AK2: Cryptanalysis of iterated hash functions[2]

Christian Rechberger

May 14, 2020

[2]Thanks to Gregor Leander for an earlier version of the slides

Outline

Introduction

Design principles
SHA-1

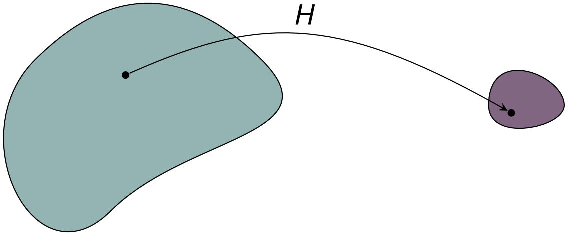
Length extension

Multicollisions

Second preimages

Introduction

- A hash function H maps strings of arbitrary length to short fixed-length bit strings (e.g., 256 bits)
- Provides a “fingerprint” of a message.



Some applications

- Digital signatures
- Password protection
- Message authentication (e.g., HMAC)
- Pseudo-randomness
- Key derivation
- ...

Password protection

User id	H(password)
...	...
La, Shangri	09283409283977
Lan, Magel	01265743912917
Lang, Serge	02973477712981
Lange, Tanja	92837540921835
Langer, Bernhard	98240254444422
...	...

Possible Attack

- Make a list of most likely passwords (only once!)
- Compute the hash values.
- Compare with the list.

Password protection, cont.

Improvement to avoid parallel attack:

User id	Salt	H(password, salt)
...	...	...
La, Shangri	68678927431	09283409283977
Lan, Magel	00000000001	01265743912917
Lang, Serge	23092839482	02973477712981
Lange, Tanja	30092341218	92837540921835
Langer, Bernhard	86769872349	98240254444422
...	...	...

Important Property of *H*

It must be hard to “invert” H.

Digital signatures

Digital Signatures

An algorithm to sign digital messages.

- Based on public key.
- Only one person can sign.
- Everybody can verify.

$$\text{sig} : \{0,1\}^n \rightarrow \{0,1\}^n$$

- The signature for n bits is n bits.
- A signed message is twice as big.
- Signing long messages is slow.

Digital signatures with hashing

Idea

Use a hashfunction!

Sign $H(m)$ instead of m

Pros:

- The signature for $*$ bits is n bits.
- A signed message is only slightly larger.
- More efficient.

Cons:

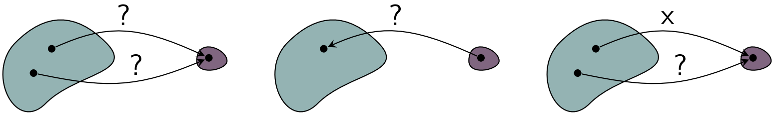
- Signature for m is signature for m' if $H(m) = H(m')$.

Important Property of H

Given m it must be hard to find m' such that $H(m) = H(m')$.

Security properties

- Collision: distinct x and x' with $H(x) = H(x')$
- Preimage: Given $H(x)$, find x' such that $H(x') = H(x)$
- Second preimage: Given x, find $x' \neq x$ such that $H(x) = H(x')$.



Why these security properties?

- Example: **digital signatures**
- Hash message, then sign it
- Collision means signature is valid for two messages (Bob can cheat)
- Second preimage means Eve can cheat
- In practice, a preimage usually also means Eve can cheat

Birthday attack

- Collision attack (n-bit hash function)
- After q queries we have $\binom{q}{2} = q(q-1)/2$ hash pairs (h, h')
- Probability that $h = h'$ is about 2^{-n}
- $\lim_{x \to \infty} (1 - 1/x)^x = 1/e$
- Hence, with 2^n pairs, we probably have a collision
- With $q = 2^{n/2}$, we have about 2^n pairs.

Brute force preimage and second preimage attacks

- Ideal n-bit hash function
- Best attack: try random messages
- Probability for each is 2^{-n}
- I.e., try about 2^n messages.

Meaningful messages?

- Random messages are not meaningful
- Choose a meaningful message
- Identify k character positions that may have either of two values
- Construct 2^k variations of message.

We hold these truths to be self evident; that all men are created equal
We hold these truths to be self evident, that all men are created equal
We hold these truths to be self-evident; that all men are created equal
We hold these truths to be self-evident, that all men are created equal

Digital signatures with hashing

Idea

Use a hashfunction!

