CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

Apr 14, 2022

Lecture 6a

Hash functions

Hash functions

Collision resistance

Compression functions and the Merkle-Damgård (MD) transform

Hash functions from block ciphers: the Davies-Meyer method

Cryptanalytic attacks

Plan

New Topic: Hash functions

- MD: MD4, MD5, MD6
- SHA2: SHA1, SHA224, SHA256, SHA384, SHA512
- SHA3: SHA3-224, SHA3-256, SHA3-384, SHA3-512

Their primary purpose is collision-resistant data compression, but they have many other purposes and properties as well ... A hash function is often treated like a magic wand ...

Some uses:

- Certificates: How you know `www.snapchat.com` really is Snapchat
- Bitcoin
- Data authentication with HMAC: TLS, ...

New Topic: Hash functions

- MD: MD4, MD5, MD6
- SHA2: SHA1, SHA224, SHA256, SHA384, SHA512
- SHA3: SHA3-224, SHA3-256, SHA3-384, SHA3-512

Their primary purpose is collision-resistant data compression, but they have many other purposes and properties as well ... A hash function is often treated like a magic wand ...

Some uses:

- Certificates: How you know `www.snapchat.com` really is Snapchat
- Bitcoin
- Data authentication with HMAC: TLS, ...

SHA = "<u>S</u>ecure <u>H</u>ash <u>A</u>lgorithm"

SHA1 is dead ...

BIZ & IT —

At death's door for years, widely used SHA1 function is now dead

Algorithm underpinning Internet security falls to first-known collision attack.

DAN GOODIN - 2/23/2017, 5:01 AM

Bob Embleton

For more than six years, the SHA1 cryptographic hash function underpinning Internet security has been at death's door. Now it's officially dead, thanks to the submission of the first known instance of a fatal exploit known as a "collision."

Definition of a hash function

Definition

A hash function is just a family of functions $H : \text{Keys} \times D \to R$ of functions, meaning for each $K \in \text{Keys}$ we have a function $H_K : D \to R$ defined by $H_K(x) = H(K, x)$. The hash function may be:

- keyless if Keys only contains the empty string: $\text{Keys} = \{\varepsilon\}$.
- keyed if Keys is a non-trivial set.

The domain D is typically all arbitrary length strings, and the range is typically a set of fixed-length bit strings.

“Hash” is a common word

WARNING: a cryptographic hash function has nothing (or very little) to do with a hash table.

- a hash table does use a `Key` $\rightarrow$ `Value` function, sometimes even called a hash function. . .
- but a cryptographic hash function has to meet much stronger criteria!

A cryptographic hash function would be good (albeit slow) for a hash table, but not the converse!

Requirements for cryptographic hash functions

Let $h : D \to \{0,1\}^n$ be some hash function (keyed or not).

- Preimage: Given a random $y \in \{0,1\}^n$, can we find $x \in D$ such that $h(x) = y$?
- Second preimage: Given a random $x \in D$, can we find $x' \in D$ such that $h(x) = h(x')$?
- Collisions: Can we find two elements $x, x' \in D$ such that $h(x) = h(x')$?

- $\uparrow$These goals are increasingly harder to reach for an adversary.
- $\downarrow$ On the other hand, when we want to define security, it is a stricter requirement to ask for even collision resistance to be infeasible in practice.

Plan

Hash functions

Collision resistance

Compression functions and the Merkle-Damgård (MD) transform

Hash functions from block ciphers: the Davies-Meyer method

Cryptanalytic attacks

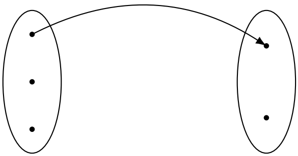
Collisions

Definition: collision

A **collision** for a function $h : D \to \{0,1\}^n$ is a pair $x_1, x_2 \in D$ of points such that

- $h(x_1) = h(x_2)$, and
- $x_1 \neq x_2$.

If $|D| > 2^n$ then the pigeonhole principle tells us that there must exist a collision for h.



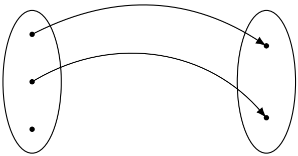
Collisions

Definition: collision

A **collision** for a function $h : D \to \{0,1\}^n$ is a pair $x_1, x_2 \in D$ of points such that

- $h(x_1) = h(x_2)$, and
- $x_1 \neq x_2$.

