

# HASH FUNCTIONS

Hash functions like MD5, SHA1, SHA256, SHA512, SHA3, ... are amongst the most widely-used cryptographic primitives.

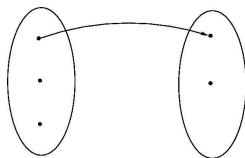
Their primary purpose is collision-resistant data compression, but they have many other purposes and properties as well.

A good hash function is often treated like a magic wand ...

# Collision resistance (CR)

**Definition:** A **collision** for a function  $h : D \rightarrow \{0, 1\}^n$  is a pair  $x_1, x_2 \in D$  of points such that  $h(x_1) = h(x_2)$  but  $x_1 \neq x_2$ .

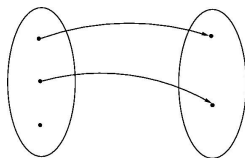
If  $|D| > 2^n$  then the pigeonhole principle tells us that there must exist a collision for  $h$ .



# Collision resistance (CR)

**Definition:** A **collision** for a function  $h : D \rightarrow \{0, 1\}^n$  is a pair  $x_1, x_2 \in D$  of points such that  $h(x_1) = h(x_2)$  but  $x_1 \neq x_2$ .

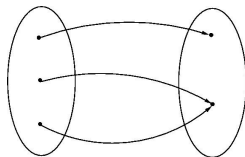
If  $|D| > 2^n$  then the pigeonhole principle tells us that there must exist a collision for  $h$ .



# Collision resistance (CR)

**Definition:** A **collision** for a function  $h : D \rightarrow \{0, 1\}^n$  is a pair  $x_1, x_2 \in D$  of points such that  $h(x_1) = h(x_2)$  but  $x_1 \neq x_2$ .

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

Game $\text{CR}_H$	<b>procedure Finalize</b> ( $x_1, x_2$ )
<b>procedure Initialize</b>	If ( $x_1 = x_2$ ) then return false
$K \xleftarrow{\$} \text{Keys}(H)$	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

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

Game $\text{CR}_H$	<b>procedure Finalize</b> ( $x_1, x_2$ )
<b>procedure Initialize</b>	If ( $x_1 = x_2$ ) then return false
$K \xleftarrow{\$} \text{Keys}(H)$	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  $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  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  $E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  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**  $A(K)$

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

Then  $\mathbf{Adv}_H^{\text{cr}}(A) = 1$  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  $H: \text{Keys}(H) \times D \rightarrow R$  is **keyless** if  $\text{Keys}(H) = \{\varepsilon\}$  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 MD5, SHA1, SHA256, SHA512, SHA3, ... are keyless.

# SHA1

**Alg** SHA1( $M$ ) //  $|M| < 2^{64}$

$V \leftarrow \text{SHF1}(5A827999 \parallel 6ED9EBA1 \parallel 8F1BBCDC \parallel CA62C1D6, M)$   
return  $V$

---

**Alg** SHF1( $K, M$ ) //  $|K| = 128$  and  $|M| < 2^{64}$

$y \leftarrow \text{shapad}(M)$   
Parse  $y$  as  $M_1 \parallel M_2 \parallel \dots \parallel M_n$  where  $|M_i| = 512$  ( $1 \leq i \leq n$ )  
 $V \leftarrow 67452301 \parallel \text{EFC DAB89} \parallel 98\text{BADCFE} \parallel 10325476 \parallel \text{C3D2E1F0}$   
for  $i = 1, \dots, n$  do  $V \leftarrow \text{shf1}(K, M_i \parallel V)$   
return  $V$

---

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

$d \leftarrow (447 - |M|) \bmod 512$   
Let  $\ell$  be the 64-bit binary representation of  $|M|$   
 $y \leftarrow M \parallel 1 \parallel 0^d \parallel \ell$  //  $|y|$  is a multiple of 512  
return  $y$

**Alg** shf1( $K, B \parallel V$ ) //  $|K| = 128, |B| = 512$  and  $|V| = 160$

Parse  $B$  as  $W_0 \parallel W_1 \parallel \dots \parallel W_{15}$  where  $|W_i| = 32$  ( $0 \leq i \leq 15$ )

