KU LEUVEN

Introduction to the Design and Cryptanalysis of Cryptographic Hash Functions

Bart Preneel
KU Leuven - COSIC
firstname.lastname@esat.kuleuven.be

Sibenik, June 2016

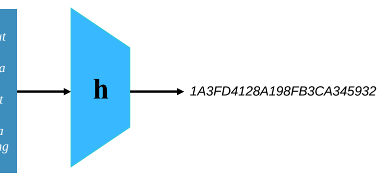
Hash functions

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 cryptographic 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).



2

Applications

- short unique identifier to a string
  - digital signatures
  - data authentication
- one-way function of a string
  - protection of passwords
  - micro-payments
- confirmation of knowledge/commitment
- pseudo-random string generation/key derivation
- entropy extraction
- construction of MAC algorithms, stream ciphers, block ciphers,…

2005: 800 uses of MD5 in Microsoft Windows

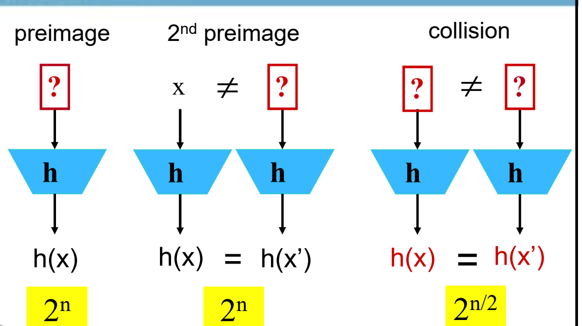
3

Agenda

- Definitions
- Iterations (modes)
- Compression functions
- Constructions
- SHA-3
- Conclusions

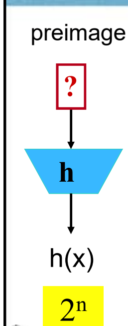
4

Security requirements (n-bit result)



5

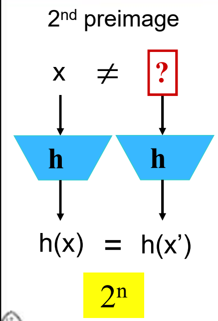
Preimage resistance

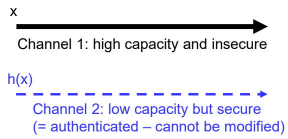


- in a password file, one does not store
  - (username, password)
- but
  - (username,hash(password))
- this is sufficient to verify a password
- an attacker with access to the password file has to find a preimage

6

Second preimage resistance



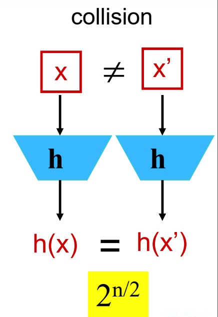


- an attacker can modify x but not h(x)
- he can only fool the recipient if he finds a second preimage of x

7

Collision resistance

- 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')



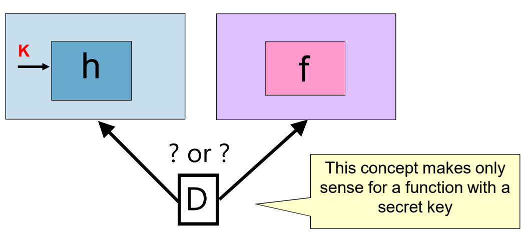
8

Pseudo-random function

computationally indistinguishable from a random function

$Adv_h^{prf} = \Pr[K \xleftarrow{\$} \mathcal{K}: A^{h_K(\cdot)} \Rightarrow 1] - \Pr[f \xleftarrow{\$} RAND(m,n): A^f \Rightarrow 1]$

RAND(m,n): set of all functions from m-bit to n-bit strings



9

Brute force (2^{nd}) 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 (2^{nd}) preimage: $\Theta(2^{2n/3})$ [Hellman'80]
- answer: randomize hash function with a parameter S (salt, key, spice,…)

