The First 30 Years of Cryptographic Hash Functions and the NIST SHA-3 Competition

RSACONFERENCE2010

SECURITY DECODED

Bart Preneel
COSIC/Kath. Univ. Leuven (Belgium)

Session ID: CRYP-202
Session Classification: Hash functions decoded

Hash functions

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

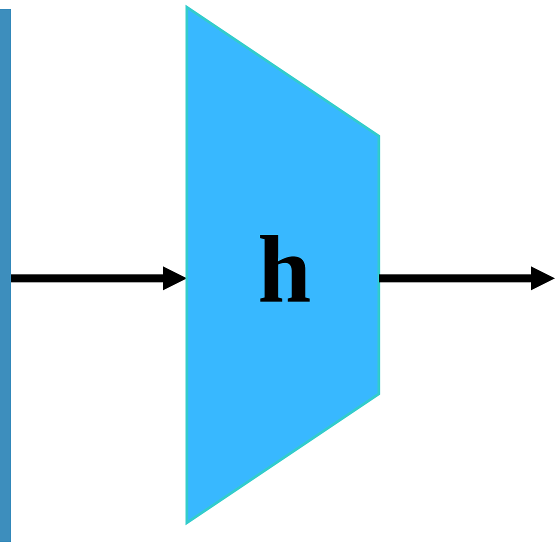
→

RIPEMD-160
SHA-256
SHA-512

→

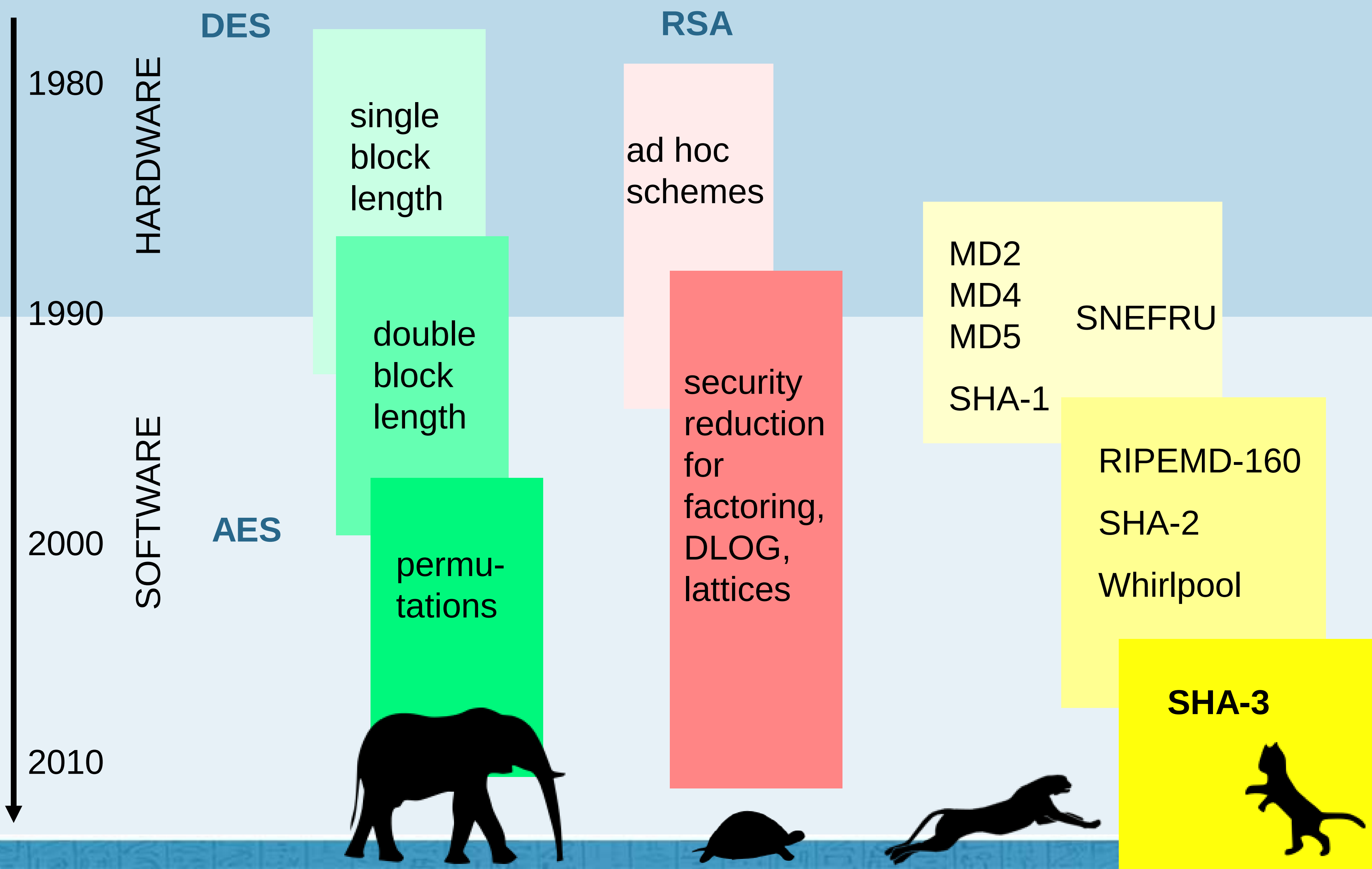
SHA-3

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).



1A3FD4128A198FB3CA345932

Hash function history 101



Applications

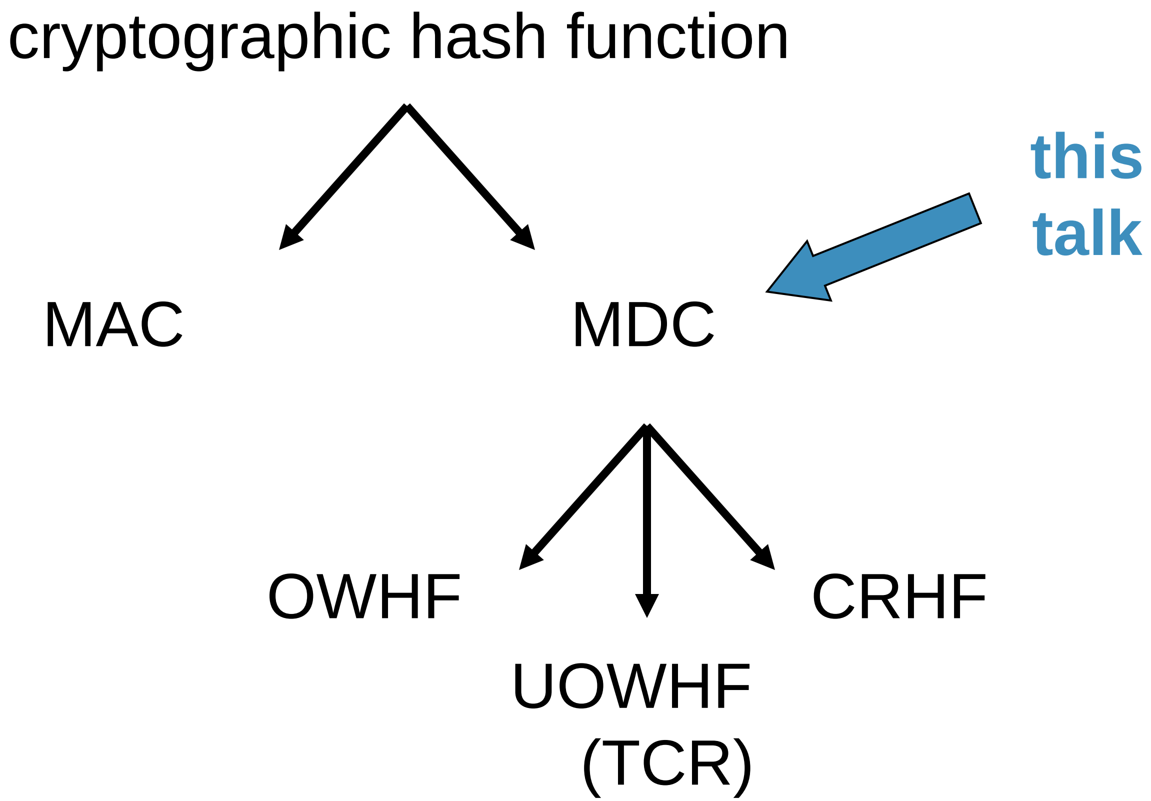
- digital signatures
- data authentication
- protection of passwords
- confirmation of knowledge/commitment
- micropayments

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

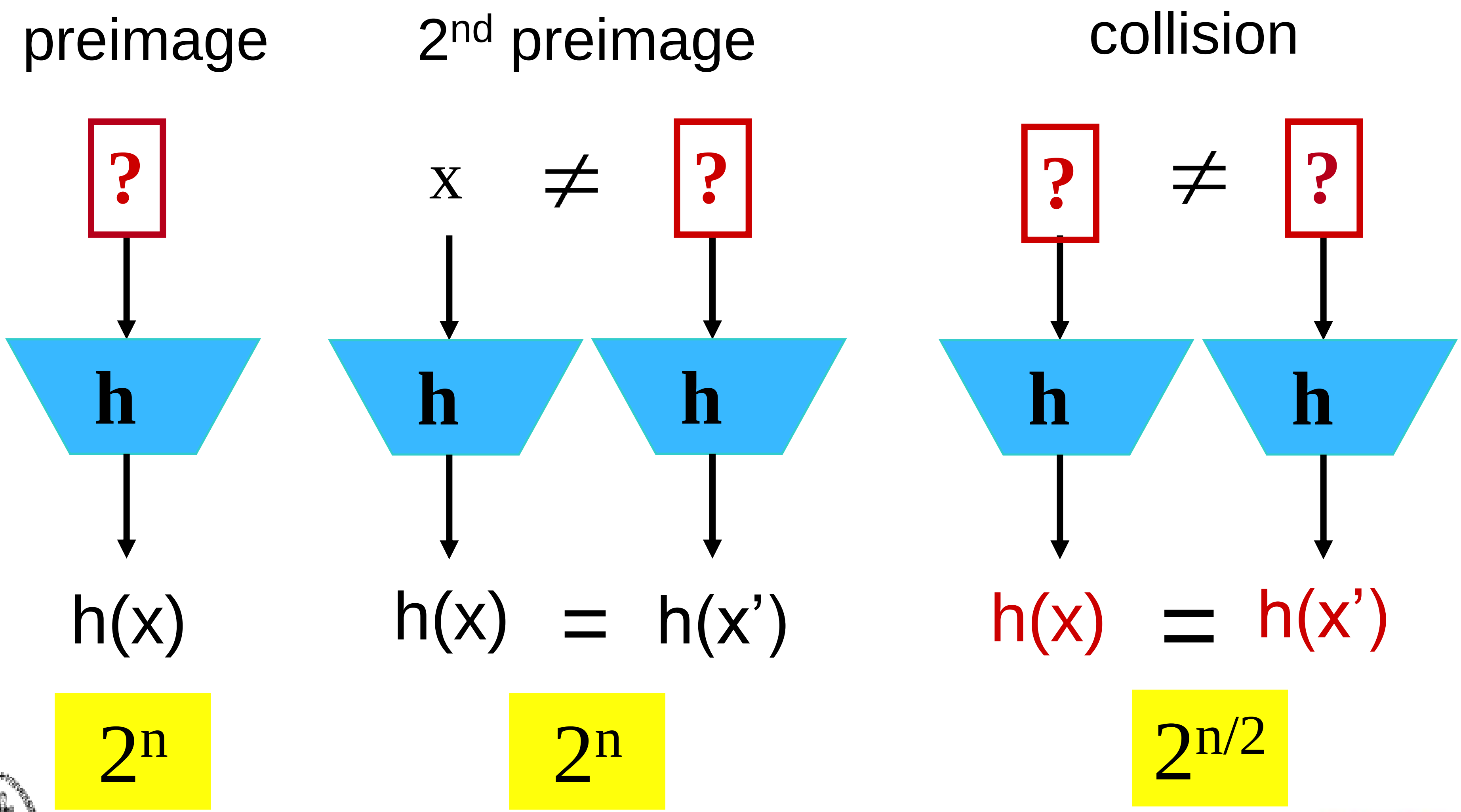
Agenda

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

Hash function flavors



Security requirements (n-bit result)



Informal definitions (1)

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

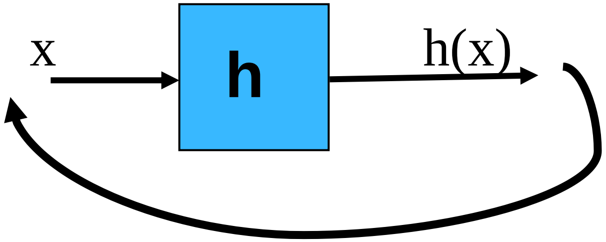
- One Way Hash Function (OWHF)
  - preimage resistance
  - 2nd preimage resistance
- Collision Resistant Hash Function (CRHF): OWHF +
  - collision resistant

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]
  - full cost per (2^{nd}) preimage from $\Theta(2^n)$ to $\Theta(2^{2n/5})$ [Wiener'02] (if $\Theta(2^{3n/5})$ targets are attacked)
- answer: randomize hash function: key, parameter, salt, spice,…

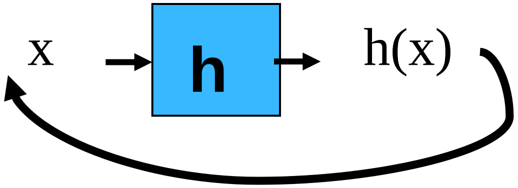
Brute force collision search

- Consider the functional graph of f

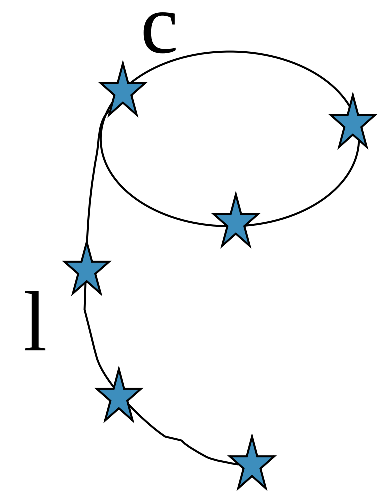


Brute force collision search

- 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

- (2^{nd}) preimage search
  - n = 128: 23 B$ for 1 year if one can attack **2^{40}** targets in parallel
- parallel collision search
  - n = 128: 1 M$ for 12 hours (or 1 year on 60K PCs)
  - n = 160: 90 M$ for 1 year
  - need 256-bit result for long term security (30 years or more)

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 formalize: requires
  - family of functions: key, parameter, salt, spice,
  - "human ignorance" trick [Stinson'06], [Rogaway'06]

Can we get rid of collision resistance?

