Foundations of Applied Cryptography

Adam O'Neill

Based on http://cseweb.ucsd.edu/~mihir/cse207/

Where we are

- We’ve seen a lower-level primitive (blockciphers) and a higher-level primitive (symmetric encryption)

Where we are

- We’ve seen a lower-level primitive (blockciphers) and a higher-level primitive (symmetric encryption)
- Now let’s see another lower-level primitive, hash functions

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:

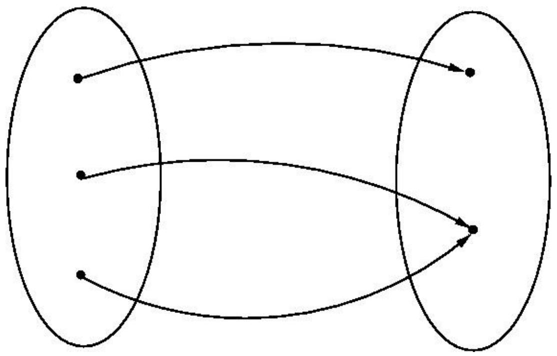
- Certificates: How you know `www.snapchat.com` really is Snapchat
- Bitcoin
- Data authentication with HMAC: TLS, ...

Collisions

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**.

The Formalism

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

Game CR_H	**procedure Finalize**(x_1, x_2)
	If $(x_1 = x_2)$ then return false
procedure Initialize	If $(x_1 \notin D$ or $x_2 \notin D)$ then return false
$K \xleftarrow{\$} \text{Keys}(H)$	Return $(H_K(x_1) = H_K(x_2))$
Return K	

Let

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

adversary gets K

The Formalism

Game CR_H	**procedure Finalize**(x_1, x_2)
procedure Initialize	If $(x_1 = x_2)$ then return false
$K \xleftarrow{\$}$ Keys	If $(x_1 \notin D$ or $x_2 \notin D)$ then return false
Return K	Return $(H_K(x_1) = H_K(x_2))$

The Return statement in **Initialize** means that the adversary A gets K as input. The key K here is not secret!

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?

Another example

$E = AES$

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

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

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

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

Another example

Let E: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher.
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 $\mathbf{Adv}_H^{\mathrm{cr}}(A) = 1$.

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

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

Another 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

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

adversary $A(K)$

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

return x_1, x_2

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

Note how we used the fact that A knows K and the fact that E is a blockcipher!

Keyless Hash Functions

We say that H: Keys $\times D \to R$ 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.

$E: \{0,1\}^{k+n} \to \{0,1\}^n$

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
$H_0^{(0)} \leftarrow$ `6a09e667` ; $H_1^{(0)} \leftarrow$ `bb67ae85` ; $\cdots$; $H_7^{(0)} \leftarrow$ `5be0cd19`
$H^{(0)} \leftarrow H_1^{(0)} H_2^{(0)} \cdots H_7^{(0)}$ // $|H_i^{(0)}| = 32$, $|H^{(0)}| = 256$
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**.

Underlying blockcipher

$\underline{\textbf{Alg } \mathrm{E}^{\mathrm{sha256}}(x, v)}$ // 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

$\quad T_1 \leftarrow S_7 + \gamma_1(S_4) + Ch(S_4, S_5, S_6) + C_t + W_t$

$\quad T_2 \leftarrow \gamma_0(S_0) + Maj(S_0, S_1, S_2)$

$\quad S_7 \leftarrow S_6 \;;\; S_6 \leftarrow S_5 \;;\; S_5 \leftarrow S_4 \;;\; S_4 \leftarrow S_3 + T_1$

$\quad S_3 \leftarrow S_2 \;;\; S_2 \leftarrow S_1 \;;\; S_1 \leftarrow S_0 \;;\; S_0 \leftarrow T_1 + T_2$

$S \leftarrow S_0 \cdots S_7$

Return S // 256-bit output

Internals

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.

Usage

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.

salt, Hash(pwd||salt)

Password authentication

- 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 how client authentication is done on the Internet, for example login to `gmail.com`.

Birthday 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 $\perp$

Interestingly, the analysis of this via the birthday problem is not trivial, but it shows 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.

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.