If $|D| > 2^n$ then the pigeonhole principle tells us that there must exist a collision for h.



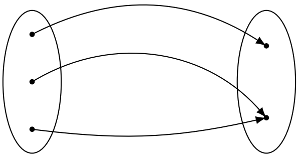
Collisions

Definition: collision

A **collision** for a function $h : D \to \{0,1\}^n$ is a pair $x_1, x_2 \in D$ of points such that

- $h(x_1) = h(x_2)$, and
- $x_1 \neq x_2$.

If $|D| > 2^n$ then the pigeonhole principle tells us that there must exist a collision for h.



We want that even though collisions exist, **they are hard to find**.

Collision-resistance of a function family

The formalism considers a family H : Keys $\times D \to R$ of functions, meaning for each $K \in$ Keys we have a function $H_K : D \to R$ defined by $H_K(x) = H(K, x)$.

Game CR_H

procedure Initialize
$K \xleftarrow{\$}$ Keys
Return K

procedure Fn(x)
Return $H_K(x)$

procedure Finalize(x_1, x_2)
If $(x_1 = x_2)$ then return false
If $(x_1 \notin D$ or $x_2 \notin D)$ then return false
Return $(H_K(x_1) = H_K(x_2))$

Let

$$\mathbf{Adv}_H^{\mathrm{cr}}(A) = \Pr\left[\mathrm{CR}_H^A \Rightarrow \mathsf{true}\right].$$

Collision-resistance

Game CR_H

procedure Initialize
$K \xleftarrow{\$} \text{Keys}$
Return K

procedure Fn(x)
Return $H_K(x)$

procedure Finalize(x_1, x_2)
If $(x_1 = x_2)$ then return false
If $(x_1 \notin D$ or $x_2 \notin D)$ then return false
Return $(H_K(x_1) = H_K(x_2))$

Here, the adversary can see inside the box!

- The Return statement in **Initialize** means that the adversary A gets K as input. The key K here is not secret!
- The **Fn** oracle uses nothing that the adversary doesn't know, so it's public.

Adversary A takes K and tries to output a collision x_1, x_2 for H_K.
A's output is the input to **Finalize**, and the game returns true if the collision is valid.

Example

Let $N = 2^{256}$ and define

$$H\colon\ \underbrace{\{1,\ldots,N\}}_{\text{Keys}} \times \underbrace{\{0,1,2,\ldots\}}_{D} \to \underbrace{\{0,1,\ldots,N-1\}}_{R}$$

by

$$H(K,x) = (x \bmod K)\ .$$

Q: Is H collision resistant?

Example

Let $N = 2^{256}$ and define

$$H\colon \underbrace{\{1,\ldots,N\}}_{\text{Keys}} \times \underbrace{\{0,1,2,\ldots\}}_{D} \to \underbrace{\{0,1,\ldots,N-1\}}_{R}$$

by

$$H(K,x) = (x \bmod K)\ .$$

Q: Is H collision resistant?

A: NO!

Why? $(x+K) \bmod K = x \bmod K$

adversary $A(K)$
$x_1 \leftarrow 0$; $x_2 \leftarrow K$; Return x_1, x_2

$\mathbf{Adv}^{\mathrm{cr}}_{H}(A) = 1$

Example

Let E: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher.
Let H: $\{0,1\}^k \times \{0,1\}^{2n} \to \{0,1\}^n$ be defined by

$\underline{\textbf{Alg } H(K, x[1]x[2])}$
$y \leftarrow E_K(E_K(x[1]) \oplus x[2])$; Return y

Let's show that H is not collision-resistant by giving an efficient adversary A such that $\mathbf{Adv}_H^{\mathrm{cr}}(A) = 1$.

Example

Let E: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher.
Let H: $\{0,1\}^k \times \{0,1\}^{2n} \to \{0,1\}^n$ be defined by

Alg $H(K, x[1]x[2])$

$y \leftarrow E_K(E_K(x[1]) \oplus x[2])$; Return y

Let's show that H is not collision-resistant by giving an efficient adversary A such that $\mathbf{Adv}_H^{\mathrm{cr}}(A) = 1$.