- 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]

$$H(\, r \,||\, x_1 \oplus r \,||\, x_2 \oplus r \,||\, \ldots \,||\, x_t \oplus r \,)$$

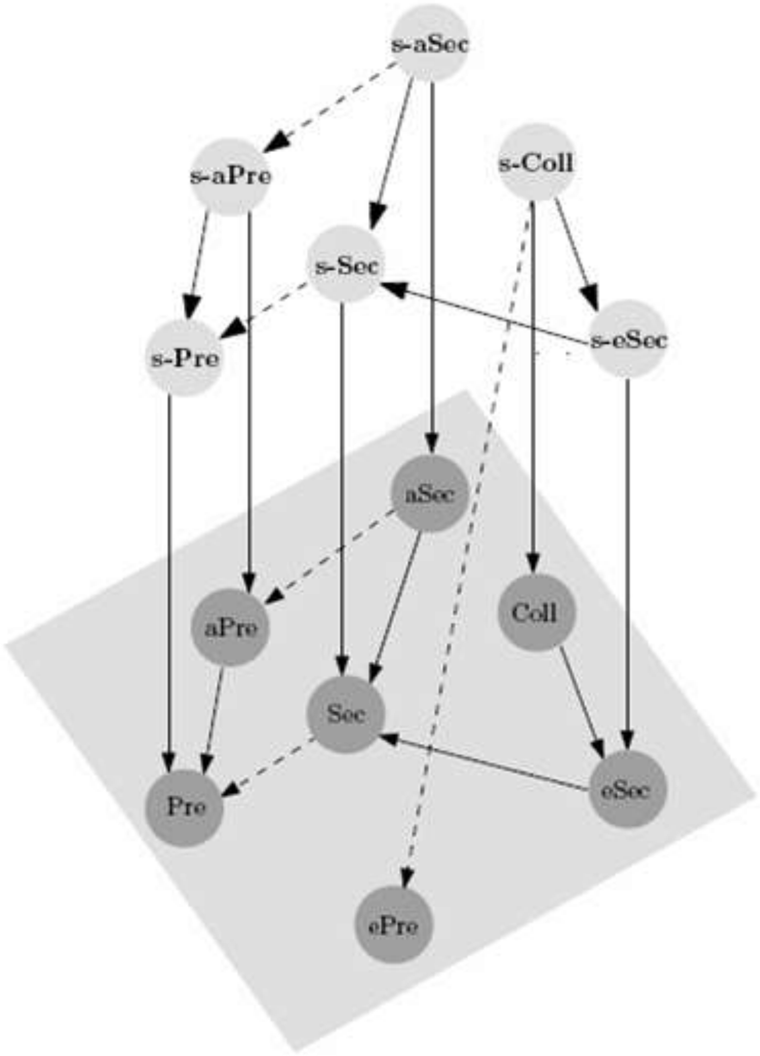
  - needs e-SPR (not met by MD5 and SHA-1 reduced to 53 rounds)
  - issues with insider attacks (i.e. attacks by the signer)

Relation between properties

[Rogaway-Shrimpton’04]

[Stinson’06]

[Reyhanitabar-Susilo-Mu’10]

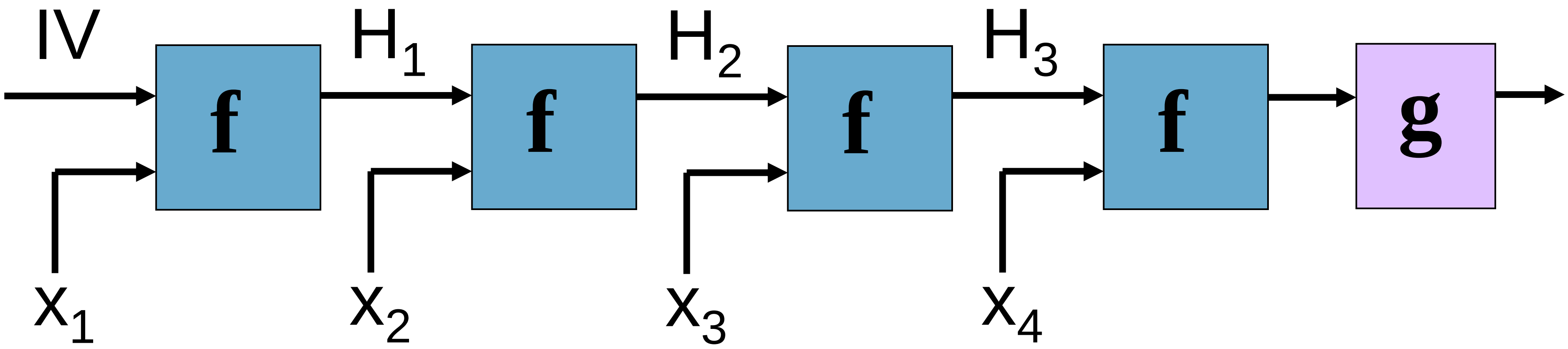


Properties in practice

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

Iteration
(mode of compression function)

Hash function: iterated structure

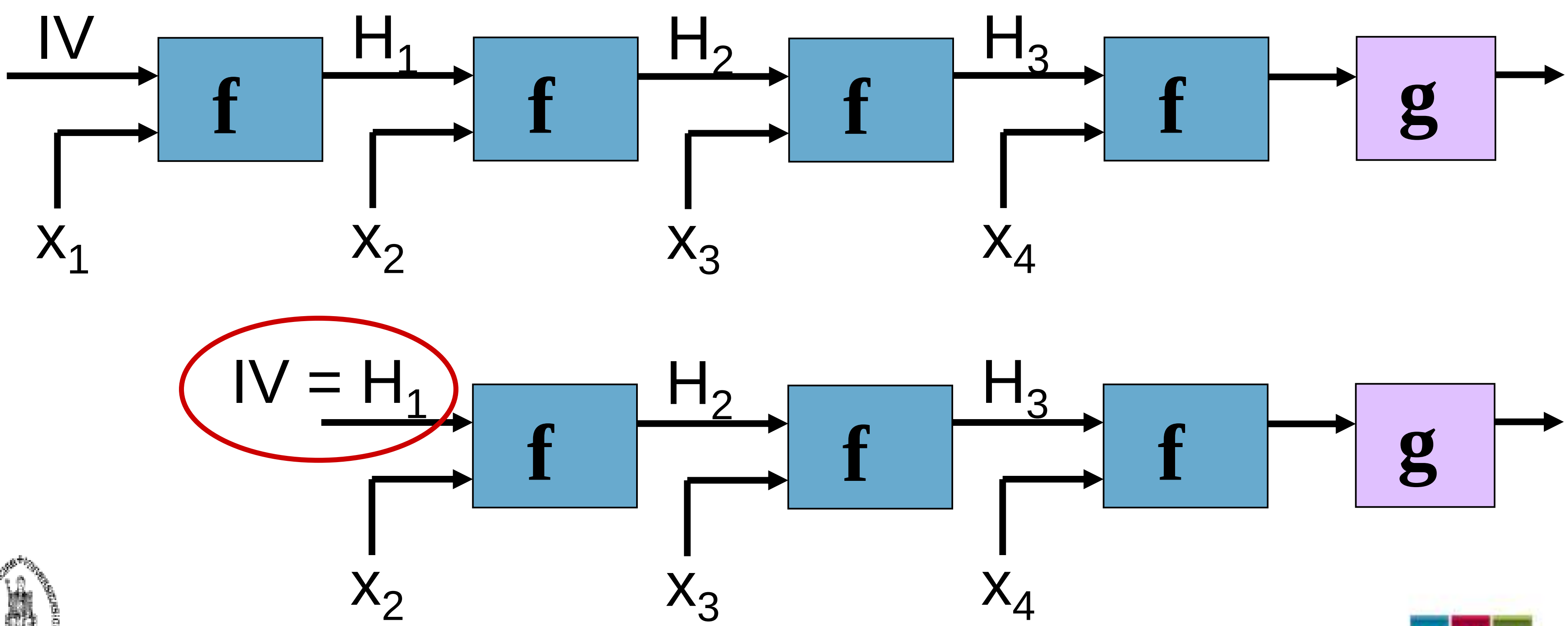


Split messages into blocks of fixed length and hash them block by block with a compression function f

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)

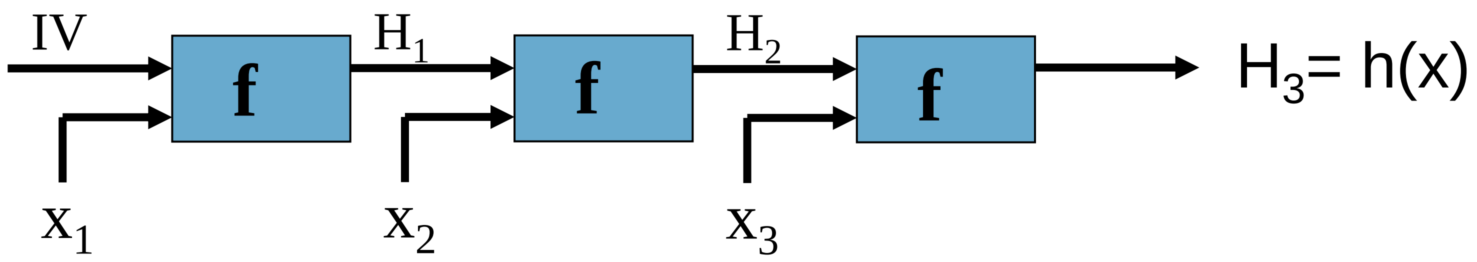
- Solution: Merkle-Damgård (MD) strengthening
  - fix IV, use unambiguous padding and insert length at the end

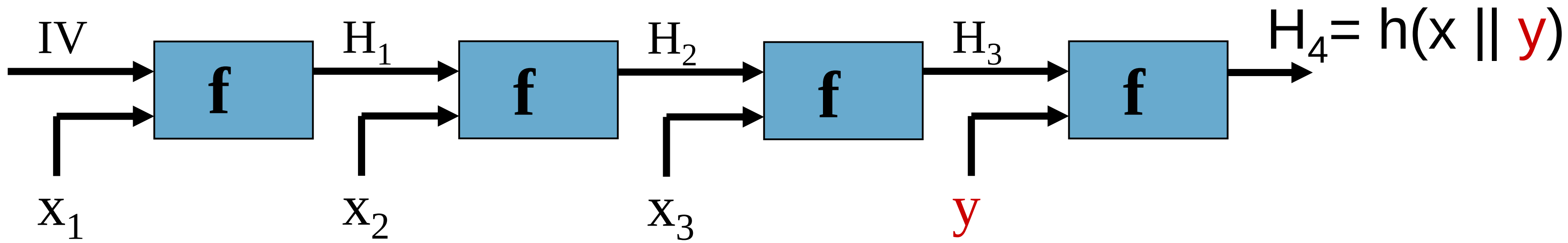
- f is collision resistant $\Rightarrow$ h is collision resistant
  [Merkle'89-Damgård'89]
- f is ideally 2nd preimage resistant $\Leftrightarrow$ h is ideally 2nd preimage resistant [Lai-Massey'92] **?**

- few hash functions have a strong compression function
- very few hash functions treat x_i and H_{i-1} in the same way

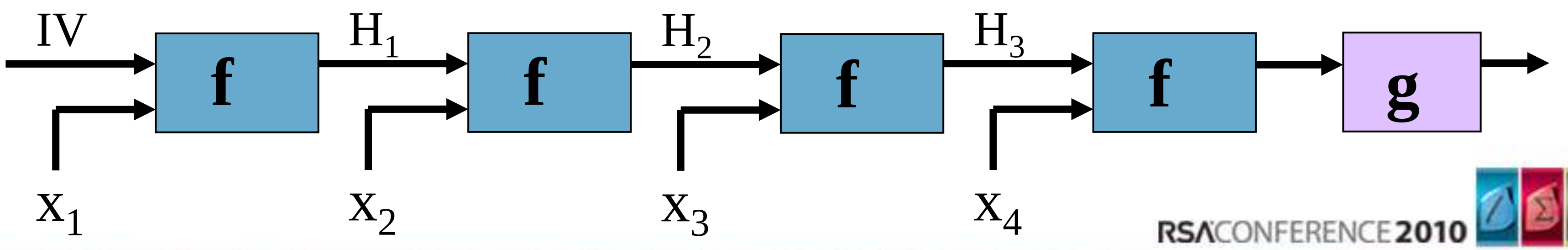
Security relation between f and h (3)

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





Solution: output transformation



Security relation between f and h (4)

- MD with output transformation preserves pseudo-random oracle (PRO) property [Coron+05]
- MD with envelope method h(K || x || K) works for pseudo-randomness/MAC [Bellare-Cannetti-Krawczyk'96]
  - but there are some problems and HMAC is a better construction
- MD preserves Preimage Awareness [Dodis-Ristenpart-Shrimpton'09]
  - Property "in between" CR (collision resistance) and PRO
- MD does not work for UOWHF [Bellare-Rogaway'97]

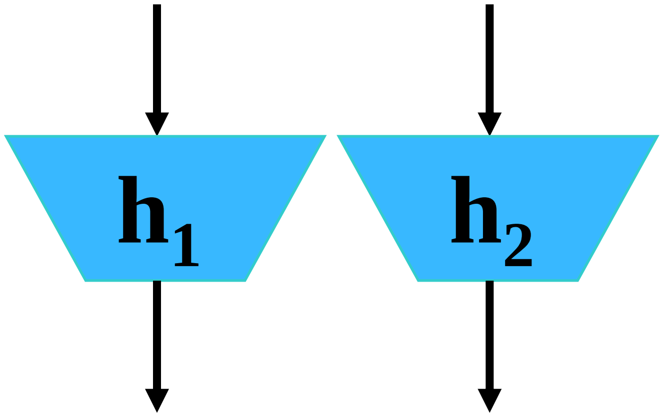
Attacks on MD: 1999-2006

- 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)
  - cost of collision attack against g at most $n1 . 2^{n2/2} + 2^{n1/2} << 2^{(n1 + n2)/2}$
- long message 2^{nd} 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 $2^{n-t+1} + t\ 2^{n/2+1}$
  - appending the length does not help here!
- 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

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]