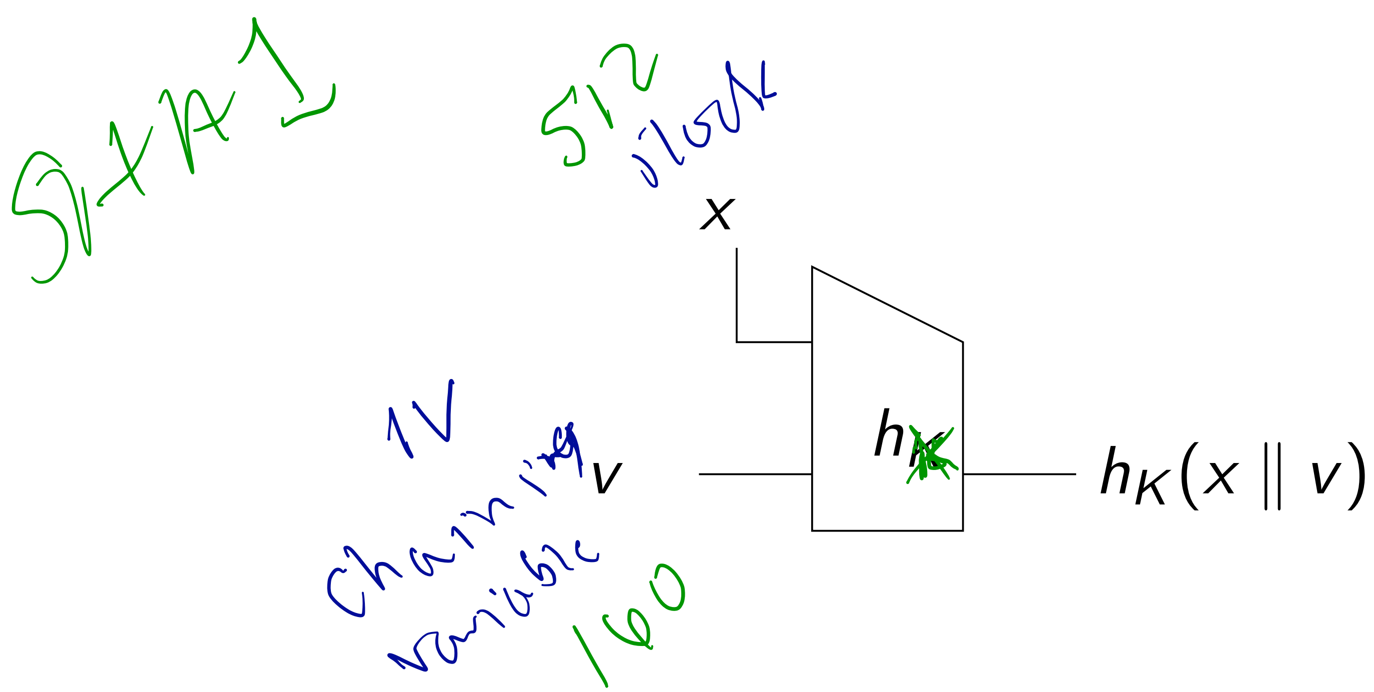
Design of hash functions aims to make the birthday attack the best collision-finding attack, meaning it is desired that there be no attack succeeding in time much less than T_B.

Compression Functions

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

E.g. $b = 512$ and $n = 160$, in which case

$$h : \{0,1\}^k \times \{0,1\}^{\boxed{672}} \to \{0,1\}^{160}$$



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

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

such that: $\boxed{\text{If } h \text{ is CR, then so is } H}$.

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.

For this reason, MD is the design used in many hash functions, including the MD and SHA2 series. SHA3 uses a different paradigm.

