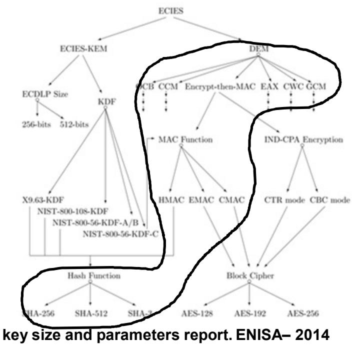
Cryptography
Lecture 3

Dr. Panagiotis Rizomiliotis
Assistant Professor
Dep. Of Informatics and Telematics
Harokopio University of Athens

Agenda

- Hash functions

OVERVIEW

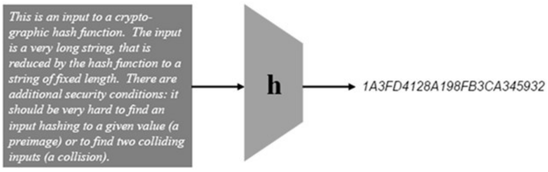


*** Algorithms, key size and parameters report. ENISA– 2014**

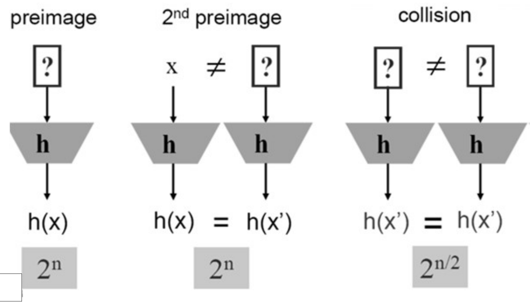
Hash functions

HASH FUNCTIONS

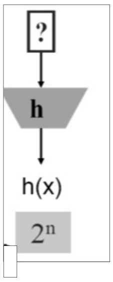
- ✓ no secret parameters
- ✓ input string x of arbitrary length ⇒ output h(x) of fixed length n (bits)
- ✓ computation "easy"
- ✓ One-way functions



CRYPTOGRAPHIC PROPERTIES



PREIMAGE



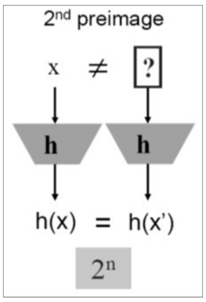
A password file must not store the passwords!
i.e. (username, password) pairs

Instead it must store:
(username,hash(password))

✓this is sufficient to verify a password
✓an attacker with access to the password file has to find a preimage

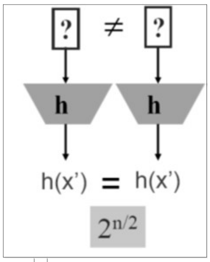
✓Still, do not use it!!!

SECOND PREIMAGE



- Can be used to protect the integrity of data x
- A secure channel is needed to send h(x) to the verifier.
- The attacker wants to modify x and remain undetected
- The attack is successful if the attacker can find a second preimage of x

COLLISION



The hacker prepares two versions of a software Let
1. x be the correct code
2. x' contain a backdoor that gives hacker access to a machine

The hacker submits x for inspection to Bob

If Bob is satisfied, he digitally signs h(x) with his private key
The hacker distributes x';
The users verify the signature with Bob's public key
This signature works for x and for x',
since h(x) = h(x')!

Birthday paradox

- the birthday problem or birthday paradox concerns the probability that, in a set of *n* randomly chosen people, some pair of them will have the same birthday.

Example: lets assume that we have a group of 23 people.

$$\binom{23}{2} = \frac{23!}{21!2!} = 253\ pairs$$

We can show that the birthday paradox is larger than 50%!

Birthday paradox

$$p(n) = 1 - \frac{n!\binom{365}{n}}{365^n}$$

n	*p*(*n*)
10	11.7%
20	41.1%
23	50.7%
30	70.6%
50	97.0%
57	99.0%
100	99.99997%
200	99.9999999999999999999999999998%
300	$(100 - (6\times10^{-80}))\%$
350	$(100 - (3\times10^{-129}))\%$
365	$(100 - (1.45\times10^{-155}))\%$
367	100%

Birthday Attack

- A birthday attack is a name used to refer to a class of brute-force attacks. More precisely,

"If some function, when supplied with a random input, returns one of |k| equally-likely values, then by repeatedly evaluating the function for different inputs, we expect to obtain the same output after about $1.2|k|^{1/2}$."