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

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

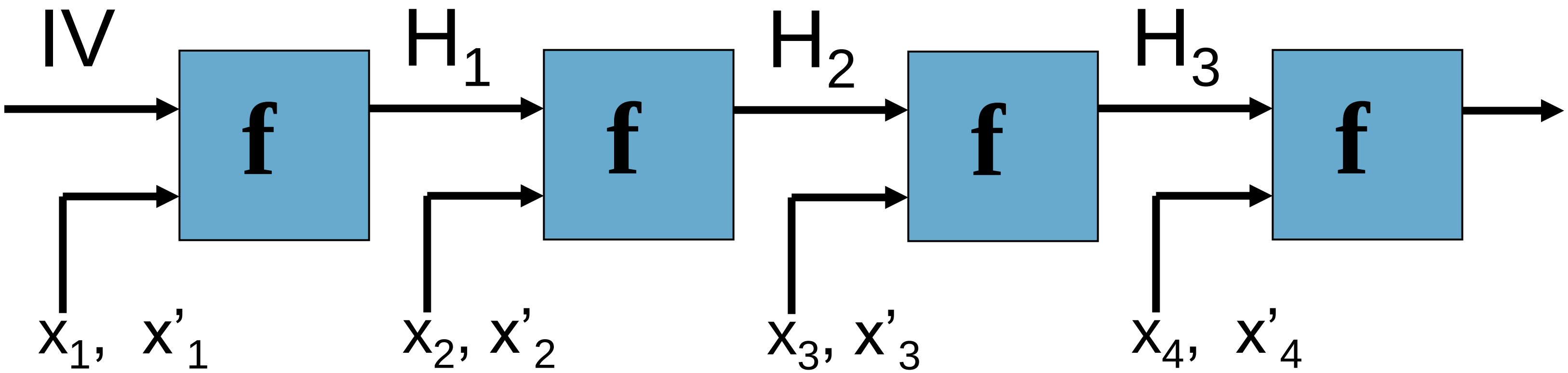
  $$n1 \cdot 2^{n2/2} + 2^{n1/2} \ll 2^{(n1 + n2)/2}$$

- Cost of (2nd) preimage attack against g at most

  $$n1 \cdot 2^{n2/2} + 2^{n1} + 2^{n2} \ll 2^{n1 + n2}$$

- 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 (if the size of the internal memory is n bits)

Multi-collisions (2) [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

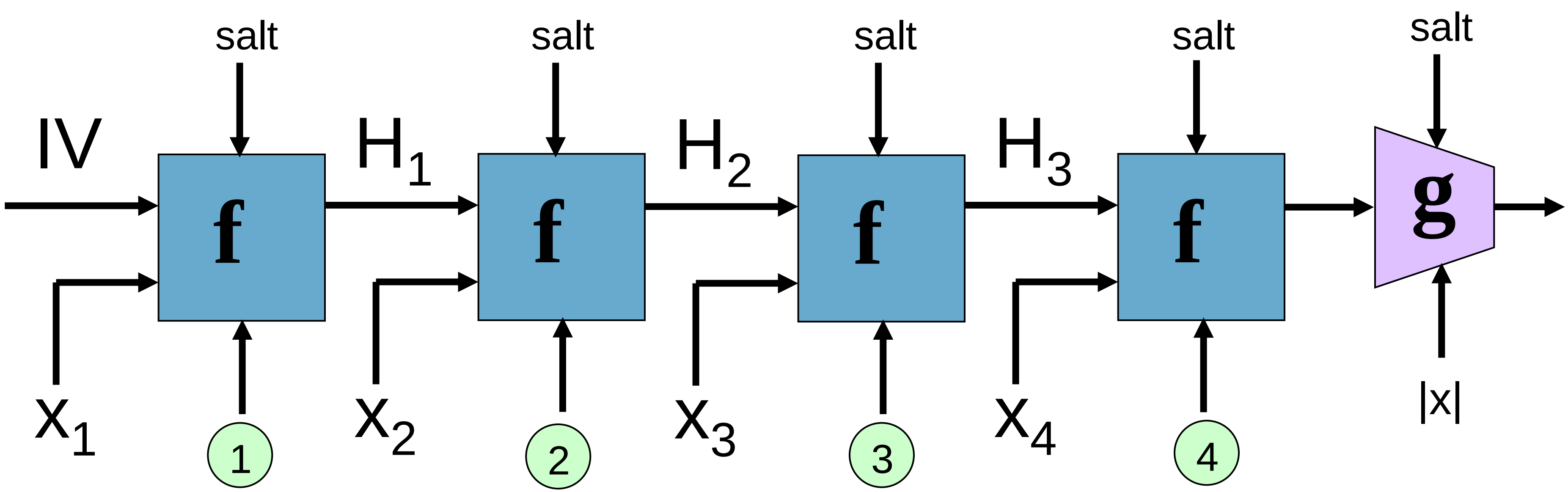
- Now $h(x_1||x_2||x_3||x_4) = h(x'_1||x_2||x_3||x_4) = h(x'_1||x'_2||x_3||x_4) = \ldots = h(x'_1||x'_2||x'_3||x'_4)$ **a 16-fold collision**

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 2^{nd} 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]

Improving MD iteration

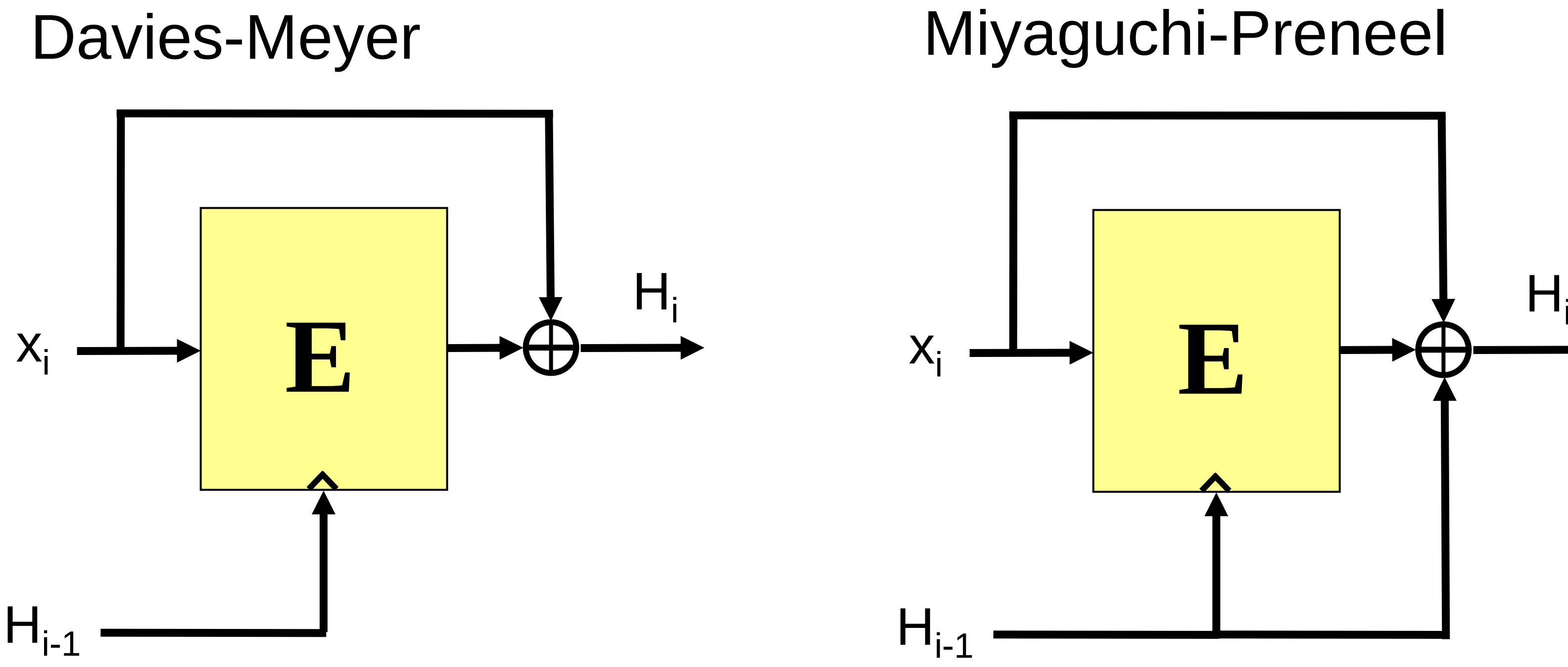
salt + output transformation + counter + wide pipe



many more results on property preservation

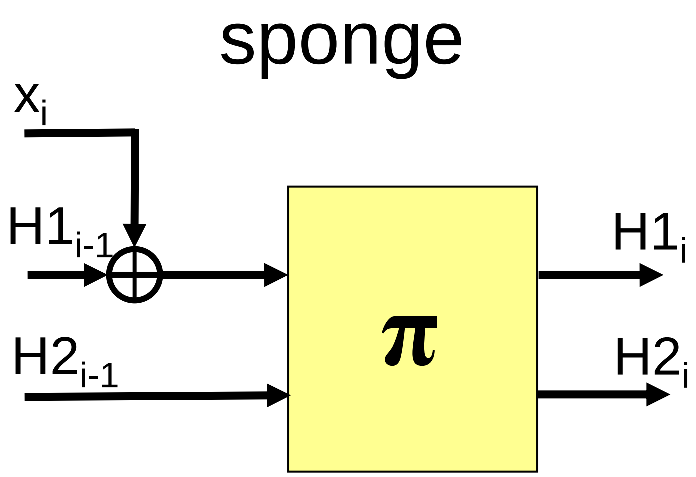
Compression functions

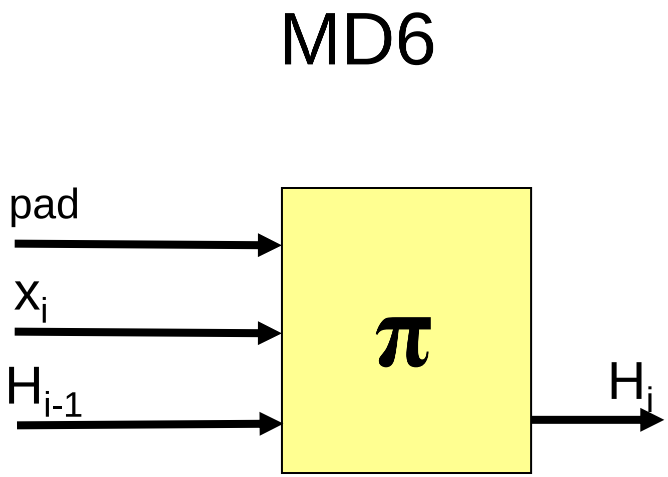
Block cipher (E_K) based



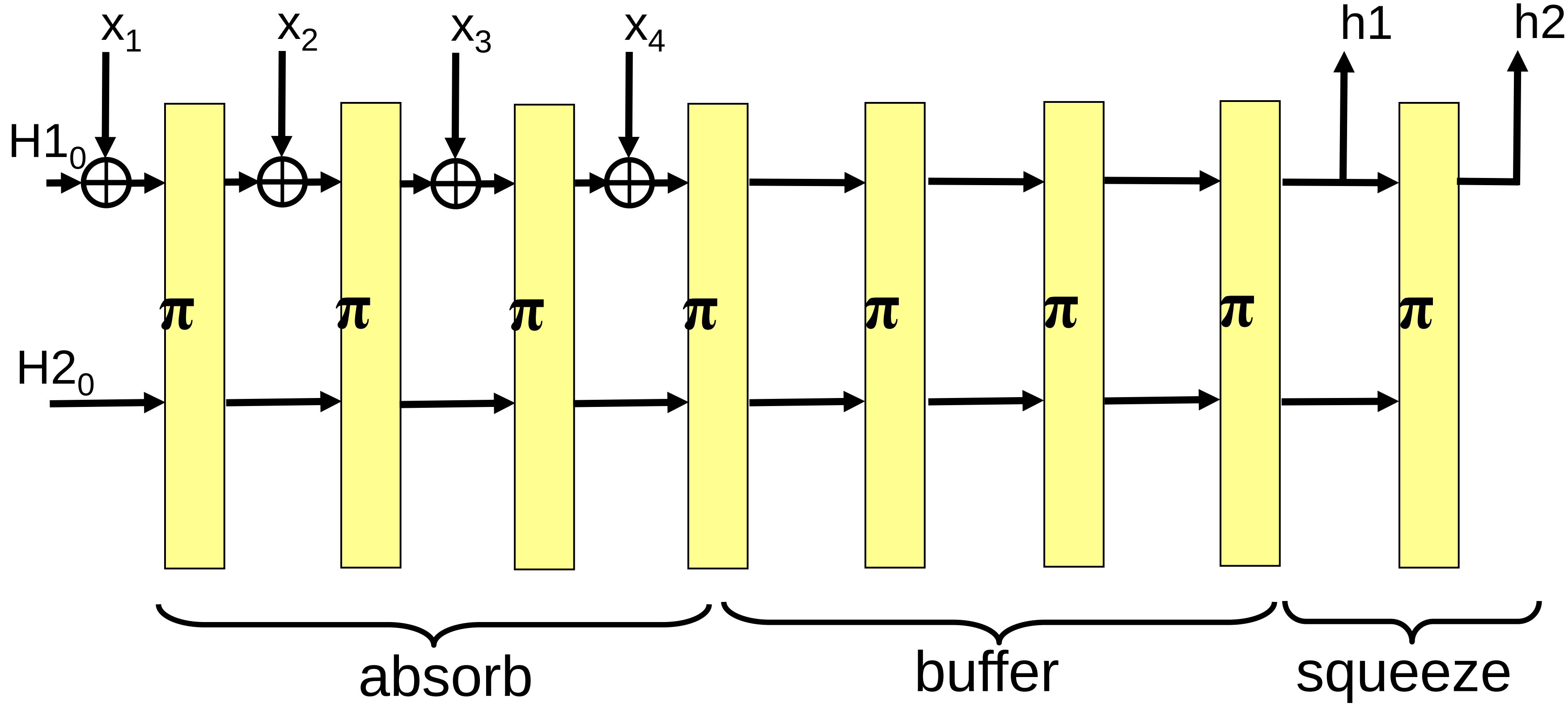
- output length = block length
- 12 secure compression functions in ideal cipher model
- requires 1 key schedule per encryption

