Introduction to the Design and Cryptanalysis of Cryptographic Hash Functions

Bart Preneel
KU Leuven - COSIC
firstname.lastname@esat.kuleuven.be
Sardinia, October 2015

Hash functions

- X.509 Annex D
- 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). 1A3FD4128A198FB3CA345932

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

Agenda

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

Security requirements (n-bit result)

preimage 2nd preimage collision
h(x) = h(x') h(x) = h(x')
2^n 2^n/2

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

h(x) = h(x')
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

h(x) = h(x')
2^n

2^n/2

• 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

h(x) = h(x')
2^n

2^n/2
Second preimage resistance

2nd preimage

\[ x \neq ? \]

Channel 1: high capacity and insecure

\[ \overline{h(x)} \]

\[ h(x) = h(x') \]

\[ 2^n \]

\[ \bullet \text{an attacker can modify } x \text{ but not } h(x) \]

\[ \bullet \text{he can only fool the recipient if he finds a second preimage of } x \]

Collision resistance

\[ x \neq x' \]

Channel 2: low capacity but secure

\[ \overline{h(x)} \]

\[ h(x) = h(x') \]

\[ 2^{n/2} \]

\[ \bullet \text{hacker Alice prepares two versions of a software driver for the O/S company Bob} \]

\[ \bullet \text{x is correct code} \]

\[ \bullet \text{x' contains a backdoor that gives Alice access to the machine} \]

\[ \bullet \text{Alice submits } x \text{ for inspection to Bob} \]

\[ \bullet \text{if Bob is satisfied, he digitally signs } h(x) \text{ with his private key} \]

\[ \bullet \text{Alice now distributes } x' \text{ to users of the O/S; these users verify the signature with Bob's public key} \]

\[ \bullet \text{this signature works for } x \text{ and for } x', \text{ since } h(x) = h(x') \]

Pseudo-random function

computationally indistinguishable from a random function

\[ \text{Adv}_{prf} = \Pr [ K, x, K': A(K,x,K',\cdot) \Rightarrow 1] - \Pr [ f \overset{\text{Rand}}{\leftarrow} (m,n): A(f,\cdot) \Rightarrow 1] \]

\[ \text{RAND}(m,n): \text{set of all functions from } m\text{-bit to } n\text{-bit strings} \]

\[ K \]

\[ h \]

\[ f \]

\[ ? \text{ or } ? \]

\[ D \]

This concept makes only sense for a function with a secret key

Indifferentiability from a random oracle or PRO property [Maurer+04]

variant of indistinguishability appropriate when distinguisher has access to inner component (e.g. building block of a hash function)

\[ \exists \text{Simulator } S, \forall \text{distinguisher } D, \text{Adv}^{PRO}(H,S) \text{ is small} \]

\[ H \]

(hash function)

\[ F I L \]

RO

\[ V I L \]

RO

\[ S \]

\[ ? \text{ or } ? \]

\[ [\text{Ristenpart-Shacham-Shrimpton'11}] \]

\[ [\text{Demay-Gaz-Hirt-Maurer'13}] \]

Brute force (2nd) preimage

\[ \bullet \text{multiple target second preimage (1 out of many):} \]

\[ \bullet \text{if one can attack } 2^n \text{ simultaneous targets, the effort to find a single preimage is } 2^n \]

\[ \bullet \text{multiple target second preimage (many out of many):} \]

\[ \text{time-memory trade-off with } \Theta(2^n) \text{ precomputation and} \]

\[ \text{storage } \Theta(2^{3n/3}) \text{ time per } (2^n) \text{ preimage}; \Theta(2^{2n/3}) \]

[Hellman'80]

\[ \bullet \text{answer: randomize hash function with a parameter } S \]

(salt, key, spice,..)

Brute force attacks in practice

\[ \langle 2^{nd} \rangle \text{ preimage search} \]

\[ n = 128: \text{14 B$ for 1 year if one can attack } 2^{40} \text{ targets in parallel} \]

\[ \text{parallel collision search: small memory using} \]

\[ \text{cycle finding algorithms (distinguished points) } \]

\[ n = 128: \text{1 M$ for 5 hours (or 1 year on 60K PCs)} \]

\[ n = 160: \text{56 M$ for 1 year} \]

\[ \text{need 256-bit result for long term security (30 years or more).} \]
Quantum computers

- In principle exponential parallelism
- Inverting a one-way function: $2^n$ reduced to $2^{n/2}$ [Grover'96]
- Collision search: 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 $2^{3n}$ computation and hardware (standard cost of $2^{2n}$)

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?

Iteration
(mode of compression function)

How not to construct a hash function

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

$\text{Message block 1: } x_1 \oplus x_2 \oplus \ldots \oplus x_t = \text{Hash value } h(x)$

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…

Security relation between $f$ and $h$

- Iterating $f$ can degrade its security
  - Trivial example: $2^{nd}$ preimage

$\text{IV} = H_1 \quad H_1 \quad H_2 \quad H_3 \quad H_4 \quad g$

- Efficient and elegant… but…

- 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…

- Iterating $f$ can degrade its security
  - Trivial example: $2^{nd}$ preimage

$\text{IV} = H_1 \quad H_1 \quad H_2 \quad H_3 \quad H_4 \quad g$
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 ⇒ h is ideally 2nd preimage resistant [Lai-Massey’92]
- many other results

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

Attacks on MD-type iterations

- long message 2nd preimage attack
  [Dean-Felten-Hu’99], [Kelsey-Schneier’05]
  - Sec security degrades linearly with number 2t of message blocks hashed: 2t+1 + t 22t+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 2n
  - on-line 2n + precomputation 2 2n/2 + storage 2t

How (NOT) to strengthen a hash function?