- Example: for the birthday paradox, we have |k|=365.

Brute force

- 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]
- answer: randomize hash function with a parameter S

(salt, key, spice,…)

Brute force attacks in practice

- (2nd) preimage search
  - n = 128: 23 B$ for 1 year if one can attack **240 targets in** parallel
- parallel collision search: small memory using cycle finding algorithms (distinguished points)

  – 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 era

- 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 $2^{n/2}$)

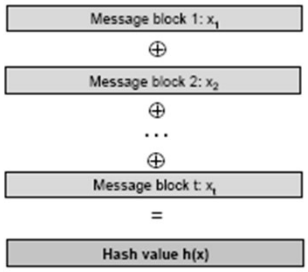
Properties in practice

- collision resistance is not always necessary
- other properties are needed:
  - PRF: pseudo-randomness if keyed (with secret key)
  - PRO: pseudo-random oracle property (formalization of security properties when there is no key)
  - near-collision resistance
  - partial preimage resistance (most of input known)
  - multiplication freeness
- how to formalize these requirements and the relation between them?

Basic Constructions

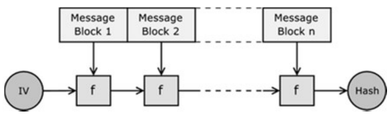
A simple approach

Divide the message into t blocks x_i of n bits each



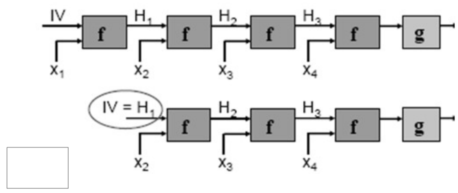
MERKLE–DAMGÅRD CONSTRUCTION

- f is a compression function
- How to choose the function
- - ad hoc
- - based on a block cipher



Iterated structure -attack

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



Merkle-Damgard strengthening

Algorithm MD-strengthening

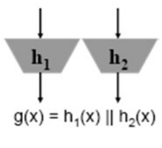
Before hashing a message $x = x_1 x_2 \dots x_t$ (where x_i is a block of bitlength r appropriate for the relevant compression function) of bitlength b, append a final length-block, x_{t+1}, containing the (say) right-justified binary representation of b. (This presumes $b < 2^r$.)

Security relation between f and h

- 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 $\overset{?}{\Leftarrow}$ h is ideally 2nd preimage resistant [Lai-Massey'92]
- property preservation has been a heavily studied topic since 2005

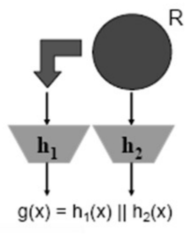
How (NOT) to strengthen a hash function?[Joux'04]

- 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…. for iterated hash functions only the strongest function matters



Multi-collisions [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: $n1 . 2^{n2/2} + 2^{n1/2} << 2^{(n1+n2)/2}$



Multi-collisions [Joux '04]

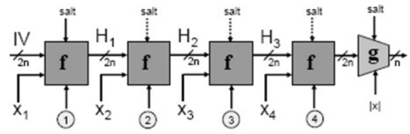
consider h_1 (n1-bit result) and h_2 (n2-bit result), with n1 ≥ n2.

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

Improving MD iteration

salt + output transformation + counter + wide pipe



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

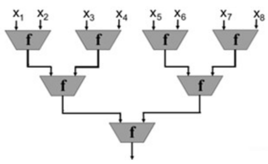
- long message 2nd preimage: preclude fix points

– counter f → fi [Biham-Dunkelman'07]

- multi-collisions, herding: avoid breakdown at $2^{n/2}$ with larger internal memory: known as wide pipe

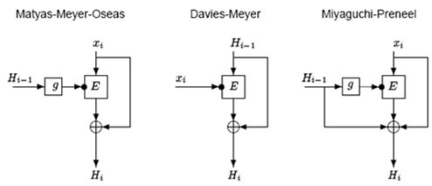
– e.g., extended MD4, RIPEMD, [Lucks'05]

