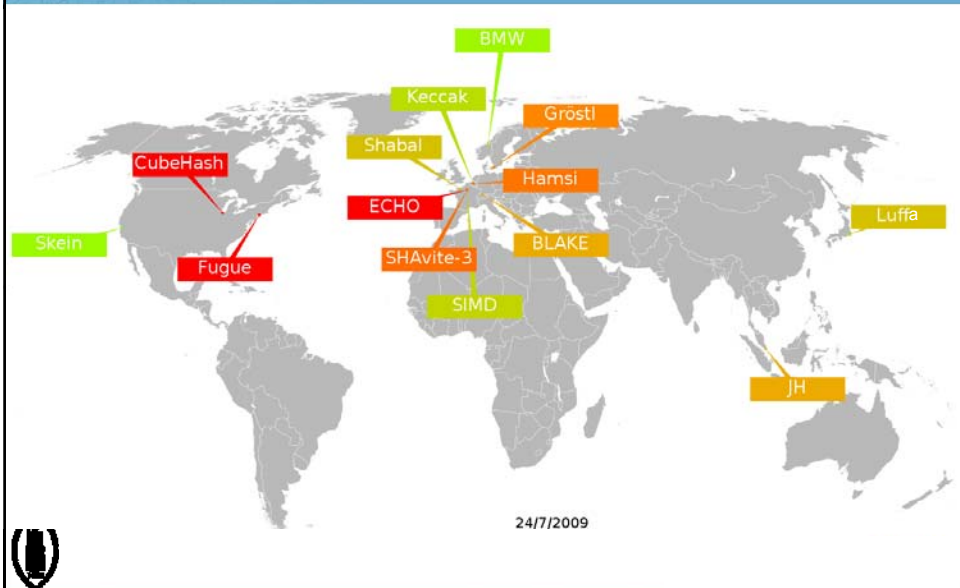

Preliminary Cryptanalysis
EDON-R
BMW
Sgail
LANE
Grøstl
2K-Crypt
Keccak
Hamsi
Waterfall
MeshHash
Twister
StreamHash
EnRUPT
Maraca
NK52D
MD6
ECOH
WaMM
Ponic
Abacus
Skein
Shabal
LUX
SHAMATA
CRUNCH
CubeHash
Spectral Hash
Cheetah
Sarmal
ECHO
DCH
SIMD
NaSHA
FSB
SWIFFTX
ESSENCE
Fugue
SHAvite-3
BLAKE
Blender
SANDstorm
HASH 2X
Tangle
Vortex
BOOLE
MCSSHA3
AURORA
DynamicSHA 2
Luffa
ARIRANG
Lesamnta
DynamicSHA
Khichidi-1
JH
TIB3
CHI
16/06/2009
Slide credit: Christophe De Cannière

End of Round 1 Candidates



Slide credit: Christophe De Cannière

Round 2 Candidates

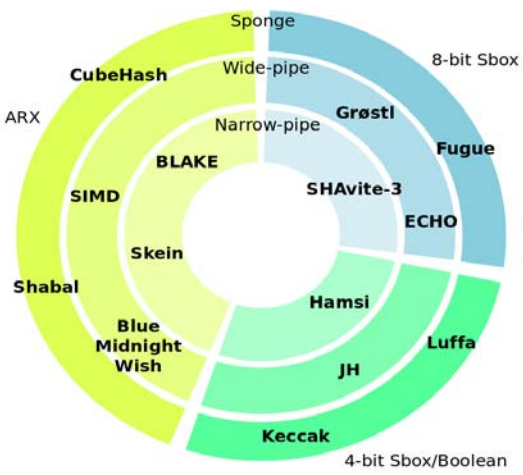


Slide credit: Christophe De Cannière

Compression function/iteration

	Block cipher	Permutation	MD/HAIFA
Blake	PGV variant		HAIFA
BMW	PGV variant		EMD
Cubehash		Sponge	
ECHO			HAIFA
Fugue		Sponge	
Grøstl		2-permutation	MD
Hamsi		Truncated/Sponge	
JH			JH-specific
Keccak		Sponge	
Luffa		Sponge-like	
Shabal		Sponge	
Shavite-3	Davies-Meyer		HAIFA
SIMD	PGV variant		MD
Skein	Davies-Meyer		MD/Tree