Idea: Pick $x_1 = x_1[1]x_1[2]$ and $x_2 = x_2[1]x_2[2]$ so that

$$E_K(x_1[1]) \oplus x_1[2] = E_K(x_2[1]) \oplus x_2[2]$$

Example

Alg $H(K, x[1]x[2])$

$y \leftarrow E_K(E_K(x[1]) \oplus x[2])$; Return y

Idea: Pick $x_1 = x_1[1]x_1[2]$ and $x_2 = x_2[1]x_2[2]$ so that

$$E_K(x_1[1]) \oplus x_1[2] = E_K(x_2[1]) \oplus x_2[2]$$

Many possible answers:

Example

$\underline{\textbf{Alg } H(K, x[1]x[2])}$

$y \leftarrow E_K(E_K(x[1]) \oplus x[2])$; Return y

Idea: Pick $x_1 = x_1[1]x_1[2]$ and $x_2 = x_2[1]x_2[2]$ so that

$$E_K(x_1[1]) \oplus x_1[2] = E_K(x_2[1]) \oplus x_2[2]$$

Many possible answers:

adversary $A_1(K)$

$x_1[1] \leftarrow 0^n$; $x_2[1] \leftarrow 1^n$; $x_1[2] \leftarrow E_K(x_1[1])$; $x_2[2] \leftarrow E_K(x_2[1])$

return x_1, x_2

Then $\mathbf{Adv}^{\mathrm{cr}}_H(A_1) = 1$ and A_1 is efficient, so H is not CR.

Example

<u>**Alg** $H(K, x[1]x[2])$</u>

$y \leftarrow E_K(E_K(x[1]) \oplus x[2])$; Return y

Idea: Pick $x_1 = x_1[1]x_1[2]$ and $x_2 = x_2[1]x_2[2]$ so that

$$E_K(x_1[1]) \oplus x_1[2] = E_K(x_2[1]) \oplus x_2[2]$$

Many possible answers:

adversary $A_2(K)$
$x_1 \leftarrow 0^n1^n$; $x_2[2] \leftarrow 0^n$; $x_2[1] \leftarrow E_K^{-1}(E_K(x_1[1]) \oplus x_1[2] \oplus x_2[2])$
$\mathrm{return}\ x_1, x_2$

Then $\mathbf{Adv}_H^{\mathrm{cr}}(A_2) = 1$ and A_2 is efficient, so H is not CR.

Note how we used the fact that A_2 knows K and the fact that E is a block cipher!

Exercise

Let E: $\{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$ be a block cipher. Let D be the set of all strings whose length is a positive multiple of ℓ.

Define the hash function H: $\{0,1\}^k \times D \to \{0,1\}^\ell$ as follows:

Alg $H(K, M)$

$M[1]M[2]\ldots M[n] \leftarrow M$

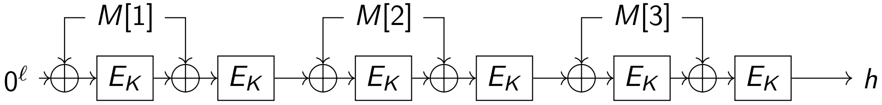
$C[0] \leftarrow 0^\ell$

For $i = 1, \ldots, n$ do

  $B[i] \leftarrow E(K, C[i-1] \oplus M[i])$;

  $C[i] \leftarrow E(K, B[i] \oplus M[i])$

Return $C[n]$



Show that H is not CR by giving an efficient adversary A such that $\mathbf{Adv}_H^{\mathrm{cr}}(A) = 1$.

Keyless hash functions

We say that H: Keys $\times D \rightarrow R$ is keyless if Keys $= \{\varepsilon\}$ consists of just one key, the empty string.

In this case we write $H(x)$ in place of $H(\varepsilon, x)$ or $H_\varepsilon(x)$.

Practical hash functions like the MD, SHA2 and SHA3 series are keyless.

Plan

Collision resistance

- An example: SHA256
- Applications of hash functions
- Inherent limitations: birthday attacks

SHA256

The hash function SHA256: $\{0,1\}^{<2^{64}} \to \{0,1\}^{256}$ is **keyless**, with

- Inputs being strings X of any length strictly less than 2^{64}
- Outputs always having length 256.