Sign $H(m)$ instead of m

Pros:

- The signature for $*$ bits is n bits.
- A signed message is only slightly larger.
- More efficient.

Cons:

- Signature for m is signature for m' if $H(m) = H(m')$.

Question

Do random collisions matter in practice?

More than you might think!

Collision in Postscript (Daum-Lucks 2005)

Postscript

- Postscript is a kind of programming language
- (S1)(S2)eqT1T2ifelse
- Meaning: If $S1 = S2$ then display $T1$ else display $T2$

Random Collisions are important!

- Find random messages S1 and S2 which collide under hash function
- Construct PS1 and PS2 for arbitrary T1 and T2
- PS1:...(S1)(S2)eqT1T2ifelse...
- PS2: ...(S2)(S2)eqT1T2ifelse...

Collision in Postscript (Daum-Lucks 2005)

Random Collisions are important!

- Find random messages S1 and S2 which collide under hash function
- Construct PS1 and PS2 for arbitrary T1 and T2
- PS1:...(S1)(S2)eqT1T2ifelse...
- PS2: ...(S2)(S2)eqT1T2ifelse...
- PS1 and PS2 have the same hash value.
- PS1 displays T2.
- PS2 displays T1.

Consequences of the Attacks

SHA-3 initiative

- Researchers were evaluating alternative hash functions in the SHA-3 initiative organized by NIST
- NIST selected Keccak as SHA-3

Transition from SHA-1 to SHA-2

- NIST proposed the transition from SHA-1 to the SHA-2 family
- Companies and organization are expected to migrate to SHA-2

Outline

Introduction

Design principles
- SHA-1

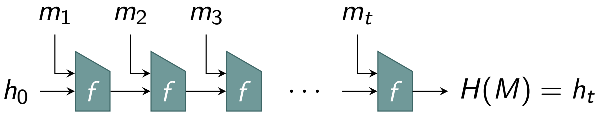
Length extension

Multicollisions

Second preimages

Typical design method

- Design a *compression function* $f : \{0,1\}^b \rightarrow \{0,1\}^n$
- Initial n-bit state value h_0
- Message $M = m_1 \| m_2 \| m_3 \| \cdots \| m_t$, $|m_i| = b - n$
- $h_1 = f(h_0 \| m_1)$, $h_2 = f(h_1 \| m_2)$, ...
- Final value h_t is the hash
- (Padding may be necessary).



Advantages of iterating

- Message may come in small packets, hashing can start before all is received
- Limited amount of memory needed
- Once a message block is hashed, it can be forgotten.

Merkle-Damgård

- Take a collision resistant *compression function f*
- Pad message M by appending '0' bits *and* the length $|M|$ (MD-strengthening)
- Iterate as described
- Now you have a collision resistant *hash function*.

The challenges of hash function design

- It is “easy” to design a secure hash function. But is it fast?
- It is “easy” to design a fast hash function. But is it secure?
- Remember: nothing is secret!

Hash functions in real-life

Scheme	Bits in hash code	Compression fct.	Designer	Year
MD4	128	$\{0,1\}^{512+128} \to \{0,1\}^{128}$	Rivest	1990
MD5	128	$\{0,1\}^{512+128} \to \{0,1\}^{128}$	Rivest	1991
SHA-0	160	$\{0,1\}^{512+160} \to \{0,1\}^{160}$	US Gov.	1993
SHA-1	160	$\{0,1\}^{512+160} \to \{0,1\}^{160}$	US Gov.	1995
SHA-256	256	$\{0,1\}^{512+256} \to \{0,1\}^{256}$	US Gov.	2002
SHA-512	512	$\{0,1\}^{1024+512} \to \{0,1\}^{512}$	US Gov.	2002

MD: Message Digest   SHA: Secure Hash Algorithm

SHA-1

SHA-1

- Widely used hash function.
- Iterated MD hash function
- 160 bit output.
- Based on compression function

$$h : \{0,1\}^{512} \times \{0,1\}^{160} \rightarrow \{0,1\}^{160}$$

- *h* itself is round based.
- Uses XOR, modular additions and rotations.

Hashing with SHA-1

MD for SHA-1

1. pad message, s.t. last block is 512-64 bits
2. append 64-bit block containing length of original message
3. Set $H_0 = (A, B, C, D, E)$
4. for each message block M_i of 512 bits:
   4.1 compute $H_{i+1} = h(M_{i+1}, H_i)$
5. The final value H_j is the hash value.