Properties: bits and bytes
[Watanabe'10]



Security Reductions [Mennink-Andreeva-Preneel'10]

	type	sf	pf	$\mathbf{Adv}_f^{\mathrm{pre}}$	$\mathbf{Adv}_f^{\mathrm{sec}}$	$\mathbf{Adv}_f^{\mathrm{col}}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{pre}}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{sec}}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{gcol}}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{col}}$	$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{pro}}$
BLAKE	HAIFA	✓	✓								
BMW	chop-(MD+FT)	✓	✗								
CubeHash	chop-(MD+FT)	✗	✗								
ECHO	chop-HAIFA	✓	✓								
Fugue	chop-(MD+FT)	✓	✗								
Grøstl	chop-(MD+FT)	✓	✗								
Hamsi	MD+FT	✓	✗								
JH	chop-MD	✓	✗								
Keccak	chop-MD	✗	✗								
Luffa	chop-(MD+FT)	✗	✗								
Shabal	chop-MD	✓	✓								
SHAvite-3	HAIFA	✓	✓								
SIMD	chop-(MD+FT)	✓	✗								
Skein	MD	✓	✓								

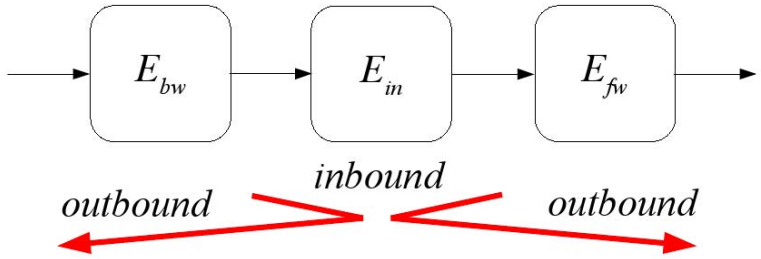
Table 1. A schematic summary of all results. The *first* column describes the hash function construction, and the *second* and *third* column show which hash functions have a suffix-free (sf) or prefix-free (pf) padding. A *green* box indicates the existence of a non-trivial upper bound, a *red* box means that an efficient adversary is known for the security notion, and a *yellow* box indicates that no result is known, but recent literature gives some confidence in the existence of a non-trivial bound.

Issues arisen during Round 1

- round 1 was very short; several functions received no outside analysis
- 7 out of 14 designs were tweaked at the beginning of round 2
- security:
  - controversy around pseudo-collision attacks and memory requirements
  - proofs have not helped much to survive
- performance: roughly as fast or faster than SHA-2
  - tunable security/performance tradeoff: nominal parameters?
  - large memory (> 100 bytes) may be a problem for small devices
  - can we exploit 64 or 128 cores? Intel AES instruction?

Rebound Attack

a new variant of differential cryptanalysis

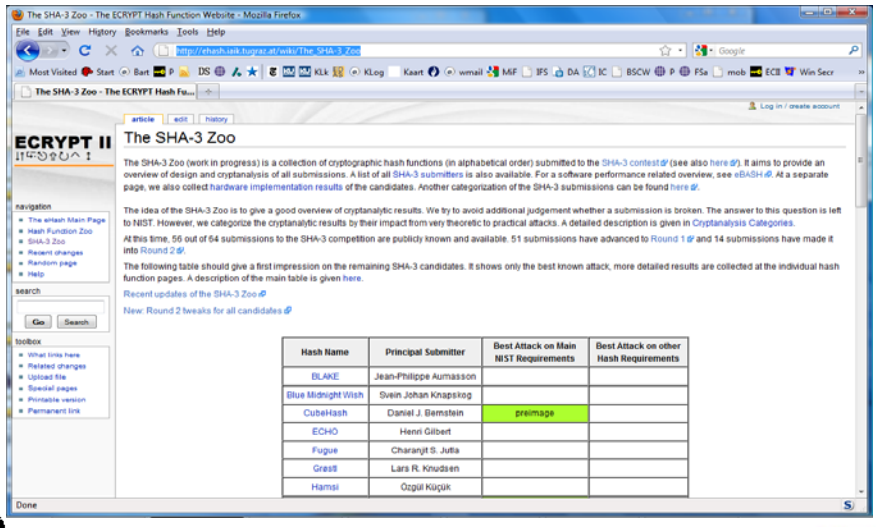


developed during the design of Grøstl [MRST09]

already successfully applied to Whirlpool and the SHA-3 candidates Twister, Lane, and reduced versions of others