<u>**Alg** SHA256(X)</u> // $|X| < 2^{64}$

$M \leftarrow \text{shapad}(X)$ // $|M| \bmod 512 = 0$

$M^{(1)}M^{(2)}\cdots M^{(n)} \leftarrow M$ // Break M into 512 bit blocks

$H_0^{(0)} \leftarrow \texttt{6a09e667}$; $H_1^{(0)} \leftarrow \texttt{bb67ae85}$; $\cdots$; $H_7^{(0)} \leftarrow \texttt{5be0cd19}$

$H^{(0)} \leftarrow H_0^{(0)}H_1^{(0)}\cdots H_7^{(0)}$ // $|H_i^{(0)}| = 32$, $|H^{(0)}| = 256$

For $i = 1, \ldots, n$ do $H^{(i)} \leftarrow \text{sha256}(M^{(i)} \| H^{(i-1)})$

Return $H^{(n)}$

sha256: $\{0,1\}^{512+256} \to \{0,1\}^{256}$ is the **compression function**.

Padding, and initialization vector $H^{(0)}$

Alg shapad(X) // $|X| < 2^{64}$

$d \leftarrow (447 - |X|) \bmod 512$ // Chosen to make $|M|$ a multiple of 512

Let ℓ be the 64-bit binary representation of $|X|$

$M \leftarrow X \,\|\, 1 \,\|\, 0^d \,\|\, \ell$ // $|M|$ is a multiple of 512

return M

The 32-bit word $H_j^{(0)}$ was obtained by taking the first 32 bits of the fractional part of the square root of the j-th prime number ($0 \leq j \leq 7$).

Padding, and initialization vector $H^{(0)}$

Alg shapad(X) // $|X| < 2^{64}$
$d \leftarrow (447 - |X|) \bmod 512$ // Chosen to make $|M|$ a multiple of 512
Let ℓ be the 64-bit binary representation of $|X|$
$M \leftarrow X \,\|\, 1 \,\|\, 0^d \,\|\, \ell$ // $|M|$ is a multiple of 512
return M

The 32-bit word $H_j^{(0)}$ was obtained by taking the first 32 bits of the fractional part of the square root of the j-th prime number ($0 \leq j \leq 7$).

Question: Why square roots as opposed to simply picking the words at random and embedding them in the code?

Speculation: Perhaps to prevent suspicion of subversion (planting of a backdoor)?

Compression function sha256

Compression function sha256: $\{0,1\}^{512+256} \to \{0,1\}^{256}$ takes a $512 + 256 = 768$ bit input and returns a 256-bit output.

<u>**Alg** sha256($x \parallel v$)</u> // $|x|$=512, $|v|$=256

$w \leftarrow \mathsf{E}^{\mathsf{sha256}}(x, v)$

$w_0 \cdots w_7 \leftarrow w$ // Break w into 32-bit words

$v_0 \ldots v_7 \leftarrow v$ // Break v into 32-bit words

For $j = 0, \ldots, 7$ do $h_j \leftarrow w_j + v_j$

$h \leftarrow h_0 \ldots h_7$ // $|h| = 256$

Return h

Here and on next slide, "$+$" denotes addition modulo 2^{32}.

$\mathsf{E}^{\mathsf{sha256}}$: $\{0,1\}^{512} \times \{0,1\}^{256} \to \{0,1\}^{256}$ is a **block cipher** with 512-bit keys and 256-bit blocks.

Block cipher $\mathsf{E}^{\mathsf{sha256}}$

Alg $\mathsf{E}^{\mathsf{sha256}}(x, v)$ // x is a 512-bit key, v is a 256-bit input
$x_0 \cdots x_{15} \leftarrow x$ // Break x into 32-bit words
For $t = 0, \ldots, 15$ do $W_t \leftarrow x_t$
For $t = 16, \ldots, 63$ do $W_t \leftarrow \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$
$v_0 \cdots v_7 \leftarrow v$ // Break v into 32-bit words
For $j = 0, \ldots, 7$ do $S_j \leftarrow v_j$ // Initialize 256-bit state S
Fot $t = 0, \ldots, 63$ do // 64 rounds
    $T_1 \leftarrow S_7 + \gamma_1(S_4) + Ch(S_4, S_5, S_6) + C_t + W_t$
    $T_2 \leftarrow \gamma_0(S_0) + Maj(S_0, S_1, S_2)$
    $S_7 \leftarrow S_6$; $S_6 \leftarrow S_5$; $S_5 \leftarrow S_4$; $S_4 \leftarrow S_3 + T_1$
    $S_3 \leftarrow S_2$; $S_2 \leftarrow S_1$; $S_1 \leftarrow S_0$; $S_0 \leftarrow T_1 + T_2$
$S \leftarrow S_0 \cdots S_7$
Return S // 256-bit output