Parse  $V$  as  $V_0 \parallel V_1 \parallel \dots \parallel V_4$  where  $|V_i| = 32$  ( $0 \leq i \leq 4$ )

Parse  $K$  as  $K_0 \parallel K_1 \parallel K_2 \parallel K_3$  where  $|K_i| = 32$  ( $0 \leq i \leq 3$ )

for  $t = 16$  to  $79$  do  $W_t \leftarrow \text{ROTL}^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$

$A \leftarrow V_0; B \leftarrow V_1; C \leftarrow V_2; D \leftarrow V_3; E \leftarrow V_4$

for  $t = 0$  to  $19$  do  $L_t \leftarrow K_0; L_{t+20} \leftarrow K_1; L_{t+40} \leftarrow K_2; L_{t+60} \leftarrow K_3$

for  $t = 0$  to  $79$  do

- if ( $0 \leq t \leq 19$ ) then  $f \leftarrow (B \wedge C) \vee ((\neg B) \wedge D)$
- if ( $20 \leq t \leq 39$  OR  $60 \leq t \leq 79$ ) then  $f \leftarrow B \oplus C \oplus D$
- if ( $40 \leq t \leq 59$ ) then  $f \leftarrow (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
- $temp \leftarrow \text{ROTL}^5(A) + f + E + W_t + L_t$
- $E \leftarrow D; D \leftarrow C; C \leftarrow \text{ROTL}^{30}(B); B \leftarrow A; A \leftarrow temp$

$V_0 \leftarrow V_0 + A; V_1 \leftarrow V_1 + B; V_2 \leftarrow V_2 + C; V_3 \leftarrow V_3 + D; V_4 \leftarrow V_4 + E$

$V \leftarrow V_0 \parallel V_1 \parallel V_2 \parallel V_3 \parallel V_4$ ; return  $V$

<http://www.sha1-online.com/>

Home Page | [SHA1 in JAVA](#) | [Secure password generator](#) | [Linux](#)

## SHA1 and other hash functions online generator

CSE 107 is cool

sha-1

Result for sha1: **0186cbeb87dc60c55cacd79168d242c9b04e21e6**

# Applications of hash functions

The killer-app is hashing before digitally signing, but we'll discuss a few others:

- primitive in cryptographic schemes
- tool for security applications
- tool for non-security applications



# Password verification

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

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

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

# Password verification

- 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  $H(PW) = \overline{PW}$

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

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

But we will revisit this when we consider dictionary attacks!

# File comparison

- $A$  has a large file  $F_A$  and  $B$  has a large file  $F_B$ . For example, music collections.
- They want to know whether  $F_A = F_B$
- $A$  sends  $F_A$  to  $B$  and  $B$  checks whether  $F_A = F_B$

$$A^{F_A} \xrightarrow{F_A} B^{F_B}$$

**Problem:** Transmission could be slow.

# Compare-by-hash

- $A$  has a large file  $F_A$  and  $B$  has a large file  $F_B$  and they want to know whether  $F_A = F_B$
- $A$  computes  $h_A = H(F_A)$  and sends it to  $B$ , and  $B$  checks whether  $h_A = H(F_B)$ .

$$A^{F_A} \xrightarrow{h_A} B^{F_B}$$

Collision-resistance of  $H$  guarantees that  $B$  does not accept if  $F_A \neq F_B$ !

An executable may be available at lots of sites  $S_1, S_2, \dots, S_N$ . Which one can you trust?

- Provide a safe way to get the hash  $h = H(X)$  of the correct executable  $X$ .
- Download an executable from anywhere, and check hash.

# Birthday attacks

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 finds a collision with probability about

$$\frac{q^2}{2^{n+1}}.$$

So a collision can be found in about  $q = \sqrt{2^{n+1}} \approx 2^{n/2}$  trials.