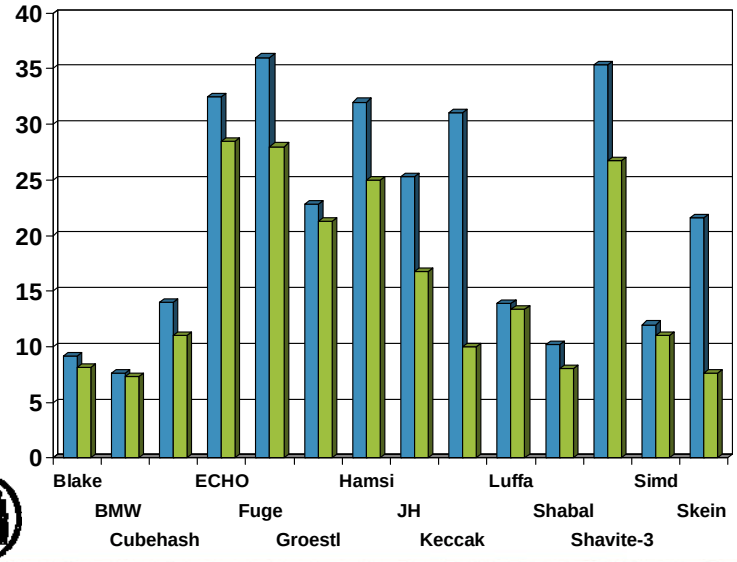
Slide credit: Christian Rechberger

Security: SHA-3 Zoo
http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo



Performance of hash functions

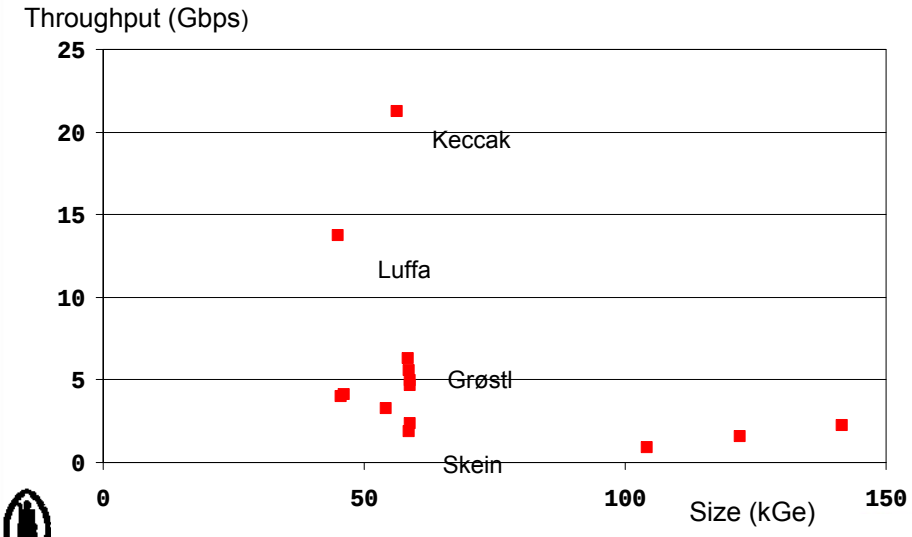
[Bernstein09] http://bench.cr.yp.to/ebash.html



256-bit hash,
32/64-bit code
(cycles/byte)

Hardware Performance

[Tillich+'09] IACR ePrint 2009/510



SHA-4?

- an open competition such as SHA-3 is bound to result in new insights between 2009-2012
- only few of these can be incorporated using “tweaks”
- the winner selected in 2012 will reflect the state of the art in October 2008
- nevertheless, it is unlikely that we will have a SHA-4 competition before 2030

Hash functions: conclusions

- SHA-1 would have needed 128-160 steps instead of 80
- 2004-2009 attacks: cryptographic meltdown but not dramatic for most applications
  - clear warning: upgrade asap
- theory is developing for more robust iteration modes and extra features; still early for building blocks
- nirwana: efficient hash functions with security reduction