10

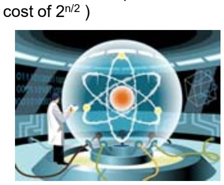
Brute force attacks in practice

- (2^{nd}) preimage search
  - n = 128: 14 B$ for 1 year if one can attack $\mathbf{2^{40}}$ targets in parallel
- parallel collision search: small memory using cycle finding algorithms (distinguished points)
  - n = 128: 1 M$ for 5 hours (or 1 year on 60K PCs)
  - n = 160: 56 M$ for 1 year
  - need 256-bit result for long term security (30 years or more)

11

Quantum computers

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



12

Properties in practice

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

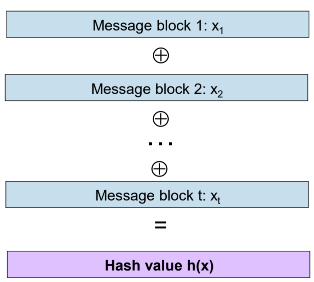
13

Iteration
(mode of compression function)

14

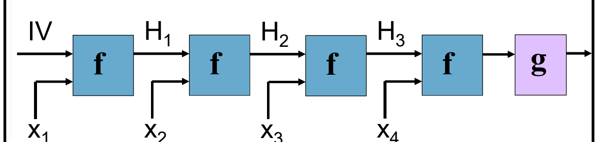
How not to construct a hash function

- Divide the message into t blocks x_i of n bits each



15

Hash function: iterated structure



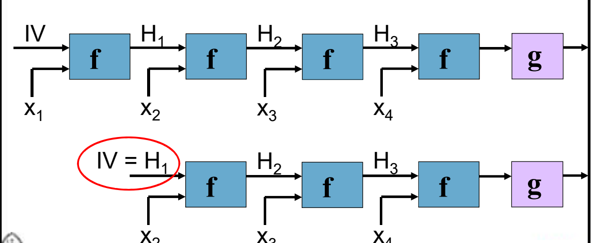
- split messages into blocks of fixed length and hash them block by block with a compression function f
- need padding at the end

efficient and elegant…. but …

16

Security relation between f and h

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



17

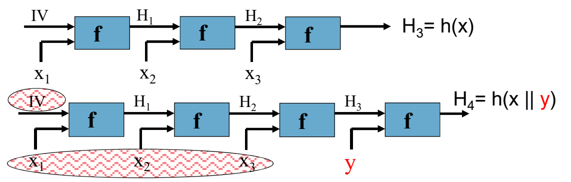
Security relation between f and h (2)

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

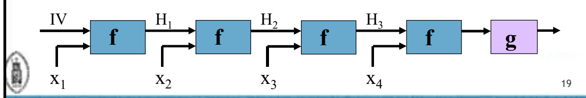
18

Security relation between f and h (3)

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



solution: output transformation



19

Attacks on MD-type iterations

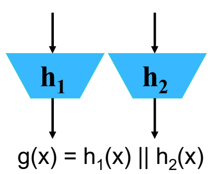
- **long message 2nd preimage attack** [Dean-Felten-Hu'99], [Kelsey-Schneier'05]
  - Sec security degrades lineary with number 2^t of message **blocks** hashed: $2^{n-t+1} + t\ 2^{n/2+1}$
  - appending the length does not help here!
- **multi-collision attack and impact on concatenation** [Joux'04]
- **herding attack** [Kelsey-Kohno'06]
  - reduces security of commitment using a hash function from 2^n
  - on-line 2^{n-t} + precomputation $2.2^{(n+t)/2}$ + storage 2^t

20

How (NOT) to strengthen a hash function?

[Coppersmith'85][Joux'04]

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



- 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….