MD Setup

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

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

Since $M \in D$, its length $\ell = |M|$ is a multiple of the block length b. We let $\|M\|_b = |M|/b$ be the number of b-bit blocks in M, and parse as

$$M[1] \ldots M[\ell] \leftarrow M \ .$$

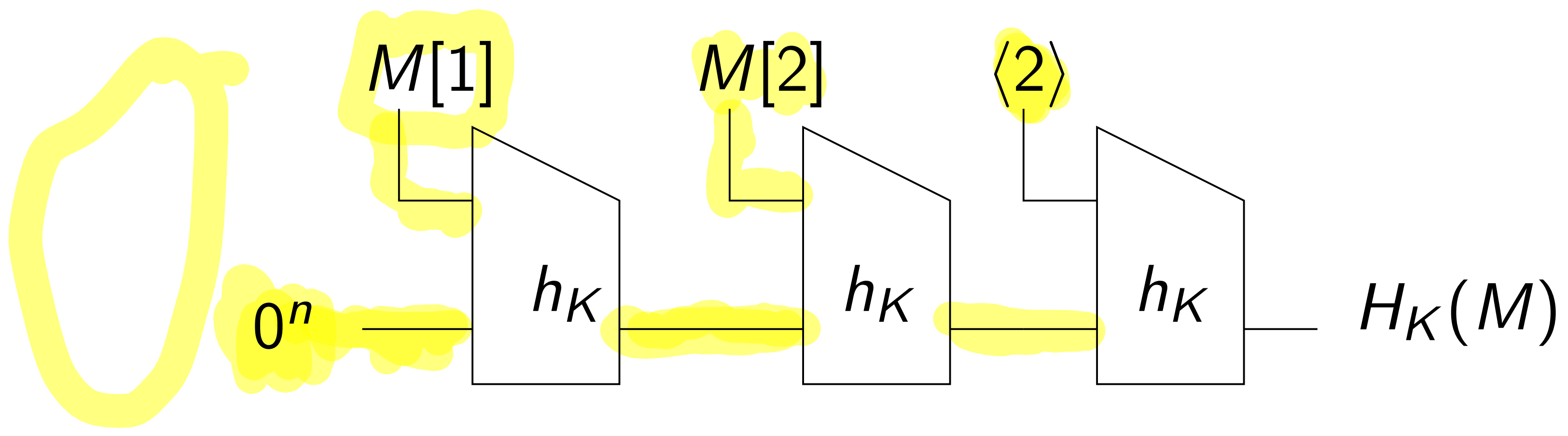
Let $\langle \ell \rangle$ denote the b-bit binary representation of $\ell \in \{0, \ldots, 2^b - 1\}$.

The Transform

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

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

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



MD preserves CR

Theorem: 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

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

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

Implication:

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

$$H(m) = H(m')$$

Wanted: CF from blockcipher

Compression functions

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

$$\boxed{h(x \| v) = E_x(v) \ .}$$

Question: Is h collision resistant?

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

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

Compression functions

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

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

Question: Is h collision resistant?

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

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

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

adversary A

$x_1 \leftarrow 0^b$; $x_2 \leftarrow 1^b$; $v_1 \leftarrow 0^n$; $y \leftarrow E_{x_1}(v_1)$; $v_2 \leftarrow E^{-1}_{x_2}(y)$
Return $x_1 \| v_1$, $x_2 \| v_2$

Compression functions

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

$$h(x \| v) = \underline{E_x(v) \oplus v} \ .$$

Question: Is h collision resistant?

Davies - Meyer

Compression functions

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

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

Question: Is h collision resistant?

We seek an adversary that outputs distinct $x_1 \| v_1$, $x_2 \| v_2$ 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.

Davies-Meyer

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

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

This is called the Davies-Meyer method and is used in the MD and SHA2 series of hash functions, modulo that the $\oplus$ 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 the $\oplus$ being replaced by component-wise addition modulo 2^{32}.

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

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.

`https://shattered.io/`.

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.