Tree structure: parallelism



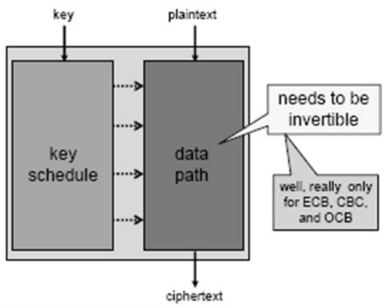
Compression Functions

Block cipher based

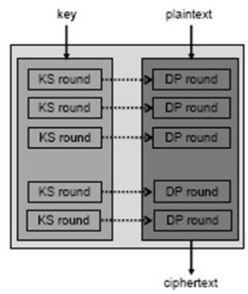


Motivation for use of a larger permutation

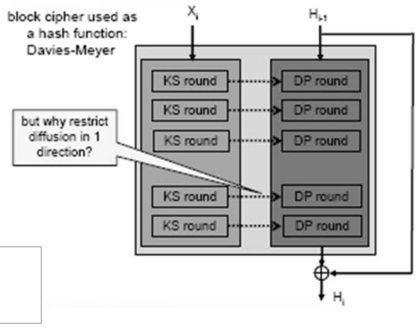
block cipher internals



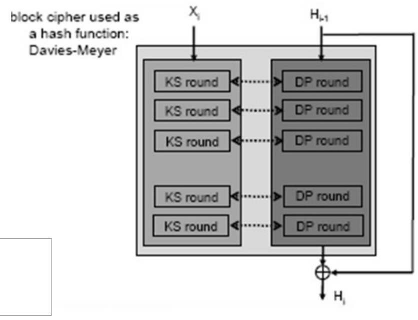
Motivation for use of a larger permutation



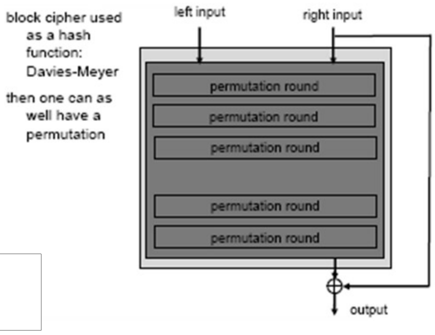
Motivation for use of a larger permutation



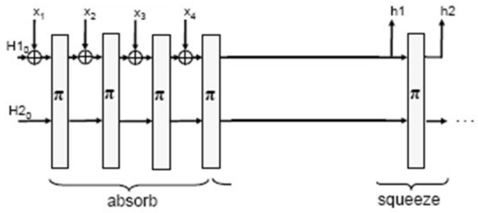
Motivation for use of a larger permutation



Motivation for use of a larger permutation



Motivation for use of a larger permutation

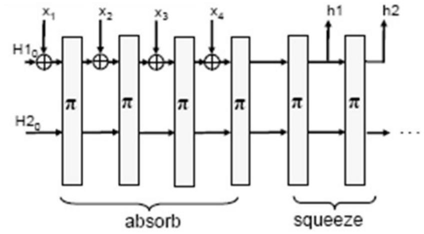


example: Keccak (no buffer)

generalization called Parazoa

JH, Cubehash, Fuge, Grindahl, Hamsi, Luffa

Motivation for use of a larger permutation



If H1 has r bits (rate) and H2 has c bits (capacity) and the permutation π is "ideal", then a sponge function has security $O(2^c)$ against (2nd) preimage attacks and $O(2^{c/2})$ against collision attacks

Iteration modes and compression functions

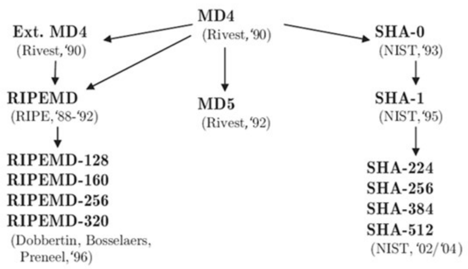
- security of simple modes well understood
  - powerful tools available
- analysis of slightly more complex schemes very difficult
  - which properties are meaningful?
  - which properties are preserved?
  - MD versus sponge is still open debate