21

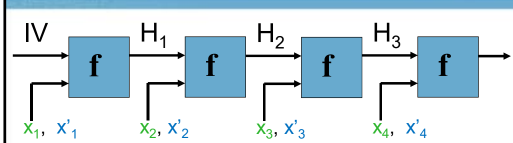
Multiple collisions ≠ multi-collision

Assume "ideal" hash function h with n-bit result

- $\Theta(2^{n/2})$ evaluations of h (or steps): 1 collision
  - $h(x)=h(x')$
- $\Theta(r.\ 2^{n/2})$ steps: r^2 collisions
  - $h(x_1)=h(x_1')$; $h(x_2)=h(x_2')$; … ; $h(x_{r^2})=h(x_{r^2}')$
- $\Theta(2^{2n/3})$ steps: a 3-collision
  - $h(x)= h(x')=h(x'')$
- $\Theta(2^{n(t-1)/t})$ steps: a t-fold collision (multi-collision)
  - $h(x_1)= h(x_2)= … =h(x_t)$

22

Multi-collisions on iterated hash function (2)



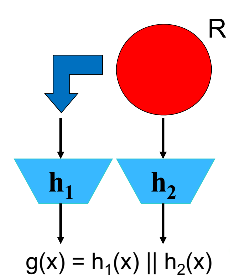
- for IV: collision for block 1: x_1, x'_1
- for H_1: collision for block 2: x_2, x'_2
- for H_2: collision for block 3: x_3, x'_3
- for H_3: collision for block 4: x_4, x'_4
- now $h(x_1||x_2||x_3||x_4) = h(x'_1||x_2||x_3||x_4) = h(x'_1||x'_2||x_3||x_4) = …$ $= h(x'_1||x'_2||x'_3||x'_4)$ **a 16-fold collision (time: 4 collisions)**

23

Multi-collisions

[Coppersmith'85][Joux '04]

- finding multi-collisions for an iterated hash function is not much harder than finding a single collision (if the size of the internal memory is n bits)
- algorithm
  - generate R = $2^{n1/2}$-fold multi-collision for h_2
  - in R: search by brute force for h_1
- Time: n1. $2^{n2/2} + 2^{n1/2}$ << $2^{(n1 + n2)/2}$



24

Multi-collisions
[Coppersmith'85][Joux '04]

consider h_1 (n1-bit result) and h_2 (n2-bit result), with n1 ≥ n2.

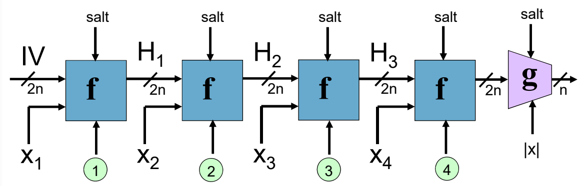
concatenation of 2 iterated hash functions ($g(x)= h_1(x) || h_2(x)$) is as most as strong as the strongest of the two (even if both are independent)

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

25

Improving MD iteration

salt + output transformation + counter + wide pipe



security reductions well understood
many more results on property preservation
impact of theory limited

26

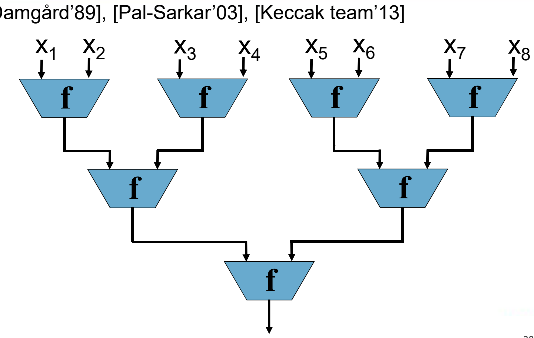
Improving MD iteration

- degradation with use: salting (family of functions, randomization)
  - or should a salt be part of the input?
- PRO: strong output transformation g
  - also solves length extension
- long message 2nd preimage: preclude fix points
  - counter $f \rightarrow f_i$ [Biham-Dunkelman'07]
- multi-collisions, herding: avoid breakdown at $2^{n/2}$ with larger internal memory: known as wide pipe
  - e.g., extended MD4, RIPEMD, [Lucks'05]