SHA1

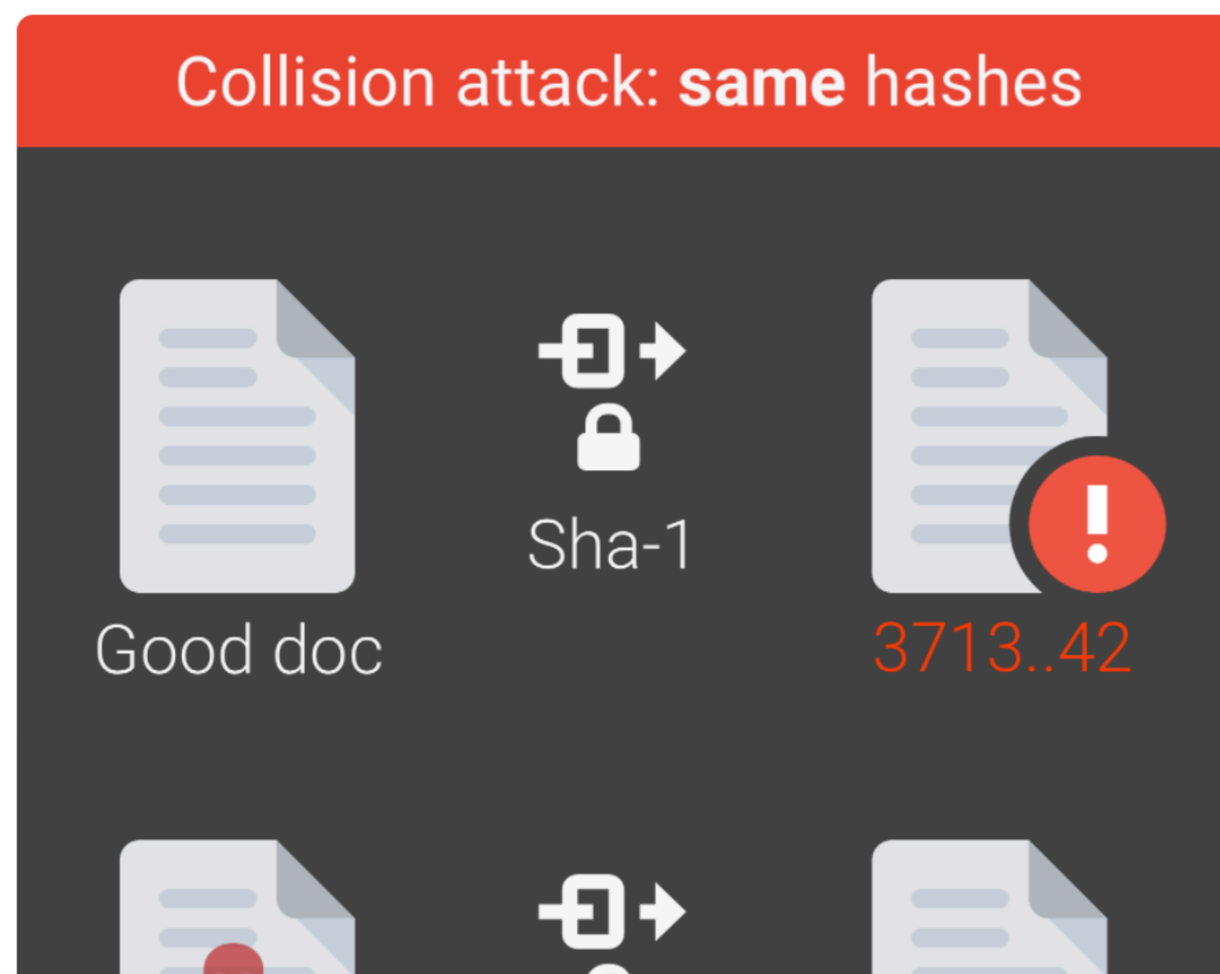
SHATTERED

We have broken SHA-1 in practice.

This industry cryptographic hash function standard is used for digital signatures and file integrity verification, and protects a wide spectrum of digital assets, including credit card transactions, electronic documents, open-source software repositories and software updates.

It is now practically possible to craft two colliding PDF files and obtain a SHA-1 digital signature on the first PDF file which can also be abused as a valid signature on the second PDF file.