## Recall Birthday Problem

for  $i = 1, \dots, q$  do  $y_i \xleftarrow{\$} \{0, 1\}^n$   
if  $\exists i, j$  ( $i \neq j$  and  $y_i = y_j$ ) then COLL  $\leftarrow$  true

$$\begin{aligned}\Pr[\text{COLL}] &= C(2^n, q) \\ &\approx \frac{q^2}{2^{n+1}}\end{aligned}$$

# Birthday attack

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

**adversary**  $A(K)$

for  $i = 1, \dots, q$  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 FAIL



# Analysis of birthday attack

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

**adversary**  $A(K)$

for  $i = 1, \dots, q$  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 FAIL

What is the probability that this attack finds a collision?

**adversary**  $A(K)$

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

if  $\exists i, j$  ( $i \neq j$  and  $y_i = y_j$ ) then COLL  $\leftarrow$  true

We have dropped things that don't much affect the advantage and focused on success probability. So we want to know what is

$\Pr[\text{COLL}]$  .

# Analysis of birthday attack

## Birthday

for  $i = 1, \dots, q$  do  
   $y_i \xleftarrow{\$} \{0, 1\}^n$   
if  $\exists i, j$  ( $i \neq j$  and  $y_i = y_j$ ) then  
  COLL  $\leftarrow$  true

$$\Pr[\text{COLL}] = C(2^n, q)$$

## Adversary $A$

for  $i = 1, \dots, q$  do  
   $x_i \xleftarrow{\$} D$ ;  $y_i \leftarrow H_K(x_i)$   
if  $\exists i, j$  ( $i \neq j$  and  $y_i = y_j$ ) then  
  COLL  $\leftarrow$  true

$$\Pr[\text{COLL}] = ?$$

Are the two collision probabilities the same?

# Analysis of birthday attack

## Birthday

for  $i = 1, \dots, q$  do  
   $y_i \xleftarrow{\$} \{0, 1\}^n$   
if  $\exists i, j$  ( $i \neq j$  and  $y_i = y_j$ ) then  
  COLL  $\leftarrow$  true

$$\Pr[\text{COLL}] = C(2^n, q)$$

## Adversary A

for  $i = 1, \dots, q$  do  
   $x_i \xleftarrow{\$} D$ ;  $y_i \leftarrow H_K(x_i)$   
if  $\exists i, j$  ( $i \neq j$  and  $y_i = y_j$ ) then  
  COLL  $\leftarrow$  true

$$\Pr[\text{COLL}] = ?$$

Are the two collision probabilities the same?

Not necessarily, because

- on the left  $y_i \xleftarrow{\$} \{0, 1\}^n$
- on the right  $x_i \xleftarrow{\$} D$ ;  $y_i \leftarrow H_K(x_i)$

# Analysis of birthday attack

We say that  $H : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$  is regular if every range point has the same number of pre-images under  $H_K$ . That is if we let

$$H_K^{-1}(y) = \{x \in D : H_K(x) = y\}$$

then  $H$  is regular if

$$|H_K^{-1}(y)| = \frac{|D|}{2^n}$$

for all  $K$  and  $y$ . In this case the following processes both result in a random output

Process 1	Process 2
$y \xleftarrow{\$} \{0, 1\}^n$	$x \xleftarrow{\$} D; y \xleftarrow{\$} H_K(x)$
return $y$	return $y$

# Analysis of birthday attack

If  $H: \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$  is regular then the birthday attack finds a collision in about  $2^{n/2}$  trials.

# Analysis of birthday attack

If  $H: \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$  is regular then the birthday attack finds a collision in about  $2^{n/2}$  trials.

If  $H$  is **not** regular, the attack may succeed **sooner**.

So we want functions to be “close to regular”.

It seems MD4, MD5, SHA1, SHA2, SHA3,... have this property.

# Birthday attack times

Function	$n$	$T_B$
MD4	128	$2^{64}$
MD5	128	$2^{64}$
SHA1	160	$2^{80}$
SHA2-256	256	$2^{128}$
SHA2-512	512	$2^{256}$
SHA3-256	256	$2^{128}$
SHA3-512	512	$2^{256}$

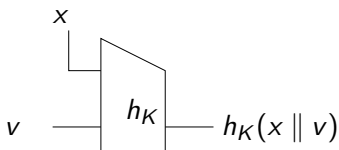
$T_B$  is the number of trials to find collisions via a birthday attack.