Large permutation





Permutation (π) based: sponge

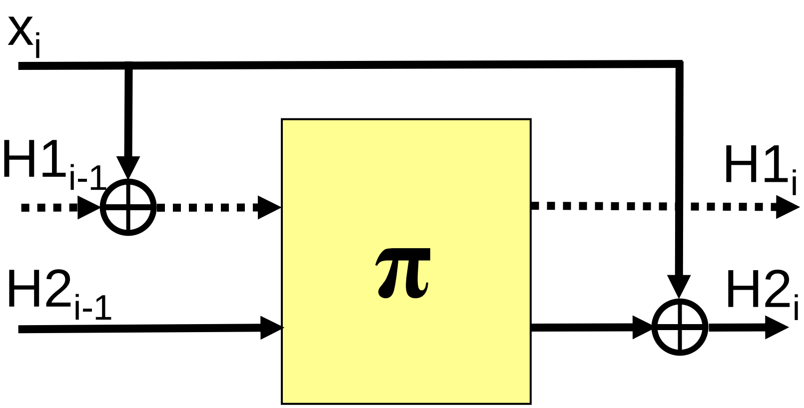


Examples: Panama, RadioGatun, Grihndahl, Keccak

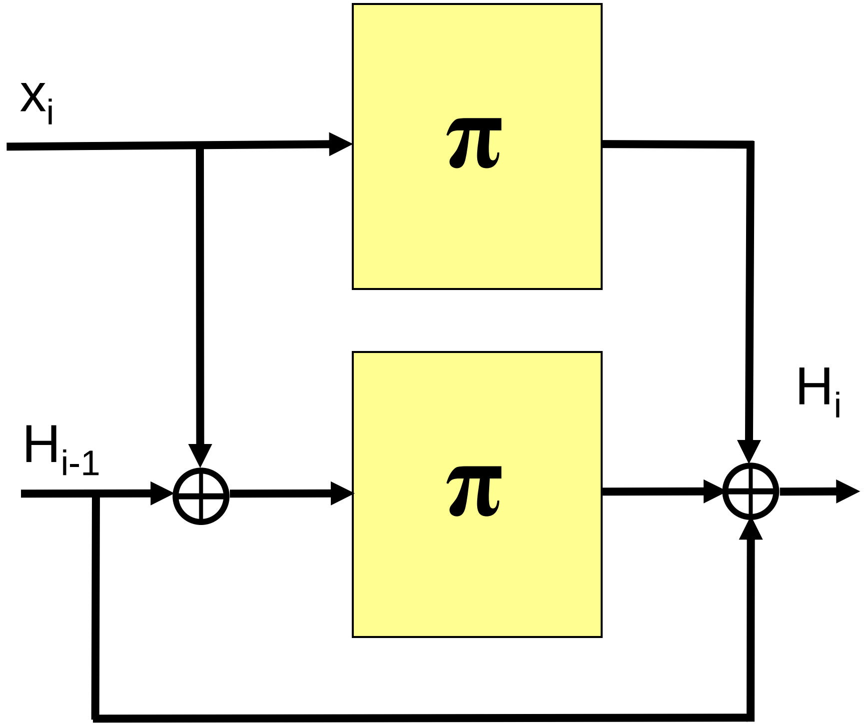
Permutation (π) based

small permutation

JH

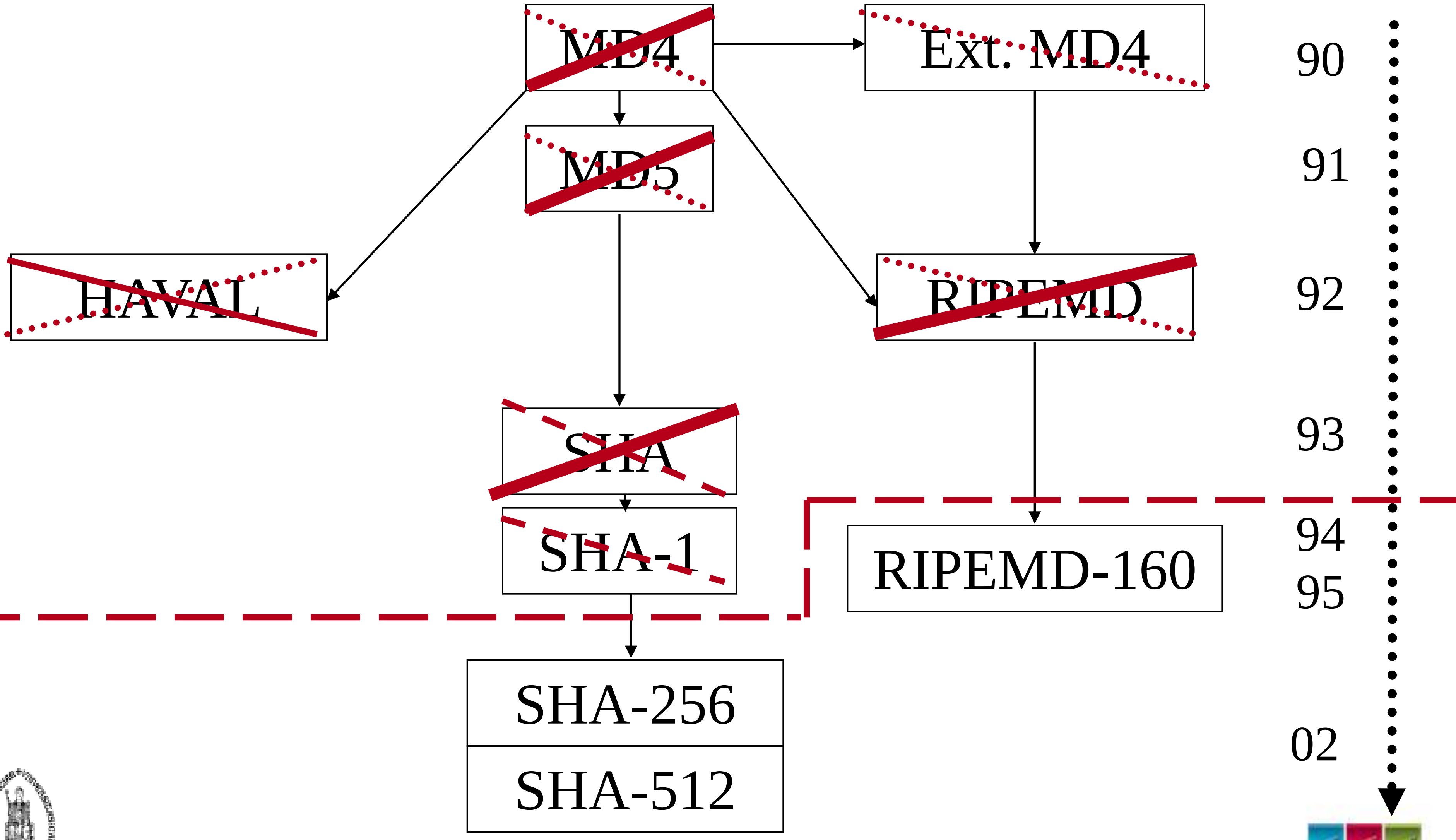


Grøstl



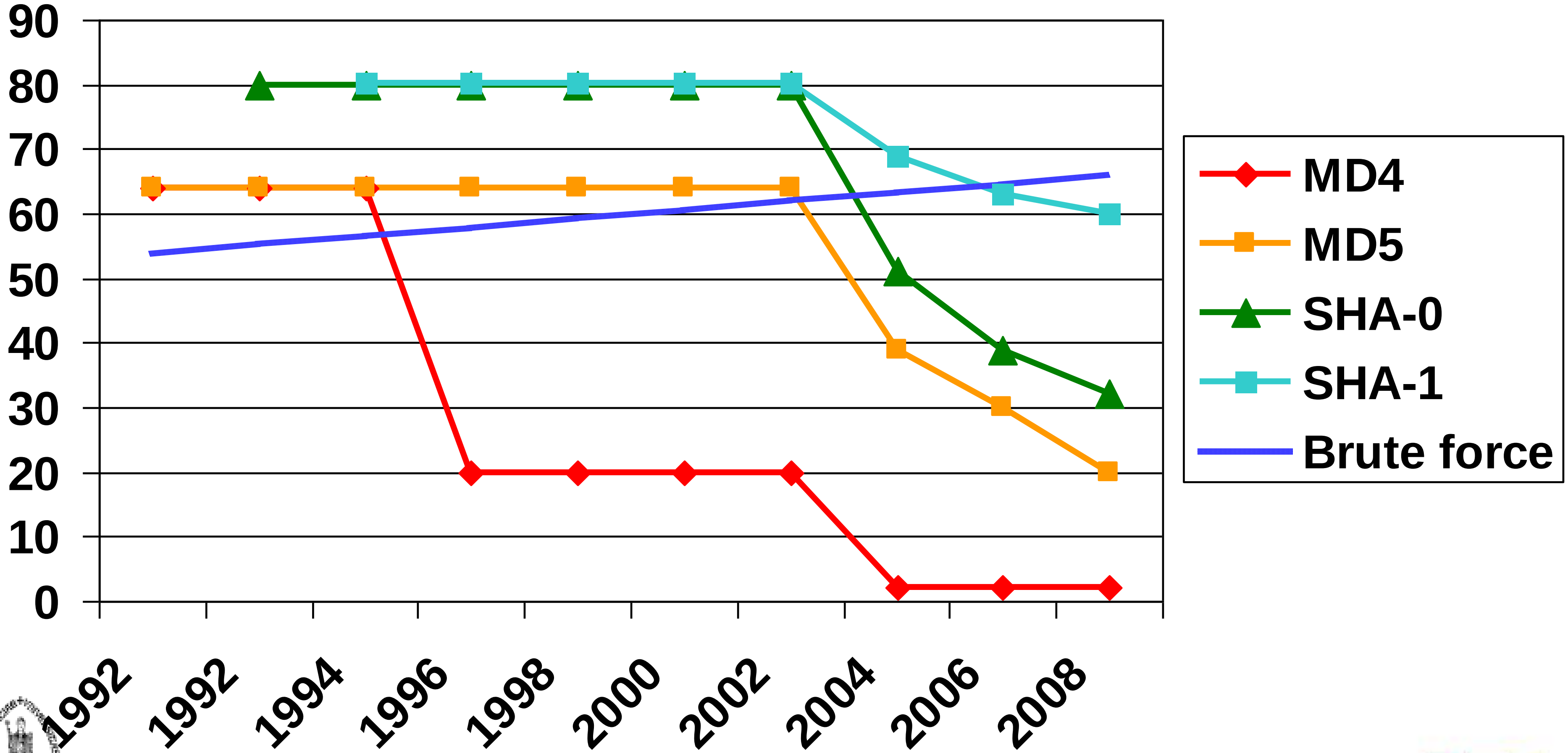
SHA-{0,1,2,3}

MDx-type hash function history



The complexity of collision attacks

Brute force: 1 million PCs or US$ 100 000 hardware



MD4 [Rivest'90]

- 3 rounds (48 steps)

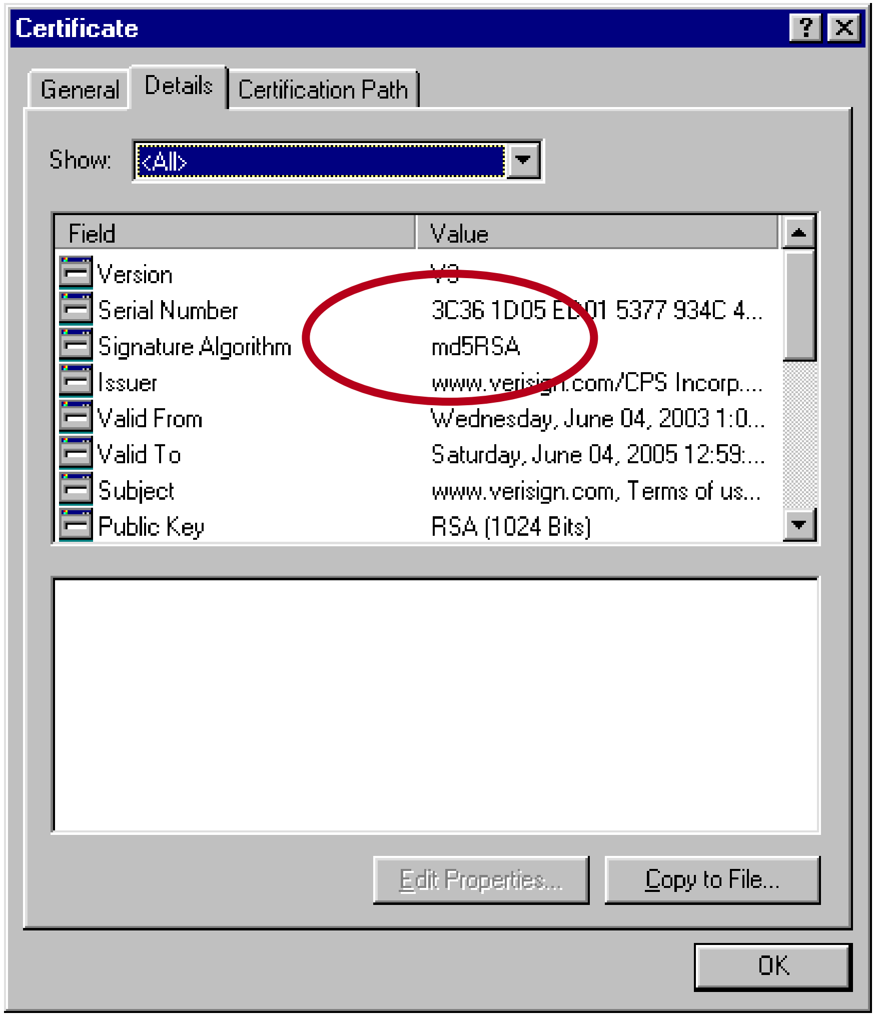
- collisions for 2 rounds [Merkle'90, denBoerBosselaers'91]
- collisions for full MD4 in 2^{20} steps [Dobbertin'96]
- (second) preimage for 2 rounds [Dobbertin'97]
- collisions for full MD4 **by hand** [Wang+'04]
- practical preimage attack for 1 in 2^{56} messages [Wang+'05]

- abandoned since 1993 (except for HMAC-MD4?)

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$ 10 hours in '09
- 2nd preimage in 2^{123} [Sasaki-Aoki'09]

- Advice (RIPE since ‘92, RSA since ‘96): stop using MD5
- Largely ignored by industry until 2009 (click on a cert...)