Internals of block cipher E^{sha256}

On the previous slide:

- $\sigma_0, \sigma_1, \gamma_0, \gamma_1, Ch, Maj$ are functions not detailed here.
- $C_1 =$ `428a2f98`, $C_2 =$ `71374491`, ..., $C_{63} =$ `c67178f2` are constants, where C_i is the first 32 bits of the fractional part of the cube root of the i-th prime.

SHA256 in action

Try it on the course web page!

- Over the years, SHA256 has become commonplace. It is (most probably) supported by your web browser, which executes the example above in Javascript.
- However, it takes time for such goodies to percolate. No SHA-3 in there yet (check the documentation for possible updates)

Plan

Usage of hash functions

Uses include hashing the data before signing in creation of certificates, data authentication with HMAC, key-derivation, Bitcoin, ...

These will have to wait, so we illustrate another use, the hashing of passwords.

Authentication via passwords

- Client A has a password PW that is also stored by server B
- A authenticates itself by sending PW to B over a secure channel (TLS)

$$A^{PW} \xrightarrow{\quad PW \quad} B^{PW}$$

Problem: The password will be found by an attacker who compromises the server.

These types of server compromises are common and often in the news: Yahoo, Equifax, ...

Hashed passwords

- Client A has a password PW and server stores $\overline{PW} = H(PW)$.
- A sends PW to B (over a secure channel) and B checks that $H(PW) = \overline{PW}$

$$A^{PW} \xrightarrow{\quad PW \quad} B^{\overline{PW}}$$

Server compromise results in attacker getting $\overline{PW}$ which should not reveal PW as long as H is one-way, which is a consequence of collision-resistance.

But we will revisit this when we consider dictionary attacks!

This is (part of) how client authentication is done on the Internet, for example login to `gmail.com`.

Plan

Collision resistance

- An example: SHA256
- Applications of hash functions
- Inherent limitations: birthday attacks

Birthday collision-finding attack

Let $H : \{0,1\}^k \times D \to \{0,1\}^n$ be a family of functions with $|D| > 2^n$. The q-trial birthday attack is the following adversary A_q for game CR_H:

adversary $A_q(K)$
for $i = 1, \ldots, q$ do $x_i \xleftarrow{\$} D$; $y_i \leftarrow H_K(x_i)$
if $\exists i, j$ ($i \neq j$ and $y_i = y_j$ and $x_i \neq x_j$) then return x_i, x_j
else return $\perp$

We can analyze this via the birthday problem, and show that

$$\mathbf{Adv}_H^{\mathrm{cr}}(A_q) \geq 0.3 \cdot \frac{q(q-1)}{2^n} .$$

So a collision can usually be found in about $q = \sqrt{2^n}$ trials.

Cost of birthday attacks

adversary $A_q(K)$

for $i = 1, \ldots, q$ do $x_i \xleftarrow{\$} D$; $y_i \leftarrow H_K(x_i)$

if $\exists i, j$ ($i \neq j$ and $y_i = y_j$ and $x_i \neq x_j$) then return x_i, x_j

else return $\perp$

If $q \approx 2^{n/2}$, this is expected to succeed.

Cost in memory: $\approx 2^{n/2}$ as well.

Cost of birthday attacks

adversary $A_q(K)$
for $i = 1, \ldots, q$ do $x_i \xleftarrow{\$} D$; $y_i \leftarrow H_K(x_i)$
if $\exists i, j$ ($i \neq j$ and $y_i = y_j$ and $x_i \neq x_j$) then return x_i, x_j
else return $\perp$

If $q \approx 2^{n/2}$, this is expected to succeed.
Cost in memory: $\approx 2^{n/2}$ as well.