Assistant Professor, Harokopio University of Athens, Greece

Constructions

Assistant Professor, Harokopio University of Athens, Greece

MD4 family



Assistant Professor, Harokopio University of Athens, Greece

TIMELINE

1990: MD4 by Ron Rivest

1991: MD5 by Ron Rivest (RFC 1321, 1992)

1992: RIPEMD by H. Dobbertin, A. Bosselaers and B. Preneel

1993: SHA-0 by U.S. Government (FIPS PUB 180)

1995: SHA-1 by U.S. Government (FIPS PUB 180-1)

2000: Whirlpool by V. Rijmen and P. Barreto

2001: SHA-2 by U.S. Government (FIPS PUB 180-2)

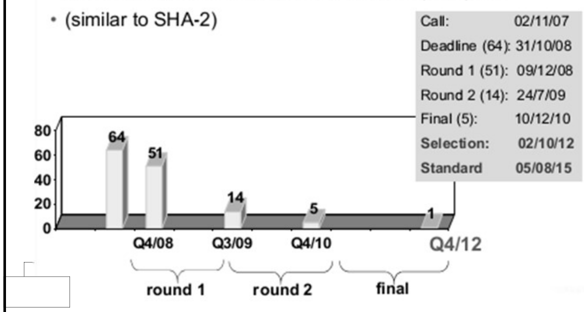
2005: First attacks against SHA-1

2015: SHA-3 by the Keccak team (FIPS 202)

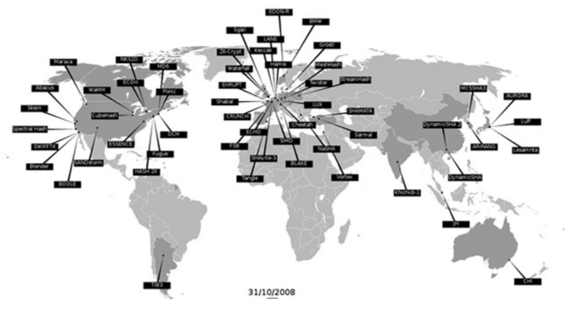
2017: February 2017, CWI Amsterdam and Google announced they had performed a collision attack against SHA-1

SHA-3 COMPETITION

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



THE CANDIDATES



From B. Preneel slides
Slides credits: Christophe De Cannière

ROUND-2 CANDIDATES

SHA-3 FINALISTS

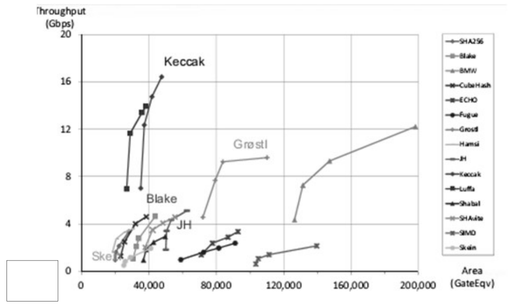
- ✓ BLAKE (Aumasson et al.)
- ✓ Grøstl (Knudsen et al.)
- ✓ JH (Hongjun Wu)
- ✓ Keccak (Keccak team, Daemen et al.)
- ✓ Skein (Schneier et al.)

Geography: 3 from Europe, 1 from Asia, 1 from America

Team members also AES finalist: 3

Hardware: post-place & route results ASIC 130nm

- [Guo-Huang-Nazhandali-Schaumont'10]



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%)
- 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})

SHA3 WINNER: KECCAK

- ✓ Not an MD construction
- ✓ Based on a new design: sponge

- ✓ Design team: Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche
- ✓ FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and
- Extendable-Output Functions
- ✓ **https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf**

STATE OF THE ART

Primitive	Output Length	Classification Legacy	Classification Future
SHA-2	256, 384, 512	✓	✓
SHA3	256,384,512	✓	✓
Whirlpool	512	✓	✓
SHA3	224	✓	✗
SHA-2	224	✓	✗
RIPEMD-160	160	✓	✗
SHA-1	160	X	✗
MD-5	128	✗	✗
RIPEMD-128	128	✗	✗

OTHER HASH FUNCTIONS

- **BLAKE2**
  - Since 2012
  - high efficiency that it offers on modern CPUs
- **Whirlpool**
  - Since 2000
  - designed by Vincent Rijmen and Paulo S. L. M. Barreto
  - 512 bits