SHA(-0) [NIST'93]

- now called SHA-0, because of '94 of publication SHA-1
- very similar to MD5:
  - 16 extra steps (from 64 to 80)
  - message expansion uses bitwise code rather than repetition

    $w_j \leftarrow (w_{j-3} \oplus w_{j-8} \oplus w_{j-14} \oplus w_{j-16})\ \ j>15$
  - quasicyclic code with $d_{min} = 23$

- 1994: withdrawn by NIST for unidentified flaw
- 2004: collisions for in 2^{51} [Joux+'04]
- 2005: collisions in 2^{39} [Wang+'05]
- 2007: collisions in 2^{32} [Joux+'07]
- 2008: collisions in 1 hour [Manuel-Peyrin'08]
- 2008: preimages for 52 of 80 steps in $2^{156.6}$ [Aoki-Sasaki'09]

SHA-1 [NIST'95]

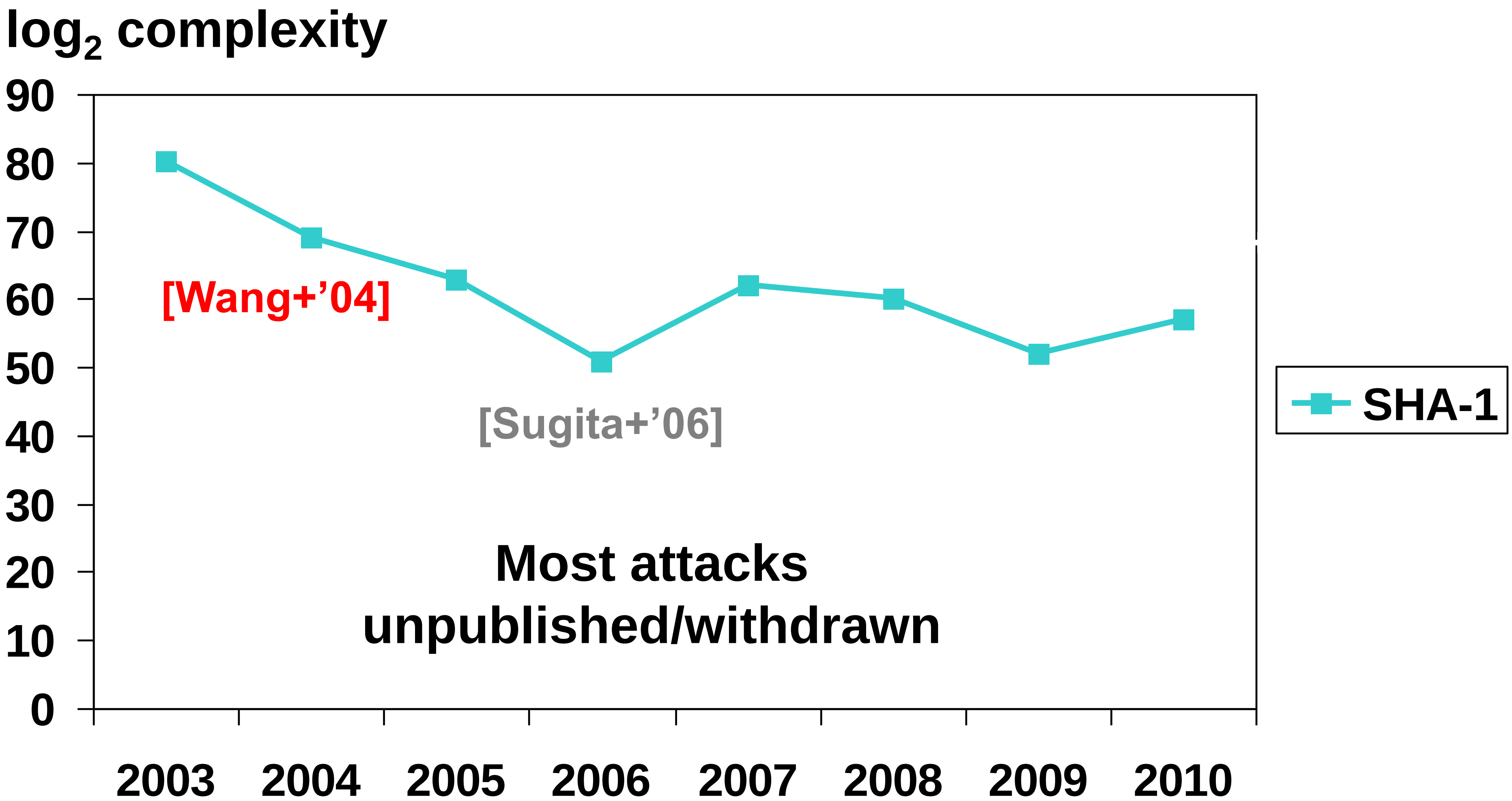
- fix to SHA-0
- add rotation to message expansion: quasicyclic code, $d_{min} = 25$
  $w_j \leftarrow (w_{j-3} \oplus w_{j-8} \oplus w_{j-14} \oplus w_{j-16}) >>> 1 \quad j > 15$

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]

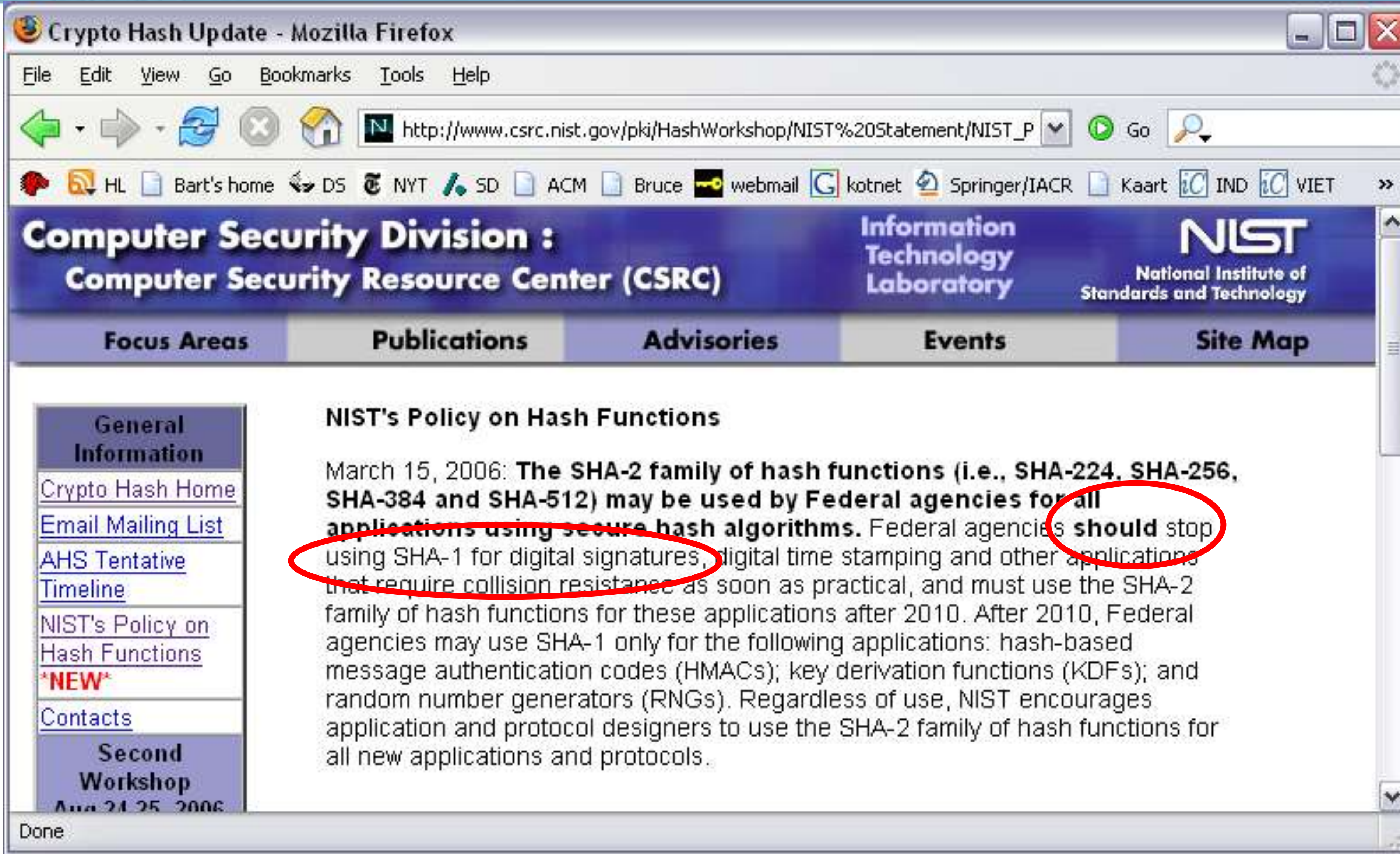
preimages for 48/80 steps in $2^{160-\varepsilon}$ [Aoki-Sasaki'09]

SHA-1



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

NIST and SHA-1

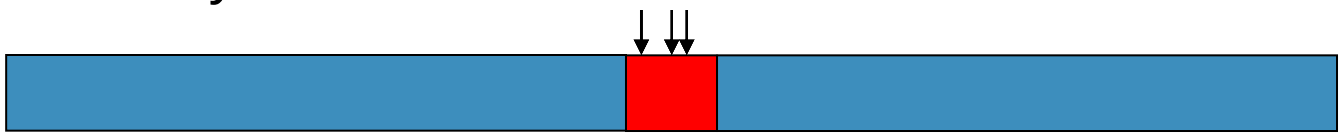


NIST's Policy on Hash Functions

March 15, 2006: **The SHA-2 family of hash functions (i.e., SHA-224, SHA-256, SHA-384 and SHA-512) may be used by Federal agencies for all applications using secure hash algorithms.** Federal agencies **should** stop using SHA-1 for digital signatures, digital time stamping and other applications that require collision resistance as soon as practical, and must use the SHA-2 family of hash functions for these applications after 2010. After 2010, Federal agencies may use SHA-1 only for the following applications: hash-based message authentication codes (HMACs); key derivation functions (KDFs); and random number generators (RNGs). Regardless of use, NIST encourages application and protocol designers to use the SHA-2 family of hash functions for all new applications and protocols.

Impact of collisions

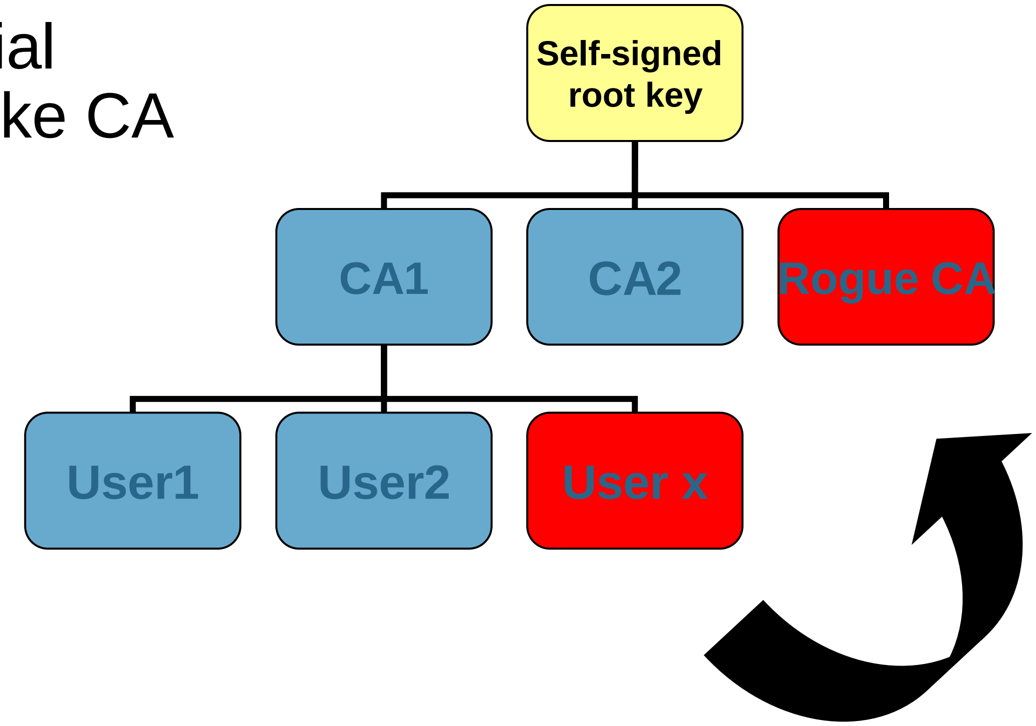
- collisions for MD5, SHA-0, SHA-1
  - 2 messages differ in a few bits in 1 to 3 512-bit input blocks
  - limited control over message bits in these blocks
  - but arbitrary choice of bits before and after them



- 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 12 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

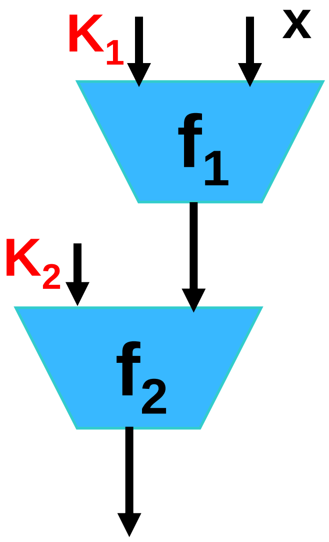
- digital signatures: only an issue if for non-repudiation
- none for signatures computed before attacks were public (1 August 2004)
- ~~none for certificates if public keys are generated at random in a controlled environment~~
- substantial for signatures after 1 August 2005 (cf. traffic tickets in 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 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



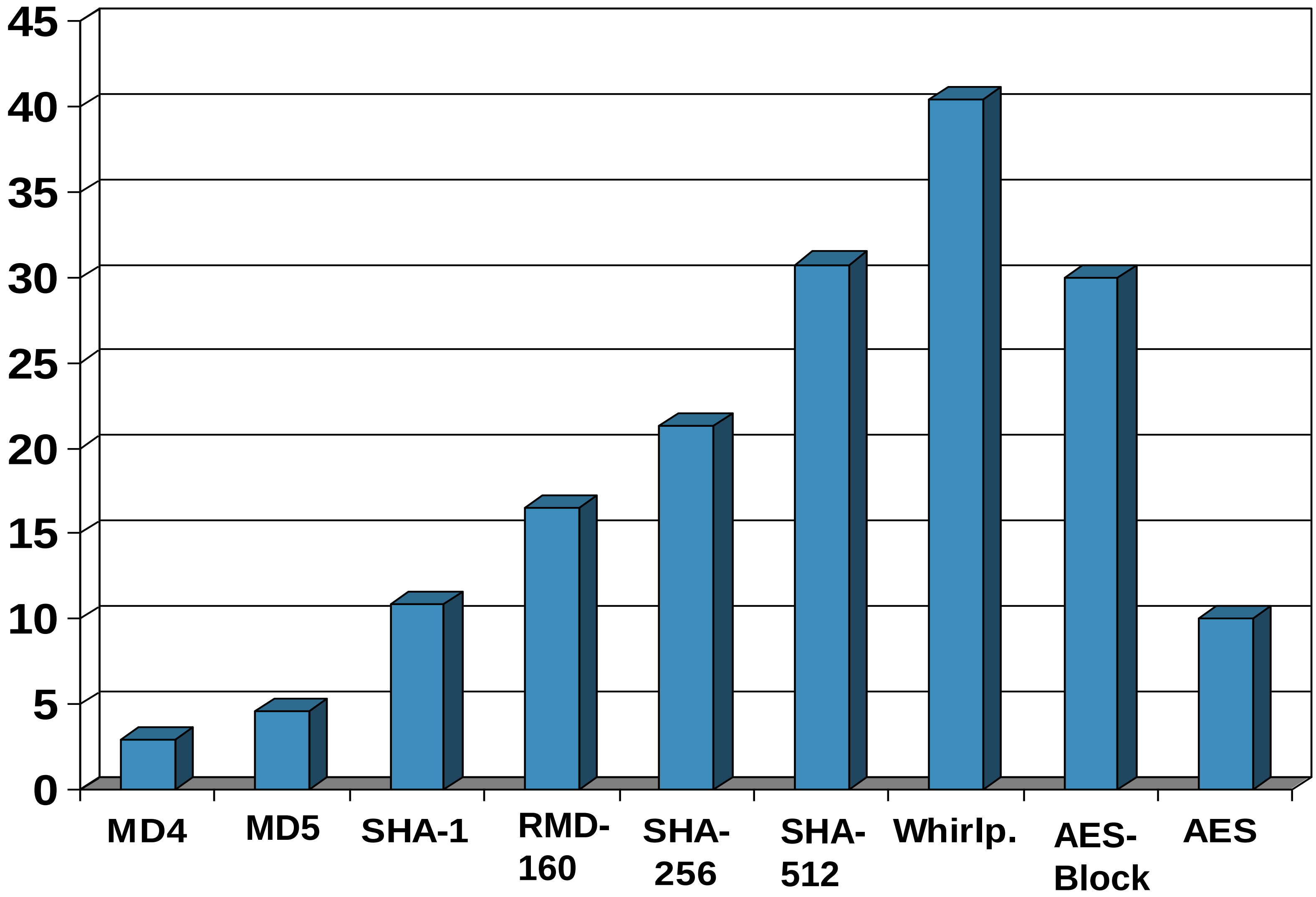
Fixes/Alternatives

- Upgrading algorithms is always hard
- TLS uses MD5 || SHA-1 to protect algorithm negotiation
- **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 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)