27

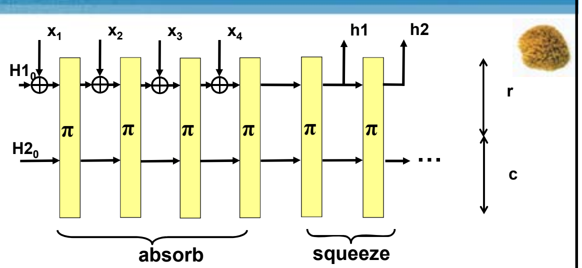
Tree structure: parallelism

[Damgård'89], [Pal-Sarkar'03], [Keccak team'13]



28

Permutation (π) based: sponge



if result has n bits, H1 has r bits (rate), H2 has c bits (capacity) and the permutation π is "ideal"

	collisions	$\min(2^{c/2}, 2^{n/2})$
	2nd preimage	$\min(2^{c/2}, 2^{n})$
	preimage	$\min(2^{c}, 2^{n})$

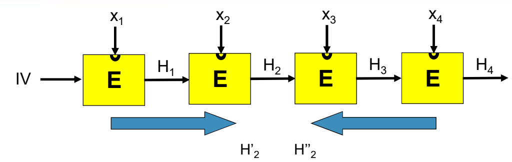
Modes: summary

- growing theory to reduce security properties of hash function to that of compression function (MD) or permutation (sponge)
  - preservation of large range of properties
  - relation between properties
- it is very nice to assume multiple properties of the compression function f, but unfortunately it is very hard to verify these
- still no single comprehensive theory

30

Compression functions

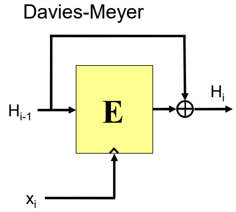
Single block length
[Rabin'78]



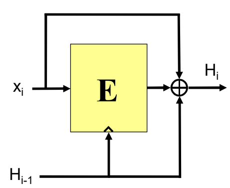
- Merkle's meet-in-the-middle: (2nd) preimage in time $2^{n/2}$
  - select $2^{n/2}$ values for (x_1,x_2) and compute forward H'_2
  - select $2^{n/2}$ values for (x_3,x_4) and compute backward H''_2
  - by the birthday paradox expect a match and thus a (2nd) preimage

Block cipher (E_K) based: single block length

Davies-Meyer



Miyaguchi-Preneel

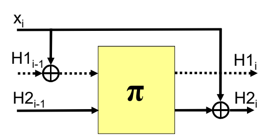


- output length = block length m; rate 1; 1 key schedule per encryption
- 12 secure compression functions (in ideal cipher model)
  - lower bounds: collision $2^{m/2}$, (2nd) preimage 2^m
- [Preneel+'93], [Black-Rogaway-Shrimpton'02], [Duo-Li'06], [Stam'09],…

Permutation (π) based

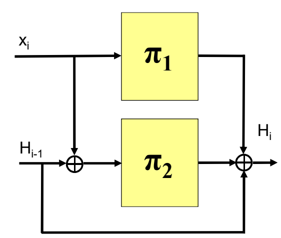
parazoa

JH

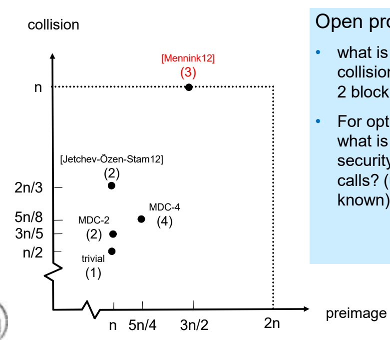


small permutation

Grøstl



Block cipher (E_K) based: double block length
(3n to 2n compression)



Open problems:

- what is the best collision/preimage security for 2 block cipher calls?
- For optimal collision security: what is the best preimage security for s block cipher calls? (upper bounds are known)