Compression function $h : \{0,1\}^{512} \times \{0,1\}^{160} \rightarrow \{0,1\}^{160}$

$h : \{0,1\}^{512} \times \{0,1\}^{160} \rightarrow \{0,1\}^{160}$

$h : \{0,1\}^{512} \times \{0,1\}^{160} \rightarrow \{0,1\}^{160}$

- 80 basic steps in compression function.
- Each step depends on a 32 part of the message block M_i^j.
- 80 message dependent 32 bit values needed M_i^j.
- Given by *Message Expansion*.

Message Expansion

Input: message $M_i = [M_i^0 \parallel M_i^1 \parallel \ldots \parallel M_i^{15}]$, where M_i^j are 32-bit words.

Output: $M_i^j = \mathrm{rot}_1(M_i^{j-3} \oplus M_i^{j-8} \oplus M_i^{j-14} \oplus M_i^{j-16}), \quad 16 \leq j \leq 79$

Compression function (continued)

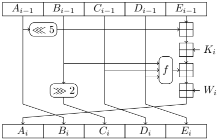


Figure: One round of SHA-1

Compression function (continued)

$h(M_{i+1}, H_i)$

Functions Used

$$
\begin{aligned}
f^i &= f_{if} &&= (X\&Y)|(\neg X\&Z), && 0 \le i \le 19 \\
f^i &= f_{xor} &&= X \oplus Y \oplus Z, && 20 \le i \le 39, 60 \le i \le 79 \\
f^i &= f_{maj} &&= (X\&Y)|(X\&Z)|(Y\&Z), && 40 \le i \le 59.
\end{aligned}
$$

Output of h

Set
$A = A + A^{80}, B = B + B^{80}, C = C + C^{80}, D = D + D^{80}, E = E + E^{80}$
Output (A, B, C, D, E).

Outline

Introduction

Design principles
SHA-1

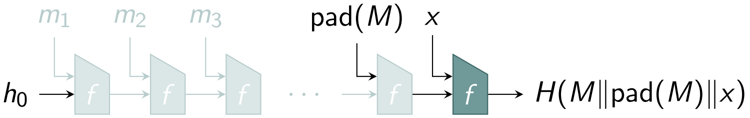
Length extension

Multicollisions

Second preimages

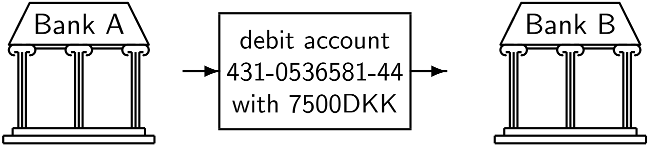
Length extension

- Assume we know $H(M)$ and $|M|$, but not M itself
- Knowing $|M|$, we can compute the padding of M (pad(M))
- Now we can compute $H(M \| \text{pad}(M) \| x)$ for any x.



This is a problem for a MAC.

A typical example: Financial transaction

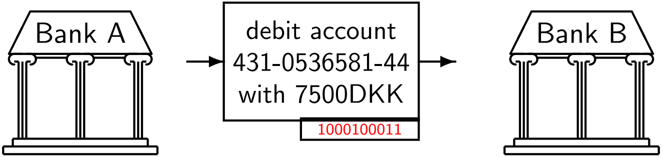


Problem

Bank B would like to check if transaction

- was not changed
- was really sent by Bank A

A typical example: Financial transaction



MAC

Compute a short bit string that

- depends on the message.
- depends on a pre-shared secret key.

Example application

- A MAC is a keyed primitive used to ensure data integrity and authenticity of a message
- Without knowing the key, it should be impossible to compute a valid message/MAC pair
- Assume $\text{MAC}_k(x) = H(k\|x)$, H an iterated hash function
- Eve sees a message/MAC pair $(M, MAC_k(M))$
- Without knowing k, she can compute $\text{MAC}_k(M^*)$ for many other messages M^*.

HMAC

Another widely used MAC:

HMAC

MAC of message x is:

$$\mathrm{MAC}_K(x) = H(K_2 \mid H(K_1 \mid x))$$

- HMAC popular with H=SHA-1 or MD5
- With H=SHA-1 or MD5, K_1 and K_2 keys of 512 bits each

HMAC

Use HMAC instead!