BUT there are multiple ways to optimize this and obtain (almost) the same cost with memory $O(1)$. One approach is to look for cycles in the graph

$$x \to H(x) \to H(H(x)) \to \ldots$$

It takes time $O(2^{n/2})$ and memory $O(1)$ to find collisions on a hash function with n-bit output.

Birthday attack times

Function	n	T_B
MD4	128	2^{64}
MD5	128	2^{64}
SHA1	160	2^{80}
SHA256	256	2^{128}
SHA512	512	2^{256}
SHA3-256	256	2^{128}
SHA3-512	512	2^{256}

T_B is the number of trials to find collisions via a birthday attack.

Design of hash functions aims to make the birthday attack the best collision-finding attack, meaning it is desired that there be no attack succeeding in time much less than T_B.

repeat until fully understood

Collision resistance security of a hash function that outputs n bits **cannot** exceed $n/2$-bit security, because of the birthday paradox attack.

(x-bit security means: breaking purportedly takes time $\geq 2^x$)

preimage and second preimage security are a different story, but these are weaker notions.

Plan

Hash functions

Collision resistance

Compression functions and the Merkle-Damgård (MD) transform

Hash functions from block ciphers: the Davies-Meyer method

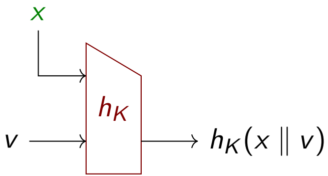
Cryptanalytic attacks

Compression functions

A compression function is a family $h : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$ of functions whose inputs are of a fixed size $b + n$, where b is called the block size.

E.g. $b = 512$ and $n = 256$, in which case

$$h : \{0,1\}^k \times \{0,1\}^{768} \to \{0,1\}^{256}$$



The MD transform

Let $h : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$ be a compression function with block length b. Let D be the set of all strings of at most $2^b - 1$ blocks.

The **MD transform** builds from h a family of functions

$$H : \{0,1\}^k \times D \to \{0,1\}^n.$$

Coming next:

- How the MD transform works
- The nice properties of MD

MD setup

Given: Compression function $h : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$.

Build: Hash function $H : \{0,1\}^k \times D \to \{0,1\}^n$.

Since $M \in D$, its length $|M|$ is a multiple of the block length b.

Definition: length in number of blocks

We let $|M|_b = |M|/b$ be the number of b-bit blocks in M.

We parse M as

$$M[1] \dots M[\ell] \leftarrow M \; .$$

Note: in PlayCrypt, this is done with `M = split(M,b_bytes)`.

Let $\langle \ell \rangle$ denote the b-bit binary representation of $\ell \in \{0, \dots, 2^b - 1\}$.

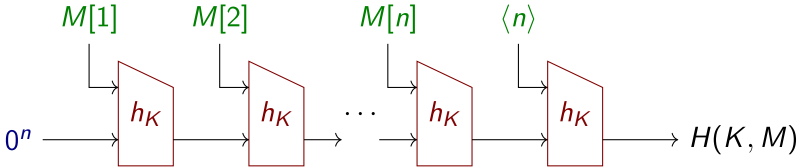
MD transform

Given: Compression function $h : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$.

Build: Hash function $H : \{0,1\}^k \times D \to \{0,1\}^n$.

Alg $H(K, M)$

$m \leftarrow |M|_b$; $M[m+1] \leftarrow \langle m \rangle$; $V[0] \leftarrow 0^n$
For $i = 1, \ldots, m+1$ do
    $V[i] \leftarrow h_K(M[i] \parallel V[i-1])$
Return $V[m+1]$



MD preserves CR

Theorem: MD preserves CR

Let $h : \{0,1\}^k \times \{0,1\}^{b+n} \to \{0,1\}^n$ be a family of functions and let $H : \{0,1\}^k \times D \to \{0,1\}^n$ be obtained from h via the MD transform. Given a cr-adversary A_H we can build a cr-adversary A_h such that

$$\mathbf{Adv}_H^{\mathrm{cr}}(A_H) \leq \mathbf{Adv}_h^{\mathrm{cr}}(A_h)$$

and the running time of A_h is that of A_H plus the time for computing H on the outputs of A_H.

