

\* Hash function is one of the atomic cryptographic primitives

\*

## HASH FUNCTIONS

Mihir Bellare

UCSD

1

## 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



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."

UCSD

2

## Hash functions

\* they are used to map long input to short output and yet difficult for the adversary to find the true input for the same output.

- MD: MD4, [MD5](#), MD6
- SHA2: SHA1, SHA224, [SHA256](#), SHA384, [SHA512](#) The no is output bits
- 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](http://www.snapchat.com) really is Snapchat
- Bitcoin
- Data authentication with HMAC: TLS, ...

## 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](http://www.snapchat.com) really is Snapchat
- Bitcoin
- Data authentication with HMAC: TLS, ...

SHA = “Secure Hash Algorithm” 😊

Mihir Bellare

UCSD

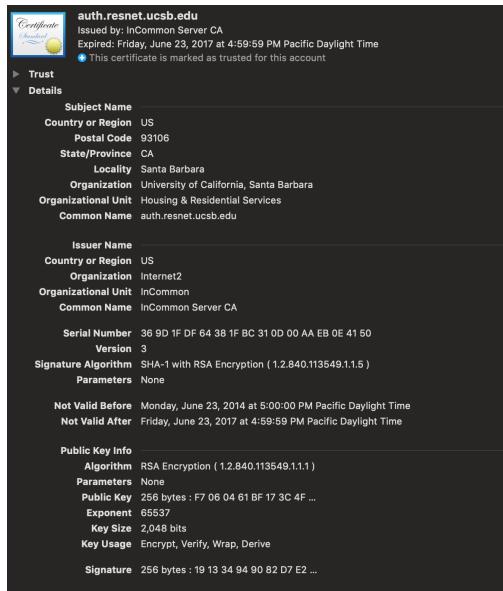
3

Mihir Bellare

UCSD

4

## A SHA1 certificate



Mihir Bellare

UCSD

5

## SHA1 certificates no longer

May 5, 2016

### Microsoft will cease support for TLS certs signed by SHA1

Greg Masters Managing Editor

Follow @https://twitter.com/gregmasters21



Microsoft browsers will no longer display a lock when on HTTPS sites protected by SHA1 certs.

Microsoft announced it will soon cease support for TLS certificates signed by the SHA1 hashing algorithm, according to ArsTechnica.

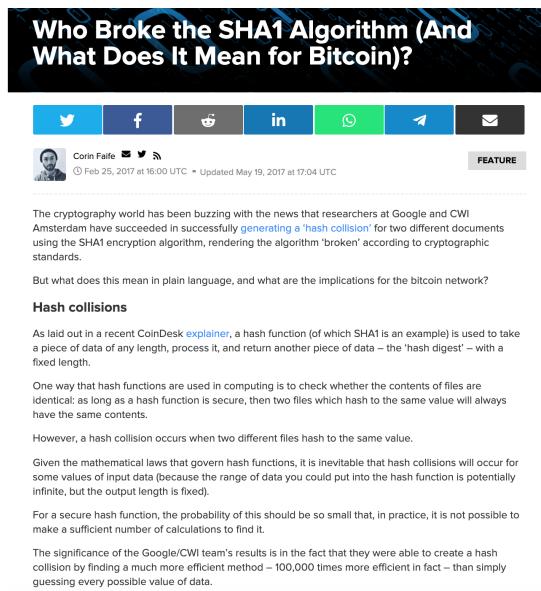
After hinting in November that it might, the tech giant made it official last week. The end was expected following new research that revealed the popular cryptographic algorithm was susceptible to collision attacks – in which miscreants attempt to find two inputs producing the same hash value. Should they succeed, they would be able to forge digital

signatures.

UCSD

6

## Implications for Bitcoin?



Mihir Bellare

UCSD

7

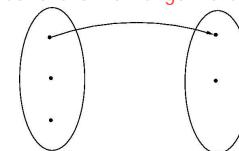
## Collisions

A **collision** for a function  $h : D \rightarrow \{0, 1\}^n$  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  $h$ .

As long as the domain that has the size larger than the range, collision will mathematically exists.



Mihir Bellare

UCSD

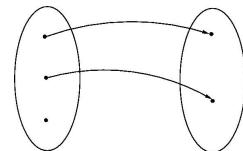
8

## Collisions

A **collision** for a function  $h : D \rightarrow \{0,1\}^n$  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  $h$ .