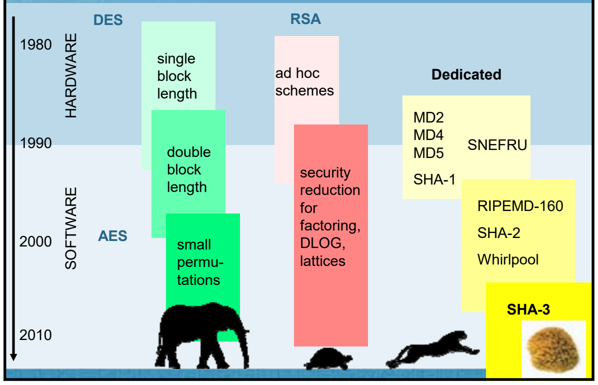
Iteration modes and compression functions

- security of simple modes well understood
- powerful tools available
- analysis of slightly more complex schemes very difficult
- MD versus sponge debate:
  - sponge is simpler
  - sponge easier to extend to authenticated encryption, MAC,..
  - should x_i and H_{i-1} be treated differently?

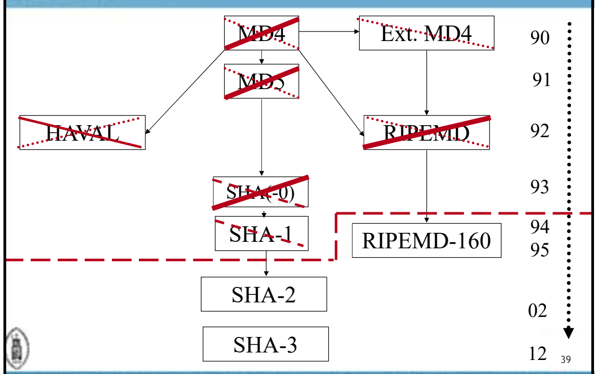
Hash function constructions

37

Hash function history 101



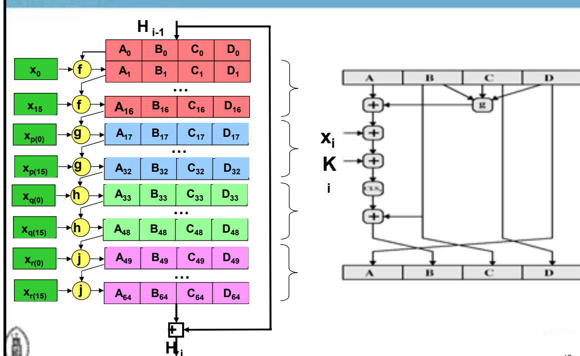
MDx-type hash function history



39

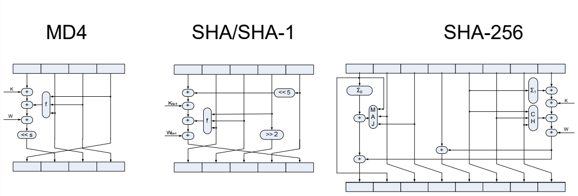
MD5: 4 rounds of 16 steps

[Rivest'91]



40

State updates in the MD4 family



Design principles copied in MD5, RIPEMD, HAVAL, SHA, SHA-1, SHA-256, ...