- answer: concatenation [Coppersmith’83] [Joux’04]
  - h1 (n1-bit result) and h2 (n2-bit result)
    g(x) = h1(x) || h2(x)
  - intuition: the strength of g against collision/(2nd) preimage attacks is the product of the strength of h1 and h2
    — if both are "independent"
  - but….

Multiple collisions ≠ multi-collision

Assume "ideal" hash function h with n-bit result
- Θ(2n/2) evaluations of h (or steps): 1 collision
  - h(x) = h(x')
- Θ(r, 2n/2) steps: r2 collisions
  - h(x1) = h(x1') ; h(x2) = h(x2') ; … ; h(xr) = h(xr')
- Θ(2n/2) steps: a 3-collision
  - h(x) = h(x') = h(x'')
- Θ(2n-1/2) steps: a t-fold collision (multi-collision)
  - h(x1) = h(x2) = … = h(x)
  - now h(x1||x2||x3||x4) = h(x1||x2||x3||x4) = h(x1||x2||x3||x4) = … = h(x1||x2||x3||x4) a 16-fold collision (time: 4 collisions)
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 $R = 2^{n/2}$-fold multi-collision for $h_2$
  - In $R$: search by brute force for $h_1$

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

$$g(x) = h_1(x) \parallel h_2(x)$$

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

Consider $h_1$ (n1-bit result) and $h_2$ (n2-bit result), with $n1 \geq n2$.
Concatenation of 2 iterated hash functions $g(x) = h_1(x) \parallel h_2(x)$ 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

Improving MD iteration

- Salt + output transformation + counter + wide pipe

- Security reductions well understood
  - Many more results on property preservation
  - Impact of theory limited

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 $f \rightarrow f$ [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

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

Permutation ($\pi$) based: sponge

If result has $n$ bits, $H1$ has $r$ bits (rate), $H2$ has $c$ bits (capacity) and the permutation $\pi$ is “ideal”

- Collisions: min ($2^{r/2}$, $2^n$)
- 2^{n}th preimage: min ($2^{-r}$, $2^n$)
- Preimage: min ($2^{-r}$, $2^n$)
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

Compression functions

Single block length

[Image: 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

\[ E \]

\[ x_1 \]

\[ H_1 \]

\[ E \]

\[ x_2 \]

\[ H_2 \]

\[ E \]

\[ x_3 \]

\[ H_3 \]

\[ E \]

\[ x_4 \]

\[ H_4 \]

\[ H'_2 \]

\[ H''_2 \]

Miyaguchi-Preneel

\[ E \]

\[ x_1 \]

\[ H_1 \]

\[ E \]

\[ x_2 \]

\[ H_2 \]

\[ E \]

\[ x_3 \]

\[ H_3 \]

\[ E \]

\[ x_4 \]

\[ H_4 \]

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-Shrimpton02], [Duo-Li06], [Stam09],...

Permutation (\( \pi \)) based

parazoa

\[ \pi \]

\[ x_1 \]

\[ H_{1,1} \]

\[ H_{1,2} \]

\[ H_{1,i} \]

JH

\[ \pi \]

\[ x_2 \]

\[ H_{2,1} \]

\[ H_{2,2} \]

\[ H_{2,i} \]

small permutation

Grøstl

\[ \pi_1 \]

\[ x_3 \]

\[ H_{3,1} \]

\[ H_{3,2} \]

\[ H_{3,i} \]

\[ \pi_2 \]

\[ x_4 \]

\[ H_{4,1} \]

\[ H_{4,2} \]

\[ H_{4,i} \]

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 history 101

MDx-type hash function history

MD5: 4 rounds of 16 steps [Rivest'91]

State updates in the MD4 family

Design principles copied in MD5, RIPEMD, HAVAL, SHA, SHA-1, SHA-256, ...
- All hash functions in use today
The complexity of collision attacks

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

Rogue CA attack

[Sitriov-Stevens-Appelbaum-Lenstra-Melian-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

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)

SHA-2: FIPS 180

[Stevens-Appelbaum-Lenstra-Melian-Osvik-de Weger '08]

- 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 ($2^{117.4}$) [Li+12]
  - non-randomness 47/64 steps (practical) [Biryukov+11][Mendel+11]
- SHA-256 preimages: 45/64 steps ($2^{256}$) [Khovratovich+12]
- implementations today faster than anticipated
- adoption accelerated by other attacks on TLS
  - since 2013 deployment in TLS 1.2
SHA-3 (bits and bytes)

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

The candidates

Preliminary cryptanalysis

End of Round 1 candidates

Round 2 candidates
Hash Functions - Bart Preneel

Properties: bits and bytes
[Watanabe'10]

Software performance
eBash [Bernstein-Lange]
logarithmic scale
slower

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

Keccak
permulation: 25, 50, 100, 200, 400, 800, 1600
nominal version:
• 5x5 array of 64 bits
• 24 rounds of 5 steps

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 = 1088 (68%)
  - SHA3-512: n=512; c = 1024; r = 1088 (68%)
  - SHAKE128: n=x; c = 256; r = 1344 (84%)
  - SHAKE256: n=x; c = 512; r = 1088 (68%)

Performance of hash functions - Bernstein-Lange
(cycles/byte) Intel Core 2 Quad Q9550; 4 x 2833MHz (2008)

pad 01
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
2^n preimage
min (2^c, 2^d)

min (2^n, 2^r)
min (2^n, 2^c)
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