For example, by crafting the two colliding PDF files

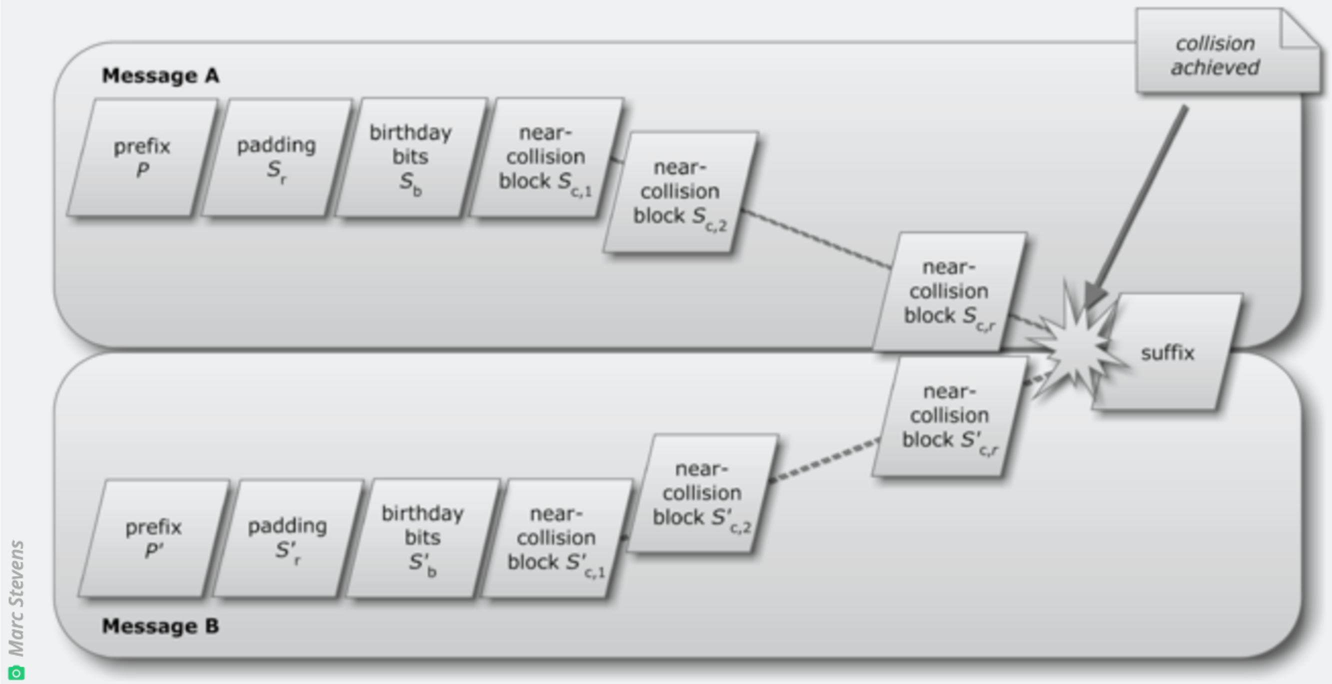


Flame exploited MD5

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



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

The tightrope

Why don't cryptographers build secure hash functions?

Cryptographers seem **perfectly capable** of building secure hash functions.

The difficulty is that they strive for VERY HIGH SPEED.

SHA256 can run at 3.5 cycles/byte (eBACS: 2018 Intel Core i3-8121U, https://bench.cr.yp.to/results-hash.html) or 0.6 ns per byte, and hardware will make it even faster.

It is AMAZING that one gets ANY security at such low cost.

If you allow cryptographers a 10x slowdown, they can up rounds by 10x and designs seem almost impossible to break.

SHA3

National Institute for Standards and Technology (NIST) held a world-wide competition to develop a new hash function standard.

Contest webpage:
`http://csrc.nist.gov/groups/ST/hash/index.html`

Requested parameters:

- Design: Family of functions with 224, 256, 384, 512 bit output sizes
- Security: CR, one-wayness, near-collision resistance, others...
- Efficiency: as fast or faster than SHA2-256