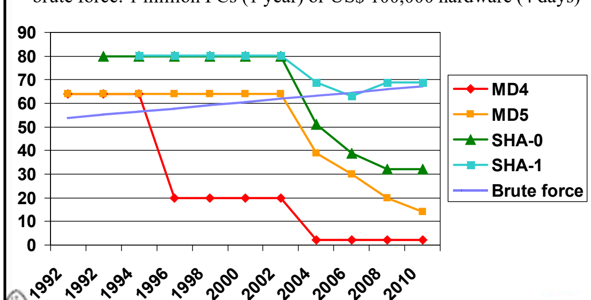
- All hash functions in use today

41

Slide credit: C. Rechberger

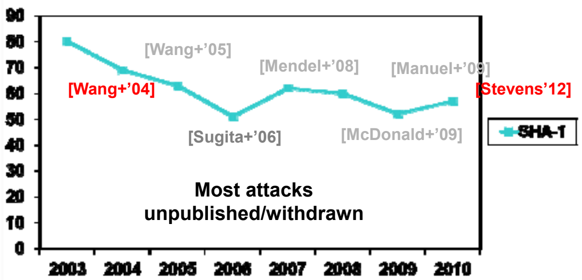
The complexity of collision attacks

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



42

SHA-1
designed by NSA in '94

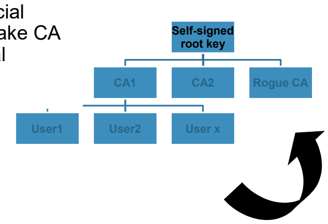


prediction: collision for SHA-1 in the next 12 months
- 75/80 steps takes $2^{57.7}$ [Grechnikov-Adinetz'11]

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

Collisions for SHA-1 compression function
[Stevens-Karpman-Peyrin'15]

- 10 days on a cluster of 64 GPUs (2K$)
  - does not lead to a collision for SHA-1 with fixed IV
  - compare to [denBoer-Bosselaers'93] for MD5
- by extrapolation: 100K$ for SHA-1 collision
- browser industry: planned stop accepting SHA-1 certs in 2017
  - September 2015: 28.2% of certs still use SHA-1

45

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 ('06) to TLS 1.2 ('08)**
  - progress in November 2013 (Google, Microsoft)
  - **but TLS 1.2 allows MD5 only!! SLOTH attack [late 2015]**

46

SHA-2 : FIPS 180
designed by NSA, published in 2002

- SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256
  - non-linear message expansion
  - 64/80 steps
  - SHA-384 and SHA-512: 64-bit architectures
- SHA-256 collisions: 31/64 steps $2^{65.5}$ [Mendel+'13]
  - free start collision: 52/64 steps ($2^{128-\varepsilon}$) [Li+12]
  - non-randomness 47/64 steps (practical) [Biryukov+11][Mendel+11]
- SHA-256 preimages: 45/64 steps ($2^{256-\varepsilon}$) [Khovratovitch'12]
- implementations today faster than anticipated
  - 18 cycles/byte on Core 2 (2008) → 7.8 cycles/byte on Haswell (2013)
- adoption accelerated by other attacks on TLS
  - since 2013 deployment in TLS 1.2

47

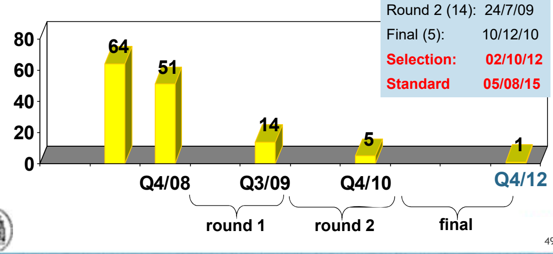
SHA-3
(bits and bytes)

48

NIST AHS competition (SHA-3)

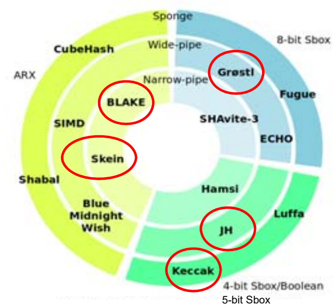
- SHA-3: 224, 256, 384, and 512-bit message digests
- (similar to SHA-2)

Call:	02/11/07
Deadline (64):	31/10/08
Round 1 (51):	09/12/08
Round 2 (14):	24/7/09
Final (5):	10/12/10
Selection:	02/10/12
Standard	05/08/15

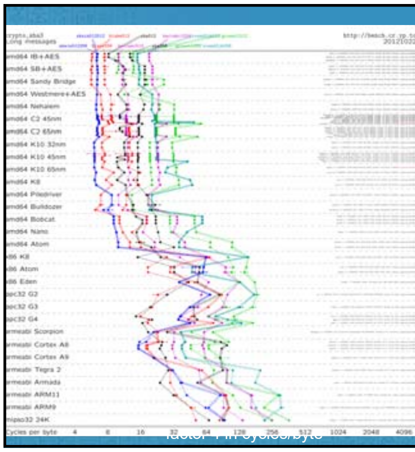


49

Properties: bits and bytes
[Watanabe'10]