## Collision-resistance of a function family

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

Game CR <sub>H</sub>	<b>procedure</b> Finalize( $x_1, x_2$ )
<b>procedure</b> Initialize	If ( $x_1 = x_2$ ) then return false
$K \leftarrow \text{Keys}$	If ( $x_1 \notin D$ or $x_2 \notin D$ ) then return false
Return $K$	Return ( $H_K(x_1) = H_K(x_2)$ )

Let

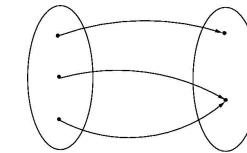
$$\mathbf{Adv}_H^{\text{cr}}(A) = \Pr_{\{\text{advantage}\}} [\text{CR}_H^A \Rightarrow \text{true}] .$$

## Collisions

A **collision** for a function  $h : D \rightarrow \{0,1\}^n$  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

Game CR<sub>H</sub>

**procedure** Initialize  
 $K \leftarrow \text{Keys}$   
 Return  $K$

**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 ( $H_K(x_1) = H_K(x_2)$ )

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

(example of finding collision in Hash functions)

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

$$H: \underbrace{\{1, \dots, N\}}_{\text{Keys}} \times \underbrace{\{0, 1, 2, \dots\}}_D \rightarrow \underbrace{\{0, 1, \dots, N-1\}}_R$$

{infinite domain}

by

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

**Q:** Is  $H$  collision resistant?

## Example

Let  $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a blockcipher.

Let  $H: \{0, 1\}^k \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$  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  $\mathbf{Adv}_H^{\text{cr}}(A) = 1$ .

## Example

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

$$H: \underbrace{\{1, \dots, N\}}_{\text{Keys}} \times \underbrace{\{0, 1, 2, \dots\}}_D \rightarrow \underbrace{\{0, 1, \dots, N-1\}}_R$$

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  $x_1, x_2$

$$\mathbf{Adv}_H^{\text{cr}}(A) = 1$$

## Example

Let  $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be a blockcipher.

Let  $H: \{0, 1\}^k \times \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$  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  $\mathbf{Adv}_H^{\text{cr}}(A) = 1$ .

**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

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

**adversary**  $A(K)$

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

Then  $\text{Adv}_H^{\text{cr}}(A) = 1$  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!

## Exercise

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

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

**Alg**  $H(K, M)$

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

$C[0] \leftarrow 0^l$

For  $i = 1, \dots, n$  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  $\text{Adv}_H^{\text{cr}}(A) = 1$ .

## Keyless hash functions

We say that  $H: \text{Keys} \times D \rightarrow R$  is **keyless** if  $\text{Keys} = \{\varepsilon\}$  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.

## SHA256

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

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

**Alg**  $\text{SHA256}(X) \quad // |X| < 2^{64}$

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

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

$H_0^{(0)} \leftarrow \underline{6a09e6677}; H_1^{(0)} \leftarrow \underline{bb67ae85}; \dots; H_7^{(0)} \leftarrow \underline{5be0cd19} \quad \{\text{underline ones are the const}\}$

$H^{(0)} \leftarrow H_1^{(0)}H_2^{(0)}\dots H_7^{(0)} \quad // |H_i^{(0)}| = 32, |H^{(0)}| = 256$

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

Return  $H^{(n)}$

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

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

```
Alg shapad( $X$ ) //  $|X| < 2^{64}$ 
 $d \leftarrow (447 - |X|) \text{ mod } 512$  // Chosen to make  $|M|$  a multiple of 512
Let  $\ell$  be the 64-bit binary representation of  $|M|$ 
 $M \leftarrow X \parallel 1 \parallel 0^d \parallel \ell$  //  $|M|$  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 ( $0 \leq j \leq 7$ ).

## Compression function sha256

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