Performance of hash functions - Bernstein
(cycles/byte) AMD Intel Pentium D 2992 MHz (f64)



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 $\mathbf{2^{64}}$ bits

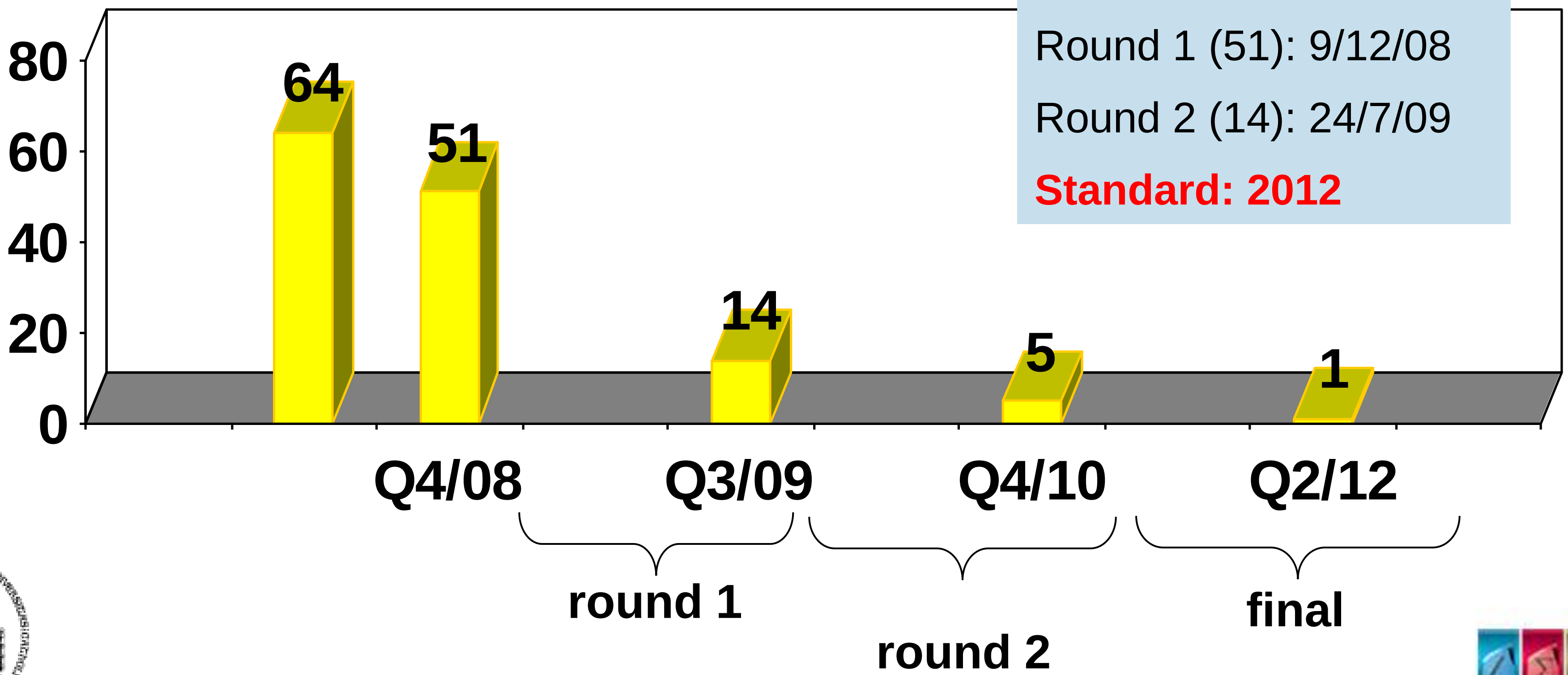
Call: 02/11/07

Deadline (64): 31/10/08

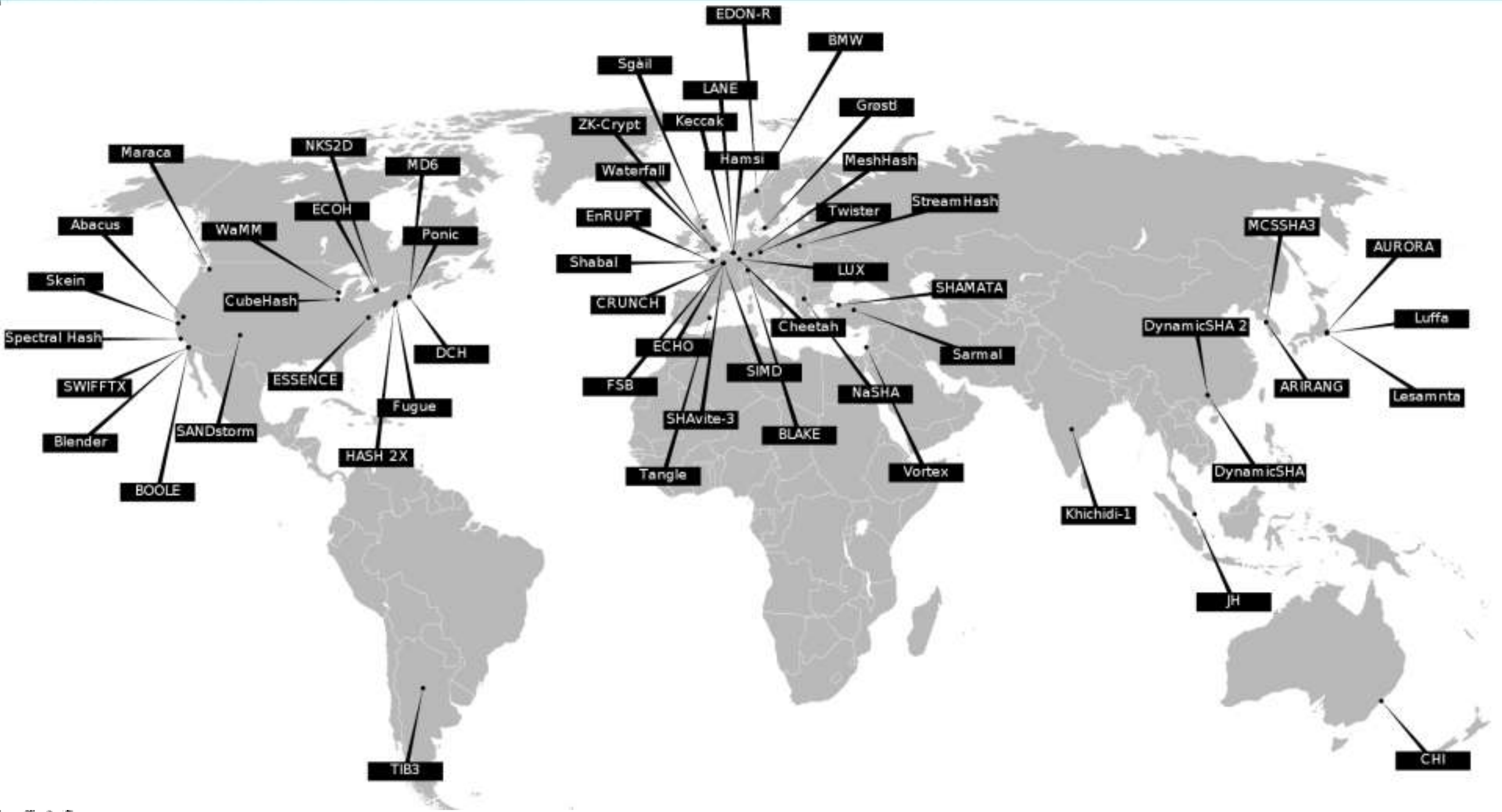
Round 1 (51): 9/12/08

Round 2 (14): 24/7/09

Standard: 2012

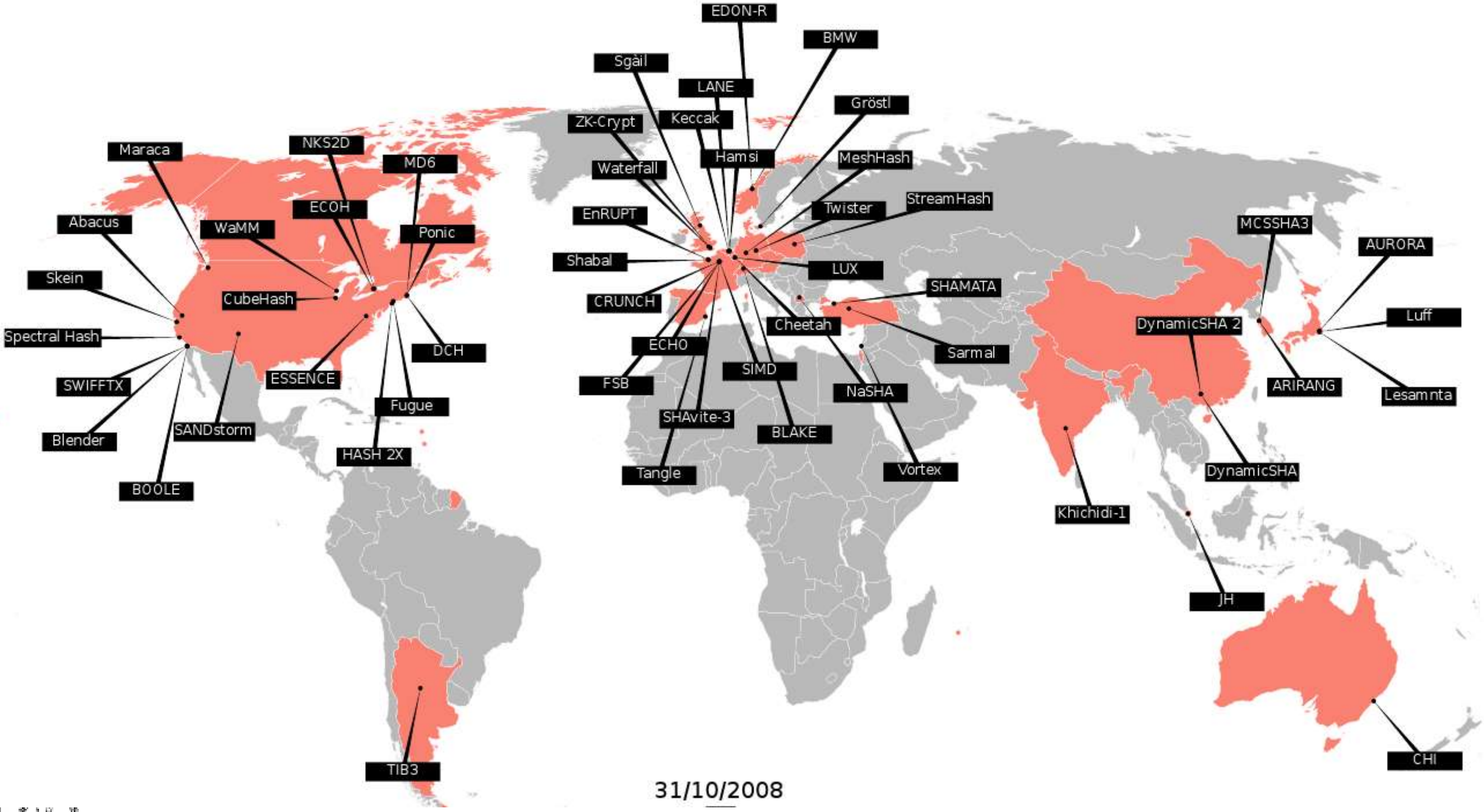


The Candidates



Slide credit: Christophe De Cannière

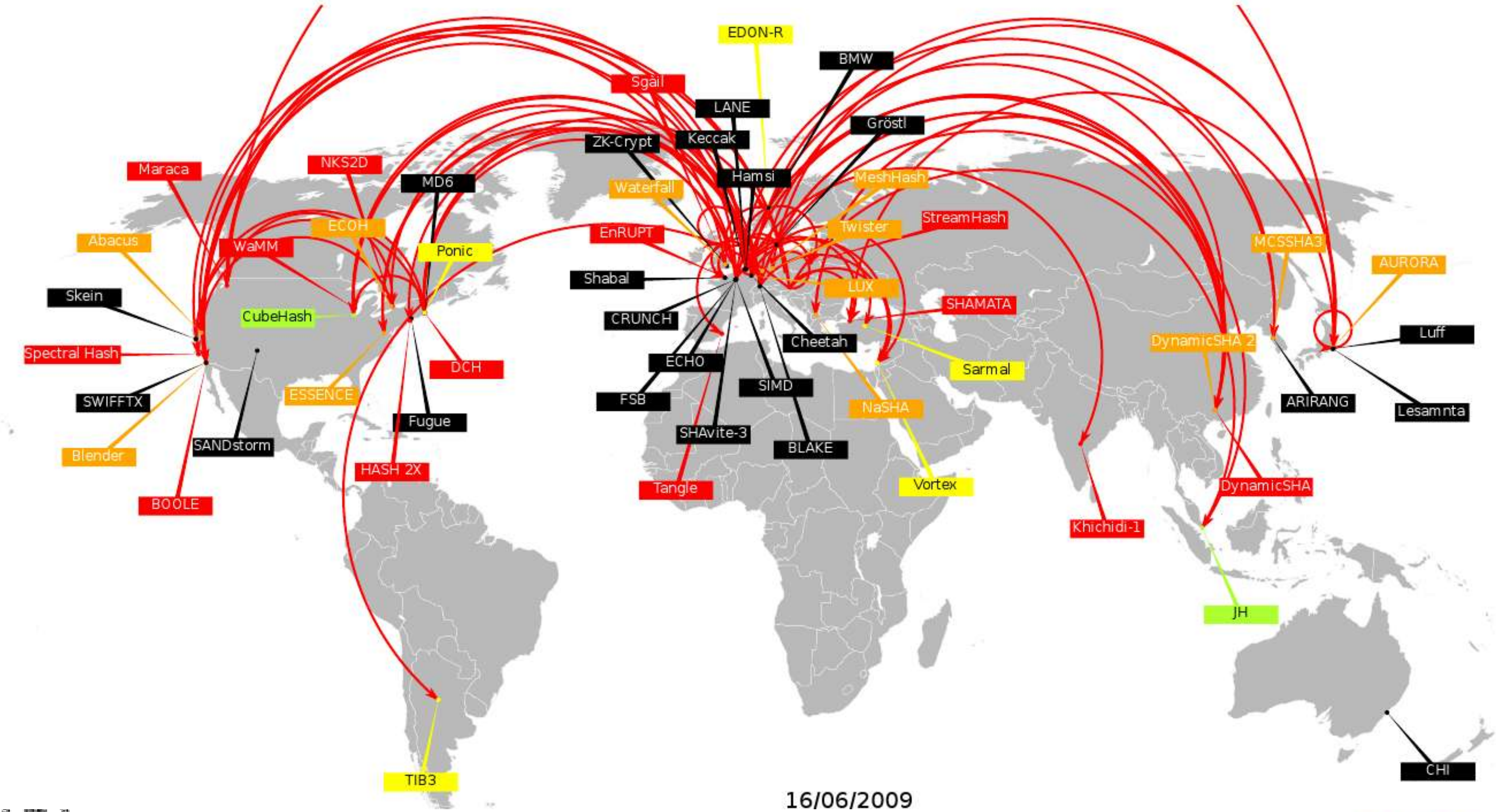
The Candidates



RSACONFERENCE2010

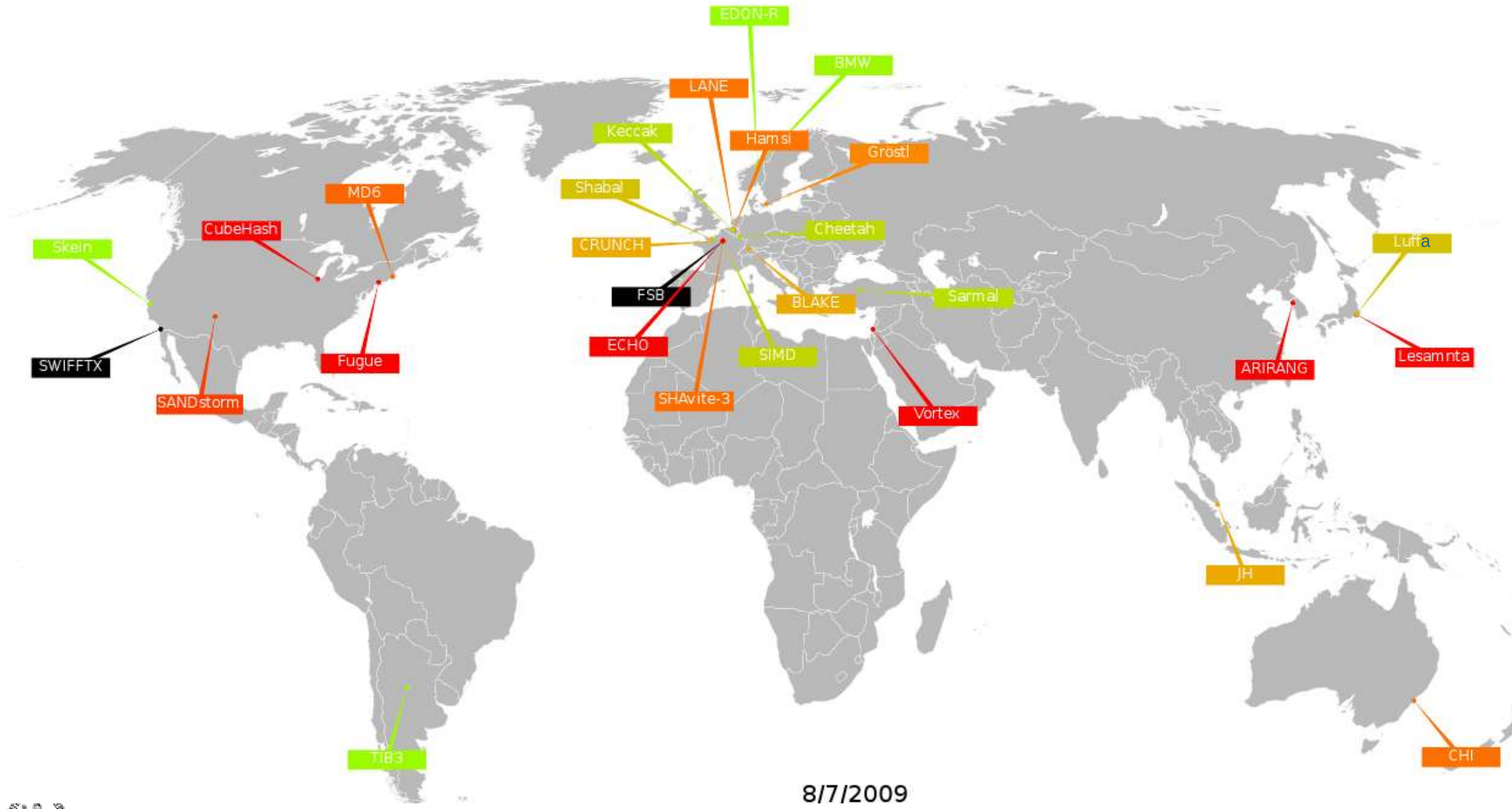
Slide credit: Christophe De Cannière

Preliminary Cryptanalysis



Slide credit: Christophe De Cannière

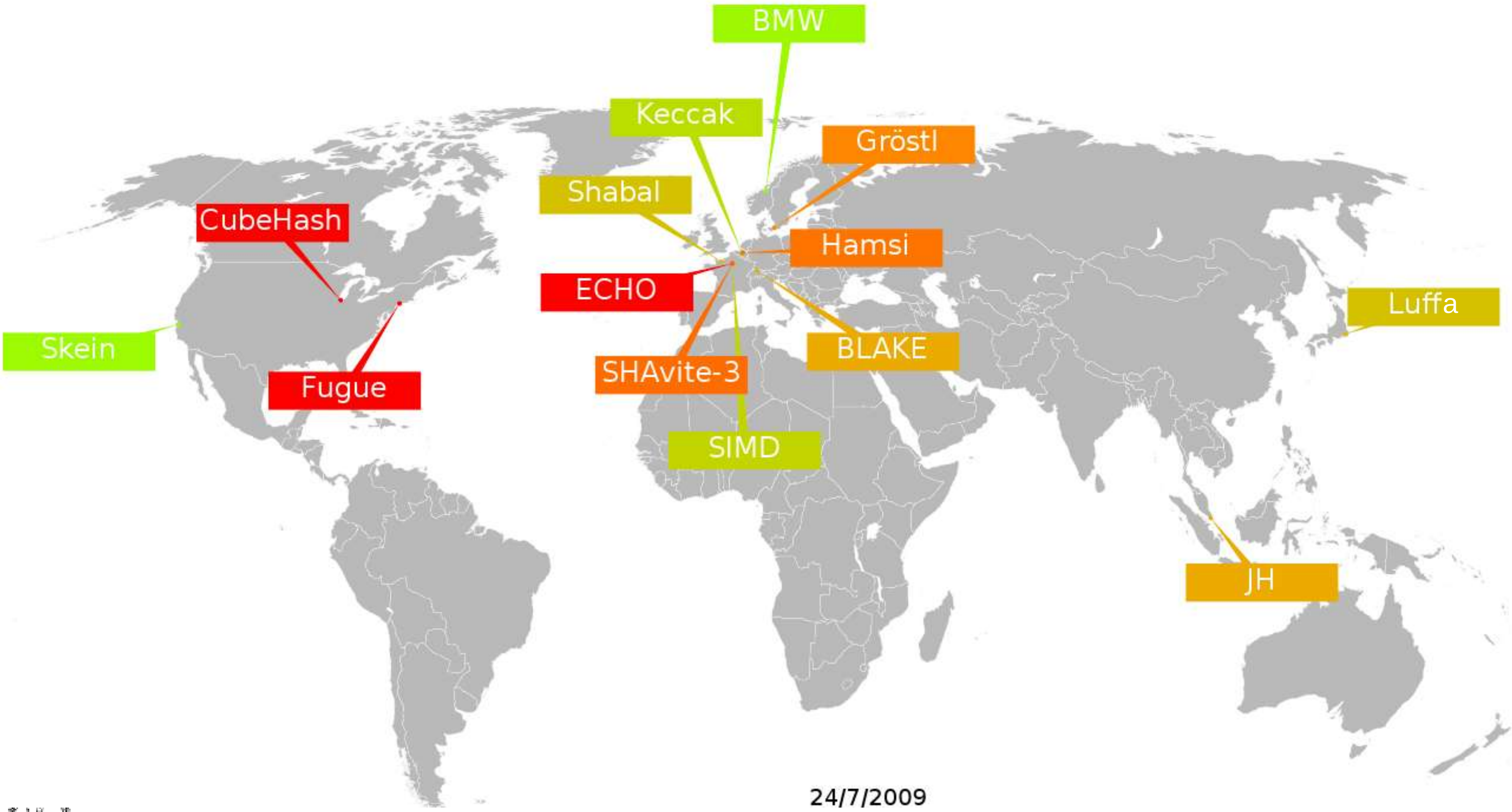
End of Round 1 Candidates



RSA CONFERENCE 2010

Slide credit: Christophe De Cannière

Round 2 Candidates



RSA CONFERENCE 2010

Slide credit: Christophe De Cannière

Iteration modes

- Wide pipe (7): BMW, Echo, Fugue, Grøstl, JH, Keccak, Simd
  - Skein has both wide and narrow pipe
- Haifa:
  - Echo, Shavite-3
  - Variant: Skein

Compression function

- Block cipher based
  - Davies-Meyer: Shavite-3, Skein
  - Miyaguchi-Preneel variant: BMW
  - Other: Shabal
- Permutation based
  - Sponge: Hamsi, Keccak
  - Sponge variant: Luffa