Implications:

$$\begin{aligned} h \text{ CR} &\Rightarrow \mathbf{Adv}_h^{\mathrm{cr}}(A_h) \text{ small} \\ &\Rightarrow \mathbf{Adv}_H^{\mathrm{cr}}(A_H) \text{ small} \\ &\Rightarrow H \text{ CR} \end{aligned}$$

MD is great

Theorem: MD preserves collision resistance

If h is CR, then so is H.

- The problem of hashing long inputs has been reduced to the problem of hashing fixed-length inputs.
- There is no need to try to attack H. You won't find a weakness in it unless h has one. That is, H is *guaranteed* to be secure *assuming* h is secure.

MD is great

There is no need to try to attack H. You won't find a weakness in it unless h has one. That is, H is *guaranteed* to be secure *assuming* h is secure.

- For this reason, MD is the design used in many hash functions, including the MD and SHA2 series. SHA3 uses a different paradigm.
- However, MD is no silver bullet, especially for uses of hash functions that we will learn about later!

Plan

A candidate compression function

Let $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let us define the keyless compression function $h : \{0,1\}^{b+n} \to \{0,1\}^n$ by

$$h(x \parallel v) = E_x(v) \ .$$

Question: Is h collision resistant?

A candidate compression function

Let $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let us define the keyless compression function $h : \{0,1\}^{b+n} \to \{0,1\}^n$ by

$$h(x \parallel v) = E_x(v) \ .$$

Question: Is h collision resistant?

We seek an adversary that outputs distinct $x_1 \parallel v_1$, $x_2 \parallel v_2$ satisfying

$$E_{x_1}(v_1) = E_{x_2}(v_2) \ .$$

A candidate compression function

Let $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let us define the keyless compression function $h : \{0,1\}^{b+n} \to \{0,1\}^n$ by

$$h(x \parallel v) = E_x(v) \; .$$

Question: Is h collision resistant?

We seek an adversary that outputs distinct $x_1 \parallel v_1$, $x_2 \parallel v_2$ satisfying

$$E_{x_1}(v_1) = E_{x_2}(v_2) \; .$$

Answer: NO, h is NOT collision-resistant, because the following adversary A has $\mathbf{Adv}_h^{\mathrm{cr}}(A) = 1$:

adversary A
$x_1 \leftarrow 0^b$; $x_2 \leftarrow 1^b$;
$v_1 \leftarrow 0^n$; $y \leftarrow E_{x_1}(v_1)$; $v_2 \leftarrow E_{x_2}^{-1}(y)$
Return $x_1 \parallel v_1$, $x_2 \parallel v_2$

The Davies-Meyer compression function

Let $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let us define the keyless compression function $h : \{0,1\}^{b+n} \to \{0,1\}^n$ by

$$h(x \parallel v) = E_x(v) \oplus v \ .$$

Question: Is h collision resistant?

The Davies-Meyer compression function

Let $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let us define the keyless compression function $h : \{0,1\}^{b+n} \to \{0,1\}^n$ by

$$h(x \parallel v) = E_x(v) \oplus v \; .$$

Question: Is h collision resistant?

We seek an adversary that outputs distinct $x_1 \parallel v_1$, $x_2 \parallel v_2$ satisfying

$$E_{x_1}(v_1) \oplus v_1 = E_{x_2}(v_2) \oplus v_2 \; .$$

The Davies-Meyer compression function

Let $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let us define the keyless compression function $h : \{0,1\}^{b+n} \to \{0,1\}^n$ by

$$h(x \parallel v) = E_x(v) \oplus v \ .$$

Question: Is h collision resistant?

We seek an adversary that outputs distinct $x_1 \parallel v_1$, $x_2 \parallel v_2$ satisfying

$$E_{x_1}(v_1) \oplus v_1 = E_{x_2}(v_2) \oplus v_2 \ .$$

Answer: Unclear how to solve this equation, even though we can pick all four variables.

The Davies-Meyer compression function

Let $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let us define keyless compression function $h : \{0,1\}^{b+n} \to \{0,1\}^n$ by

$$h(x \parallel v) = E_x(v) \oplus v \ .$$

This is called the Davies-Meyer method and is used in the MD and SHA2 series of hash functions, modulo that $\oplus$ may be replaced by addition.