Software performance
eBash [Bernstein-Lange]

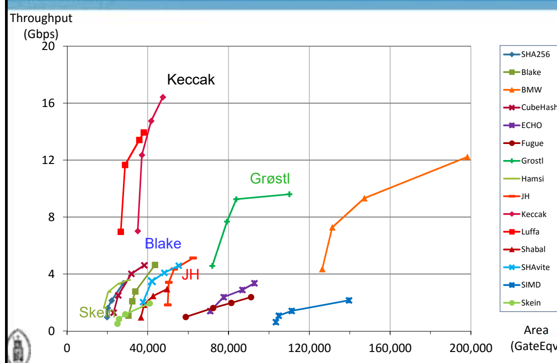


logarithmic scale

slower →

51

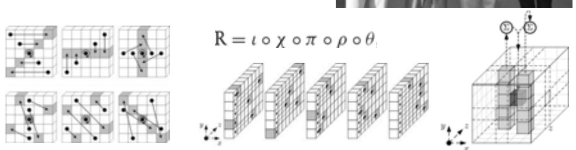
Hardware: post-place & route results
ASIC 130nm [Guo-Huang-Nazhandali-Schaumont'10]



Slide credit: Patrick Schaumont, Virginia Tech

52

Keccak

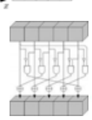


permutation: 25, 50, 100, 200, 400, 800, 1600

nominal version:

- 5x5 array of 64 bits
- 24 rounds of 5 steps

[Dinur+13]: collisions 5 out of 24 rounds



Keccak: FIPS 202
(published: 5 August 2015)

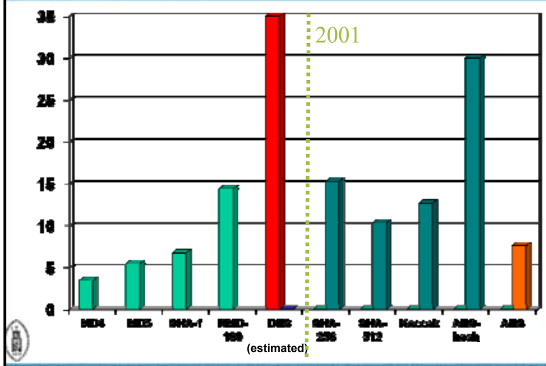
- append 2 extra bits for domain separation to allow
  - flexible output length (XOFs or eXtendable Output Functions)
  - tree structure (Sakura) allowed by additional encoding
- 6 versions
  - SHA3-224: n=224; c = 448; r = 1152 (72%)
  - SHA3-256: n=256; c = 512; r = 1088 (68%)
  - SHA3-384: n=384; c = 768; r = 832 (52%)
  - SHA3-512: n=512; c = 1024; r = 576 (36%)
  - SHAKE128: n=x; c = 256; r = 1344 (84%)
  - SHAKE256: n=x; c = 512; r = 1088 (68%)

(SHA3 versions: pad 01; SHAKE versions: pad 11 for XOF)

if result has n bits, H1 has r bits (rate), H2 has c bits (capacity) and the permutation π is "ideal"
collisions min ($2^{c/2}$, $2^{n/2}$)
2^{nd} preimage min ($2^{c/2}$, 2^n)
preimage min (2^c , 2^n)

Performance of hash functions – Bernstein-Lange

(cycles/byte) Intel Core 2 Quad Q9550; 4 x 2833MHz (2008)



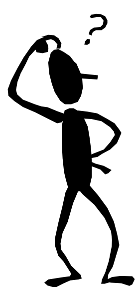
Hash functions: conclusions

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

56

The end

Thank you for your attention



57