  - Other: Echo, Grøstl, JH

Bits and bytes

- SPN (9)
- Balanced Feistel: JH, Shavite-3, Skein
- Unbalanced Feistel: Blake, SIMD

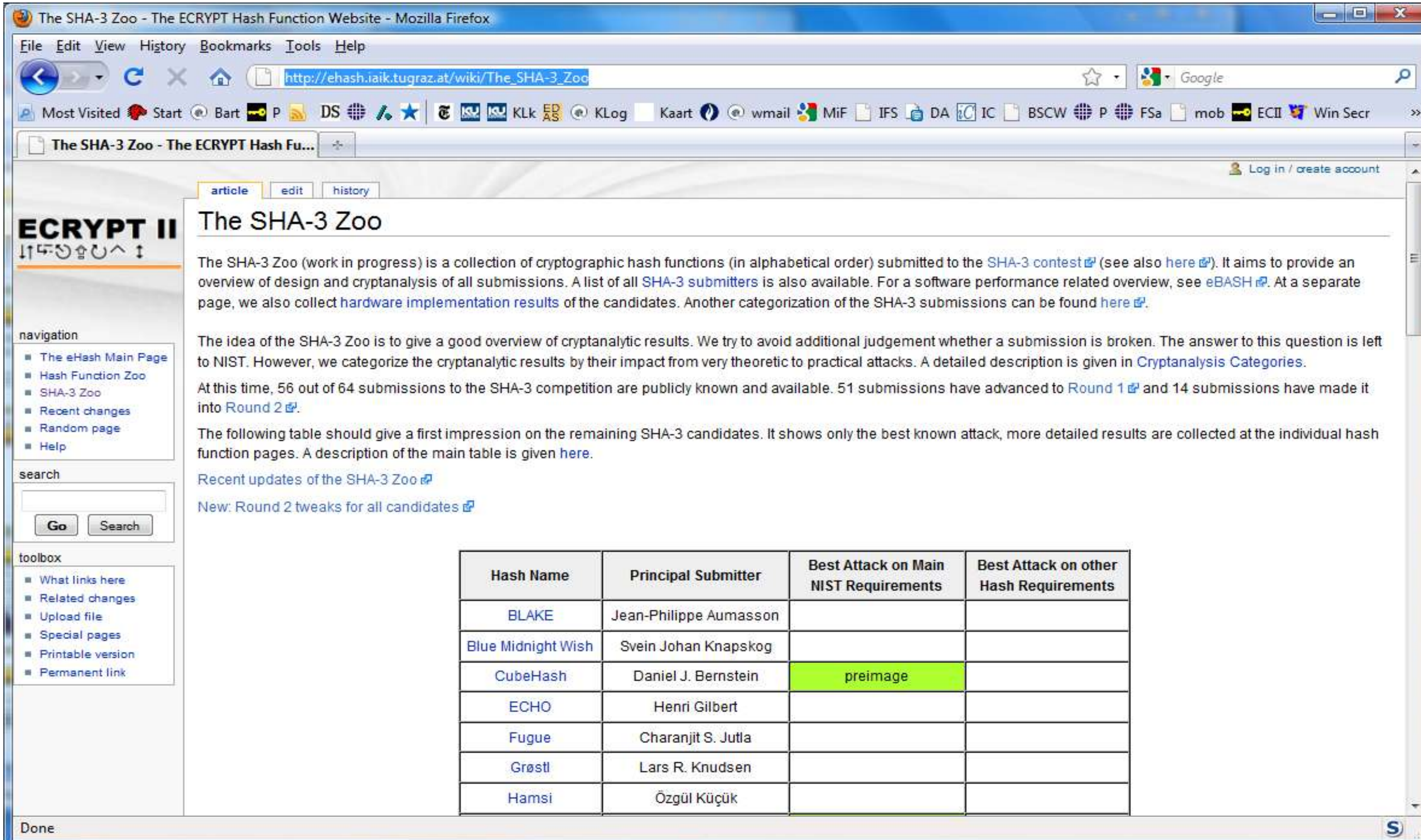
- S-boxes and diffusion (7)
  - AES-round function (8x8): ECHO, Shavite-3 (benefit from Intel AES instruction)
  - AES-inspired (8x8): Grøstl, Fugue
  - 4x4: JH, Hamsi, Luffa

- Arithmetic/logic (7)
  - ARX (addition/rotation/xor): Blake, BMW, CubeHash, Skein
  - AN (and/not): Keccac, Shabal
  - ANO (and/not/or): SIMD

Issues arisen during Round 1

- 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?
- 14 Round 2 candidates
  - most are wide-pipe designs or sponge-like designs
  - two main types: AES-based and AXR (addition/xor/rotate)

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



The SHA-3 Zoo

The SHA-3 Zoo (work in progress) is a collection of cryptographic hash functions (in alphabetical order) submitted to the SHA-3 contest (see also here). It aims to provide an overview of design and cryptanalysis of all submissions. A list of all SHA-3 submitters is also available. For a software performance related overview, see eBASH. At a separate page, we also collect hardware implementation results of the candidates. Another categorization of the SHA-3 submissions can be found here.