```
Alg sha256( $x \parallel v$ ) //  $X=512, v=256$ 
 $w \leftarrow E^{\text{sha256}}(x, v)$ 
 $w_0 \dots w_7 \leftarrow w$  // Break  $w$  into 32-bit words
 $v_0 \dots v_7 \leftarrow v$  // Break  $v$  into 32-bit words
For  $j = 0, \dots, 7$  do  $h_j \leftarrow w_j + v_j$ 
 $h \leftarrow h_0 \dots h_7$  //  $|h| = 256$ 
Return  $h$ 
```

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

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

## Block cipher $E^{\text{sha256}}$

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

## Internals of block cipher $E^{\text{sha256}}$

On the previous slide:

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

## SHA256 hash calculator

<http://www.xorbin.com/tools/sha256-hash-calculator>

SHA-256 produces a 256-bit (32-byte) hash value.

### Data

CSE 107 is way too easy!

### SHA-256 hash

7263ee434edb9568b9c70b580465f49923eb2c39677a9c862d536a42798db96f

Calculate SHA256 hash

Mihir Bellare

UCSD

25

Mihir Bellare

UCSD

26

## Authentication via passwords

How it works when you type password in your gmail account??

\* i.e. how would the password go to the network?

- 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)**

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

\* The channel is already secured using TLS. You already have a connection secured by the server certificate and everything sent over would be encrypted by session key.

**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, ...**

## Usage of hash functions

\* The important uses of Hash function is during the creation of CERTIFICATES.

\* why are we talking about collision resistant hashing function?? ==> incoming slide

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.

## Hashed passwords

\* The servers could be compromised and can lead to leak of password. So its not the password that is actually being stored but its a Hash value of it.

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

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

\* If your hash function is good, you can't reverse the operation and get the plain password out from the hashed one.

Server compromise results in attacker getting  $\bar{PW}$  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](http://gmail.com). {OK}

\* collision resistance implies one-wayness

Mihir Bellare

UCSD

27

Mihir Bellare

UCSD

28

## Birthday collision-finding attack

What is birthday paradox in collision finding attack?

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

with  $q$  # of trials, the attacker has probability  $q^2$  to find the collision  
\* even though the effort is proportional to  $q$ , the success is proportional to  $q^2$

**adversary  $A_q(K)$**

for  $i = 1, \dots, q$  do  $x_i \leftarrow D$ ;  $y_i \leftarrow H_K(x_i)$   
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

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

So a collision can usually be found in about  $q = \sqrt{2^n}$  trials, where  $n$  is total input bits

## Birthday attack times

Function	$n$	$T_B$	{output}{}
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	$2^{256}$	

\* You can't avoid the birthday attack. But the goal in designing the Hash function is, no better than birthday attack is possible

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

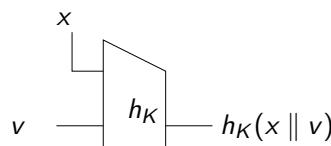
Design of hash functions aims to make the birthday attack the best collision-finding attack, meaning it is desired that there be no attack succeeding in time much less than  $T_B$ .

## Compression functions

A **compression function** is a family  $h : \{0,1\}^k \times \{0,1\}^{b+n} \rightarrow \{0,1\}^n$  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

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



## The MD transform

Let  $h : \{0,1\}^k \times \{0,1\}^{b+n} \rightarrow \{0,1\}^n$  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

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

such that: 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.

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