SHA3

Submissions: 64

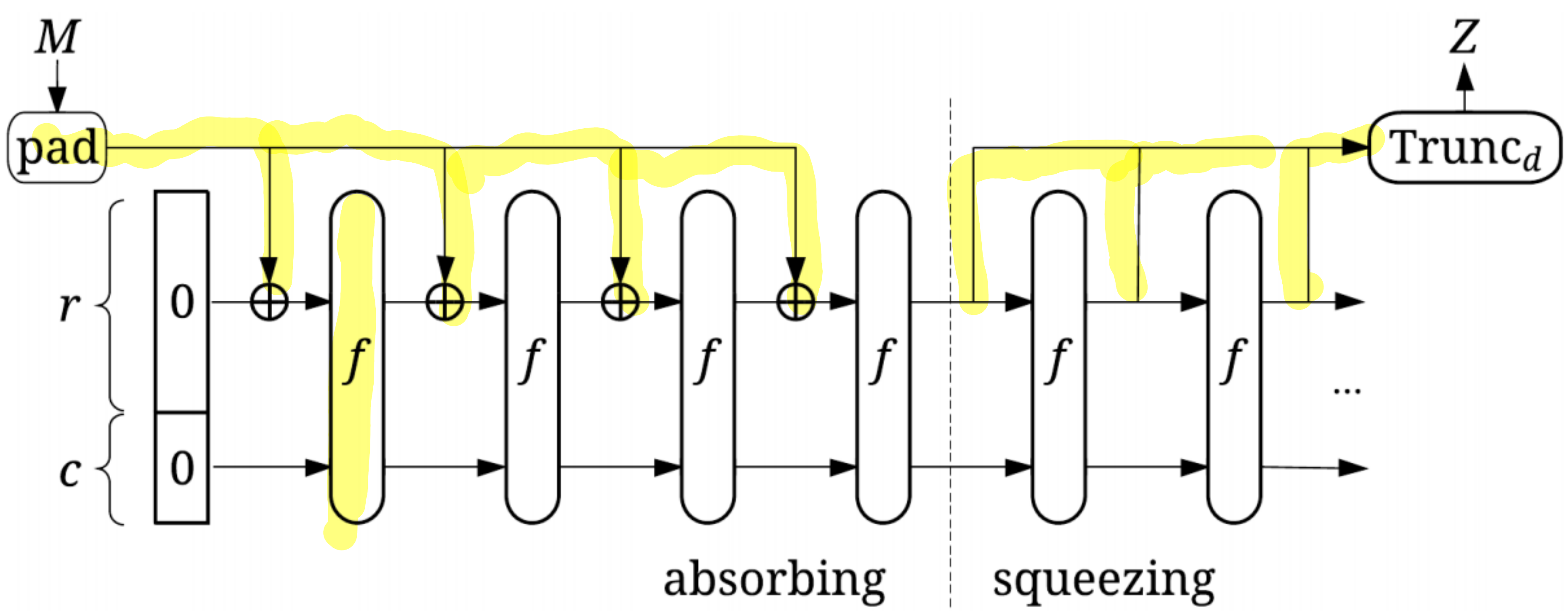
Round 1: 51

Round 2: 14: BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grostl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, Skein.

Finalists: 5: BLAKE, Grostl, JH, Keccak, Skein.

SHA3: 1: Keccak

SHA3: Sponge



$f\colon \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a (public, invertible!) permutation.
d is the number of output bits, and $c = 2d$.

SHA3 does not use the MD paradigm used by the MD and SHA2 series.

$\mathrm{Shake}(M, d)$— Extendable-output function, returning any given number d of bits.

A Hierarchy

- Universal hash functions
- Target collision-resistant hash functions
- Collision-resistant hash functions

Universal - Adversary does not get the key

$(x,y) \xleftarrow{\$} \mathcal{A}$ $\quad K \xleftarrow{\$} \mathcal{K}$ $\quad H_K(x) = H_K(y)$

$K = x \xleftarrow{\$} \mathbb{Z}_p$ $\quad p(x)$

Constructions of UHF

We start with a UHF construction using polynomials modulo a prime. Let ℓ be a (poly-bounded) length parameter and let p be a prime. We define a hash function H_{poly} that hashes a message $m \in \mathbb{Z}_p^{\leq \ell}$ to a single element $t \in \mathbb{Z}_p$. The key space is $\mathcal{K} := \mathbb{Z}_p$.

Let m be a message, so $m = (a_1, a_2, \ldots, a_v) \in \mathbb{Z}_p^{\leq \ell}$ for some $0 \leq v \leq \ell$. Let $k \in \mathbb{Z}_p$ be a key. The hash function $H_{\text{poly}}(k, m)$ is defined as follows:

$$H_{\text{poly}}\big(k,\ (a_1, \ldots, a_v)\big) := k^v + a_1 k^{v-1} + a_2 k^{v-2} + \cdots + a_{v-1} k + a_v \in \mathbb{Z}_p \tag{7.3}$$