HMAC

MAC of message x is:

$$\mathrm{MAC}_K(x) = H(K_2 \mid H(K_1 \mid x))$$

Theorem

HMAC secure if

- *SHA-1 is collision resistant for secret initial value,* **and**
- *H is a secure MAC for one-block messages.*

Length extension 2

- Assume we have found a collision (x, x^*), with $|x| = |x^*|$
- Hence, $\text{pad}(x) = \text{pad}(x^*)$
- We can find many other collisions:
  $(x \| \text{pad}(x) \| y, \quad x^* \| \text{pad}(x^*) \| y)$ for any y.

Outline

Introduction

Design principles
SHA-1

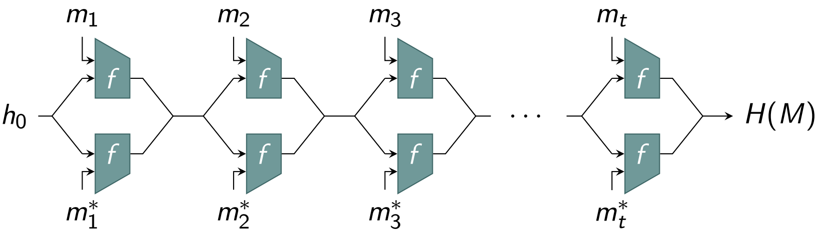
Length extension

Multicollisions

Second preimages

Multicollisions

- A set of $r \geq 2$ messages all having the same hash
- Complexity for an ideal hash function? Exercise
- Joux, 2004: 2^t-collision in time $t2^{n/2}$.



An application

Combining two Hash-Functions

Let us build a $2n$-bit hash function from two n-bit hash functions H_1 and H_2:

$$H(M) = H_1(M) \| H_2(M)$$

- What is the expected strength of that construction?
- What changes if H_1 is an iterated construction?

An application

- Let us build a $2n$-bit hash function from two n-bit hash functions H_1 and H_2:

$$H(M) = H_1(M) \| H_2(M)$$

- Assume H_1 is an iterated hash function
- Using Joux, find $2^{n/2}$-collision in H_1 (time $(n/2)2^{n/2}$)
- $2^{n/2}$ messages $\rightarrow$ collision in H_2 (with good probability)
- Total time about $n2^{n/2}$.

Outline

Introduction

Design principles

SHA-1

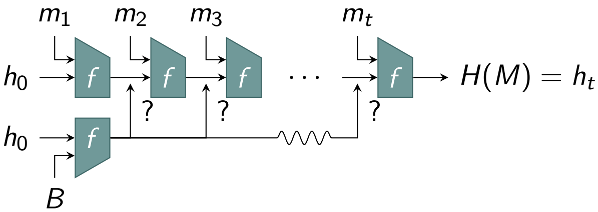
Length extension

Multicollisions

Second preimages

Second preimages for long messages

- Let M be a very long message
- A second message can map to any intermediate value to form a second preimage
- Problem: message lengths don't fit (MD-strengthening)
- Solution?



Solution 1

- Find a *fixed point* in f (Davies-Meyer)

$$f(h, m) = h$$

- Repeat m as many times as necessary to make lengths fit
- Sub-problem: $h \neq h_0$
- Sub-solution:
  - find $2^{n/2}$ fixed points, place in list L
  - find linking message block b s.t. $f(h_0, b) \in L$
  - time about $2^{n/2}$ when Davies-Meyer.

Solution 2

- Yet another application of Joux
- Find colliding messages of lengths 1 and 2 (blocks)
- Find colliding messages of lengths 1 and 3
- Find colliding messages of lengths 1 and 5
- ...
- Find colliding messages of lengths 1 and $2^k + 1$
- Combine blocks to form message of length $k + 1$, $k + 2$, ..., $2^{k+1} + k$ (2^k different message lengths).

Wrapping up

- Both solutions provide method to find multicollisions of messages of different lengths (prefix)
- From the output of the multicollision, link to an intermediate value from computation of $H(M)$
- From the match on, choose same blocks as in M
- Now choose prefix of proper length
- Time about 2^{n-t} if $|M| = 2^t$ (blocks).

Conclusions

Modern solution to avoid all these problems. Sponge-based hash functions, e.g SHA-3, that have a large internal state. Not that widely used yet.