In particular the compression function sha256 of SHA256 is underlain in this way by the block cipher $\mathrm{E}^{\mathrm{sha256}} : \{0,1\}^{512} \times \{0,1\}^{256} \to \{0,1\}^{256}$ that we saw earlier, with $\oplus$ being replaced by component-wise addition modulo 2^{32}.

Plan

Hash functions

Collision resistance

Compression functions and the Merkle-Damgård (MD) transform

Hash functions from block ciphers: the Davies-Meyer method

Cryptanalytic attacks

Cryptanalytic attacks

So far we have looked at attacks that do not attempt to exploit the structure of h.

Can we get better attacks if we *do* exploit the structure?

Ideally not, but hash functions have fallen short!

Cryptanalytic attacks against hash functions

When	Against	Time	Who
1993,1996	md5	2^{16}	[dBBo,Do]
2004	MD5	1 hour	[WaFeLaYu]
2005,2006	MD5	1 minute	[LeWadW,Kl]
2005	SHA1	2^{69}	[WaYiYu]
2017	SHA1	$2^{63.1}$	[SBKAM]

Collisions found in compression function md5 of MD5 did not yield collisions for MD5, but collisions for MD5 are now easy.

2017: Google, Microsoft and Mozilla browsers stop accepting SHA1-based certificates.

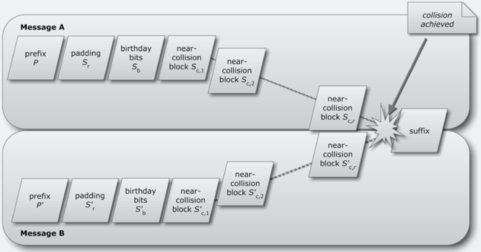
The SHA256 and SHA512 hash functions are still viewed as secure, meaning the best known attack is the birthday attack.

Flame exploited an MD5 attack

Crypto breakthrough shows Flame was designed by world-class scientists

The spy malware achieved an attack unlike any cryptographers have seen before.

DAN GOODIN - 6/7/2012, 11:20 AM



Marc Stevens

Enlarge / An overview of a chosen-prefix collision. A similar technique was used by the Flame espionage malware that targeted Iran. The scientific novelty of the malware underscored the sophistication of malware sponsored by wealthy nation states.

The Flame espionage malware that infected computers in Iran achieved mathematic breakthroughs that could only have been accomplished by world-class cryptographers, two of the world's foremost cryptography experts said.

"We have confirmed that Flame uses a yet unknown MD5 chosen-prefix collision attack," Marc Stevens wrote in an e-mail posted to a cryptography discussion group earlier this week. "The collision attack itself is very interesting from a scientific viewpoint, and there are already some practical implications." Benne de Weger, a Stevens colleague and another expert in cryptographic collision

Flame

Revealed: Stuxnet "beta's" devious alternate attack on Iran nuke program

Massive espionage malware targeting governments undetected for 5 years

Iranian computers targeted by new malicious data wiper program

New in-the-wild malware

SHA1 collision: `https://shattered.io/`

Here are two PDF files that display different content, yet have the same SHA-1 digest.

Academic cryptography vs. real-world security

MD5 was known to have weaknesses in the 1990s.
A full collision was computed in 2004.
People are still using MD5 now.

SHA-1's flaws have been known since 2005.
A full collision was computed in 2017.
Deprecation of SHA-1 has been slowed by intense resistance.

Linus Torvalds on Git's use of SHA-1: (2020: timid move towards SHA2-256)

```
I doubt the sky is falling for git as a source
control management tool. Do we want to migrate to another hash? Yes.
Is it "game over" for SHA1 like people want to say? Probably not.

I haven't seen the attack details, but I bet

 (a) the fact that we have a separate size encoding makes it much
harder to do on git objects in the first place

 (b) we can probably easily add some extra sanity checks to the opaque
data we do have, to make it much harder to do the hiding of random
data that these attacks pretty much always depend on.
```

Academic cryptography vs. real-world security

Problem: Deprecating weak algorithms or parameters breaks backwards compatibility.

Problem: Many people think they understand cryptography and can make their own security choices.

Problem: Cryptography is hard.

General Principle: Attacks get better. An "academic" break violating a theoretical definition of security may lead later on to a "real-world" vulnerability.