The idea of the SHA-3 Zoo is to give a good overview of cryptanalytic results. We try to avoid additional judgement whether a submission is broken. The answer to this question is left to NIST. However, we categorize the cryptanalytic results by their impact from very theoretic to practical attacks. A detailed description is given in Cryptanalysis Categories.

At this time, 56 out of 64 submissions to the SHA-3 competition are publicly known and available. 51 submissions have advanced to Round 1 and 14 submissions have made it into Round 2.

The following table should give a first impression on the remaining SHA-3 candidates. It shows only the best known attack, more detailed results are collected at the individual hash function pages. A description of the main table is given here.

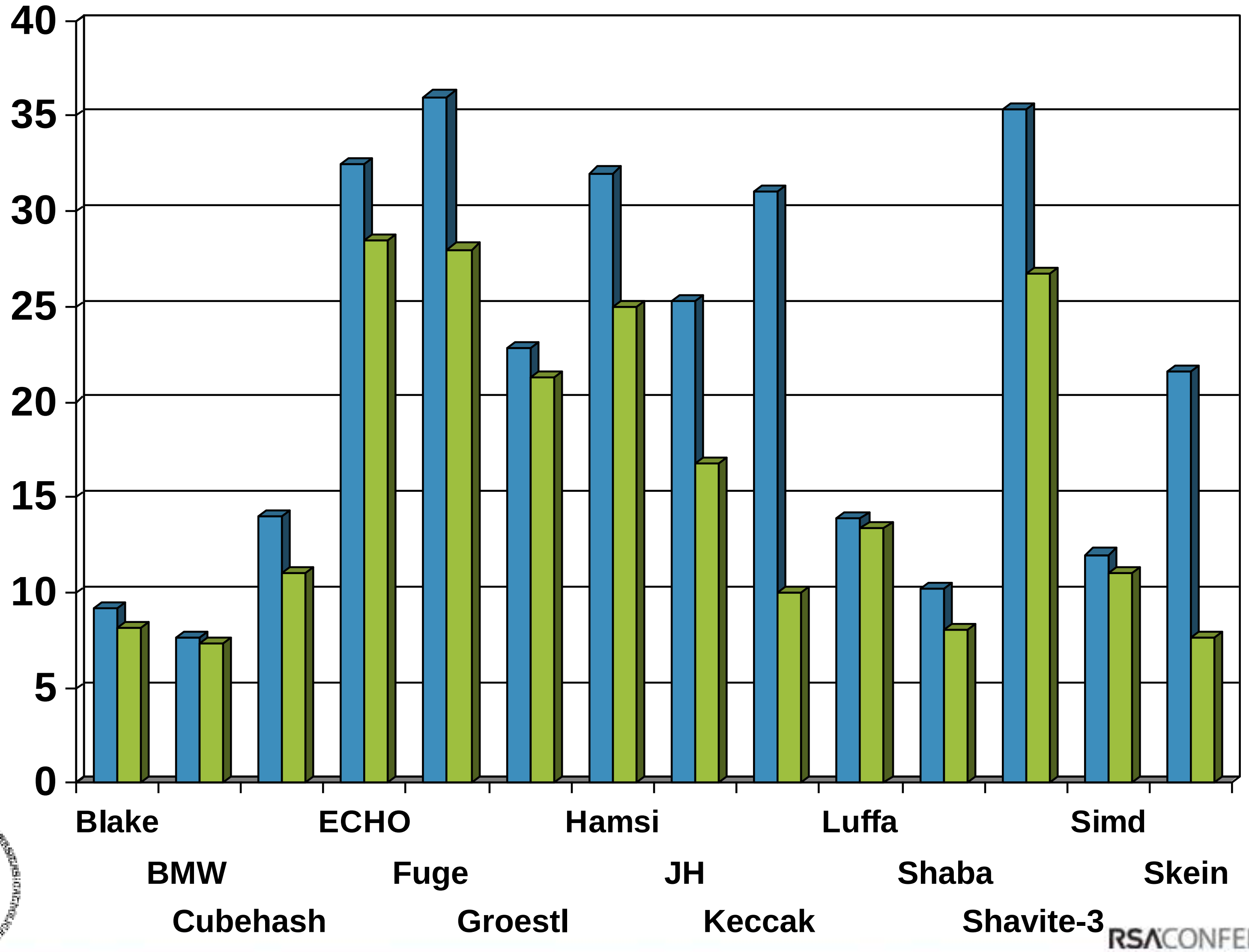
Recent updates of the SHA-3 Zoo

New: Round 2 tweaks for all candidates

Hash Name	Principal Submitter	Best Attack on Main NIST Requirements	Best Attack on other Hash Requirements
BLAKE	Jean-Philippe Aumasson		
Blue Midnight Wish	Svein Johan Knapskog		
CubeHash	Daniel J. Bernstein	preimage	
ECHO	Henri Gilbert		
Fugue	Charanjit S. Jutla		
Grøstl	Lars R. Knudsen		
Hamsi	Özgül Küçük		

Performance of hash functions

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



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

SHA-4

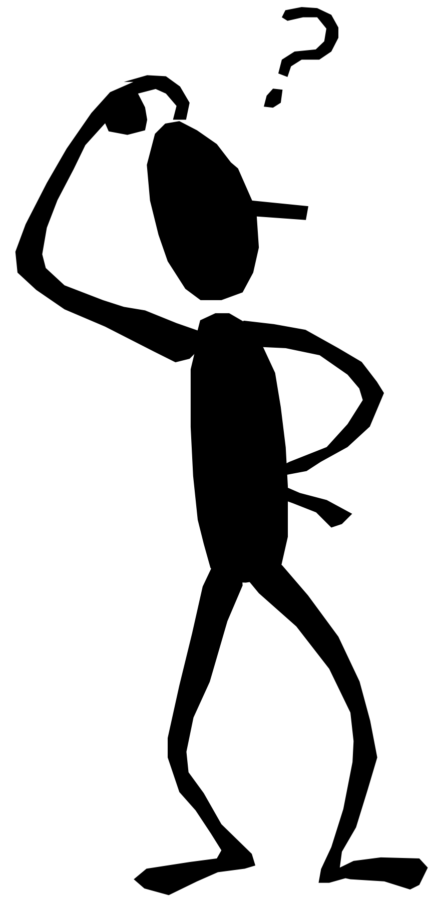
- an open competition such as SHA-3 is bound to result in new insights between 2009-2012
- only few of these can be incorporated using “tweaks”
- the winner selected in 2012 will reflect the state of the art in October 2008
- nevertheless, it is unlikely that we will have a SHA-4 competition before 2030

Hash functions: conclusions

- SHA-1 would have needed 128-160 steps instead of 80
- recent 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

The end

Thank you for your attention

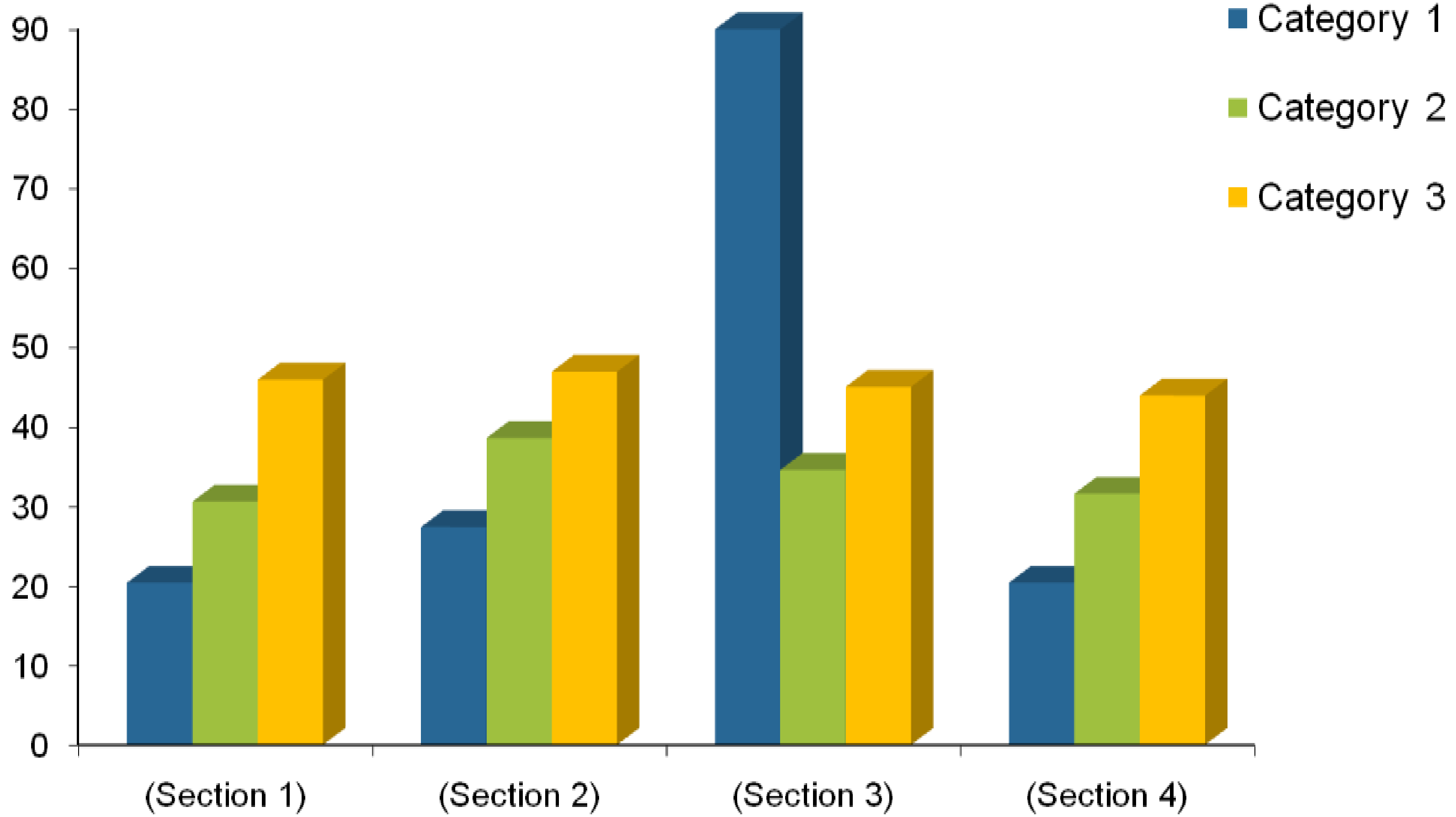


Your Headline Here (Title Caps)

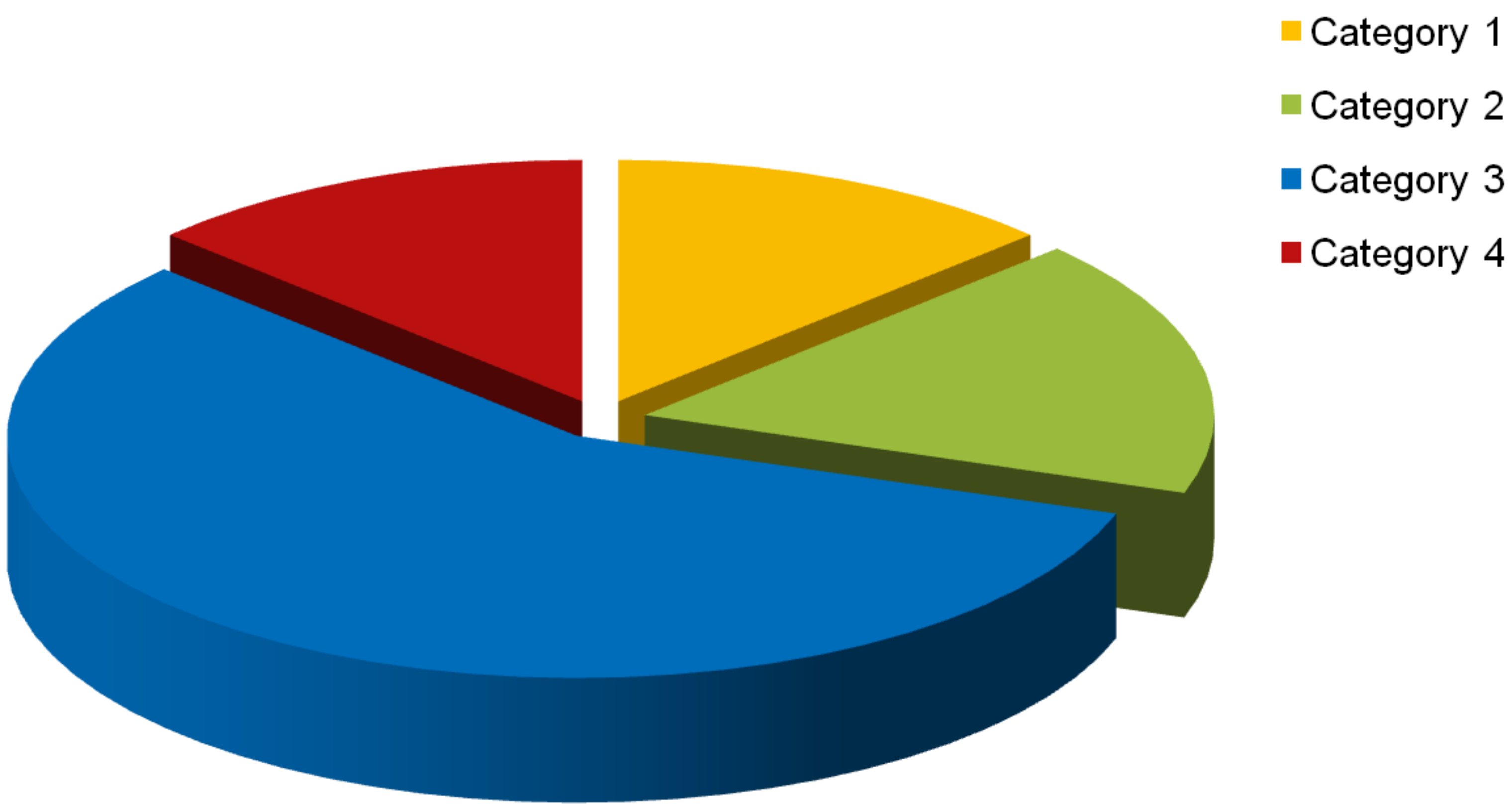
- Your talking point bullet text here
- Your next bullet point talking text here
- Third talking point, etc.

Your Headline Here (Title Caps)

Your Headline Here (Title Caps)



Your Headline Here (Title Caps)



Useful Art – Copy, paste, and resize as needed

Arrows are semi-transparent and can be placed on top of other objects

Type text in here

Type text in here

Type text in here

Type text in here

Type text in here

Type text in here

Divider Slide

(section one title here, and so on)