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

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

## MD preserves CR

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

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

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

**Implication:**

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

## MD transform

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

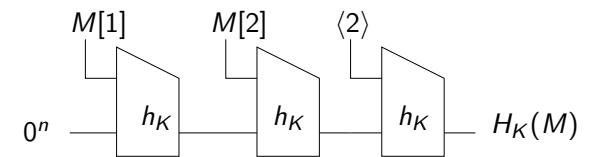
**Build:** Hash function  $H : \{0,1\}^k \times D \rightarrow \{0,1\}^n$ .

Algorithm  $H_K(M)$

$$m \leftarrow \|M\|_b ; M[m+1] \leftarrow \langle m \rangle ; V[0] \leftarrow 0^n$$

For  $i = 1, \dots, m+1$  do  $v[i] \leftarrow h_K(M[i]||V[i-1])$

Return  $V[m+1]$



## How are compression functions designed?

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

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

**Question:** Is  $h$  collision resistant?

## How are compression functions designed?

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

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

## How are compression functions designed?

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

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

**Question:** Is  $h$  collision resistant?

## How are compression functions designed?

Let  $E : \{0,1\}^b \times \{0,1\}^n \rightarrow \{0,1\}^n$  be a block cipher. Let us define keyless compression function  $h : \{0,1\}^{b+n} \rightarrow \{0,1\}^n$  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  $\text{Adv}_h^{\text{cr}}(A) = 1$ :

adversary  $A$

$$\begin{aligned} x_1 &\leftarrow 0^b; x_2 \leftarrow 1^b; v_1 \leftarrow 0^n; y \leftarrow E_{x_1}(v_1); v_2 \leftarrow E_{x_2}^{-1}(y) \\ \text{Return } x_1\|v_1, x_2\|v_2 \end{aligned}$$

## How are compression functions designed?

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

## The Davies-Meyer method

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

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

Mihir Bellare

UCSD

41

## 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,KI]
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.

Mihir Bellare

UCSD

43

## 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!

Mihir Bellare

UCSD

41

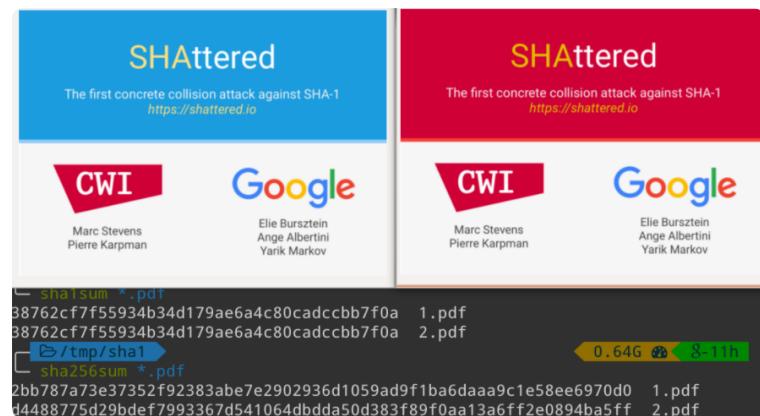
Mihir Bellare

UCSD

42

## SHA1 collision

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



Mihir Bellare

UCSD

43

Mihir Bellare

UCSD

44

## 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

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.

**Flame**  
Revealed: Stuxnet "beta's" devious alternate attack on Iran nuke program  
Massive espionage malware family undetected for 5 years  
Iranian computers targeted by new malicious data wiper Program  
New in-the-wild malware

Marc Stevens

Enlarge

[Read more](#)

[Email](#) [Facebook](#) [Twitter](#)

Mihir Bellare

UCSD

45

## Cryptographer job-performance evaluation

Why don't cryptographers build secure hash functions?

## Cryptographer job-performance evaluation

Why don't cryptographers build secure hash functions?

Assess their job performance in light of attacks by selecting a grade below:

- A** – Cryptographers are doing super well
- B** – They are OK
- C** – They suck
- F** – Just fire them all and give the job to AI

Mihir Bellare

UCSD

47

Mihir Bellare

UCSD

48

## Cryptographers' tightrope

Why don't cryptographers build secure hash functions?

## Cryptographers' 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.

Mihir Bellare

UCSD

49

## SHA3

**Submissions:** 64

**Round 1:** 51

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

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

**SHA3:** 1: Keccak

## 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

Mihir Bellare

UCSD

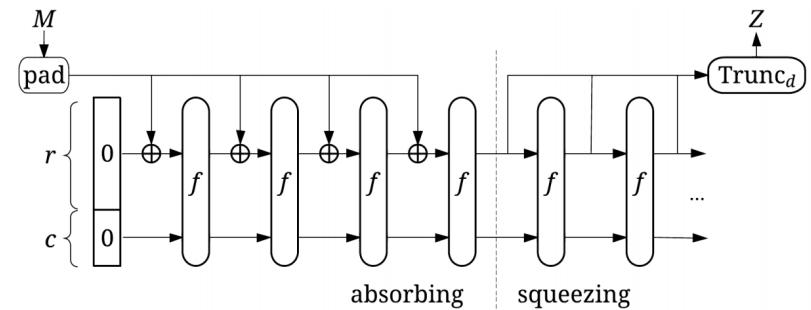
51

Mihir Bellare

UCSD

50

## SHA3: The Sponge construction



$f: \{0, 1\}^{r+c} \rightarrow \{0, 1\}^{r+c}$  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.

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

Mihir Bellare

UCSD

52