# Compression functions

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

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





Design principle: To build a CR hash function

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

where  $D = \{0, 1\}^{\leq 2^{64}}$ :

- First build a CR **compression** function  
 $h : \{0, 1\}^k \times \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$ .
- Appropriately iterate  $h$  to get  $H$ , using  $h$  to hash block-by-block.

Assume for simplicity that  $|M|$  is a multiple of  $b$ . Let

- $\|M\|_b$  be the number of  $b$ -bit blocks in  $M$ , and write  $M = M[1] \dots M[\ell]$  where  $\ell = \|M\|_b$ .
- $\langle i \rangle$  denote the  $b$ -bit binary representation of  $i \in \{0, \dots, 2^b - 1\}$ .
- $D$  be the set of all strings of at most  $2^b - 1$  blocks, so that  $\|M\|_b \in \{0, \dots, 2^b - 1\}$  for any  $M \in D$ , and thus  $\|M\|_b$  can be encoded as above.

# MD transform

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

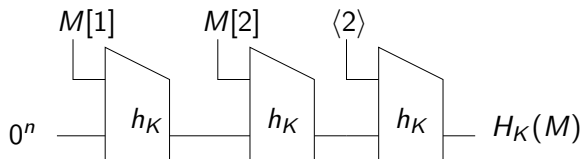
**Build:** Hash function  $H : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$ .

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  $V[m+1]$



# MD preserves CR

Assume

- $h$  is CR
- $H$  is built from  $h$  using MD

Then

- $H$  is CR too!

This means

- No need to attack  $H$ ! You won't find a weakness in it unless  $h$  has one
- $H$  is guaranteed to be secure assuming  $h$  is.

For this reason, MD is the design used in many current hash functions. Newer hash functions use other iteration methods with analogous properties.

**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:**

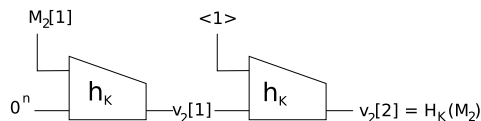
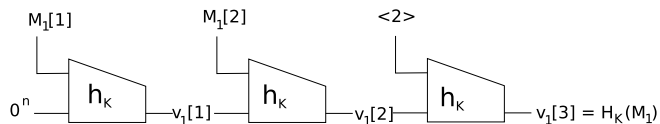
$$h \text{ CR} \Rightarrow \mathbf{Adv}_h^{\text{cr}}(A_h) \text{ small}$$

$$\Rightarrow \mathbf{Adv}_H^{\text{cr}}(A_H) \text{ small}$$

$$\Rightarrow H \text{ CR}$$

Let  $(M_1, M_2)$  be the  $H_K$ -collision returned by  $A_H$ . The  $A_h$  will trace the chains backwards to find an  $h_k$ -collision.

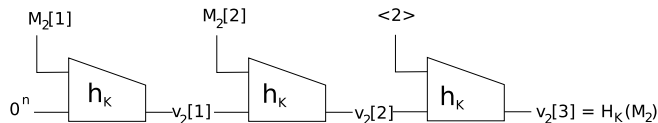
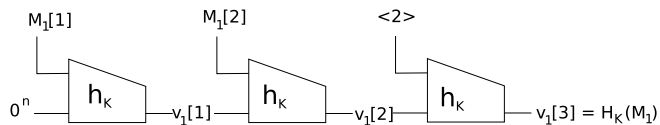
# Case 1: $\|M_1\|_b \neq \|M_2\|_b$



Let  $x_1 = \langle 2 \rangle \| v_1[2]$  and  $x_2 = \langle 1 \rangle \| v_2[1]$ . Then

- $h_K(x_1) = h_K(x_2)$  because  $H_K(M_1) = H_K(M_2)$ .
- But  $x_1 \neq x_2$  because  $\langle 1 \rangle \neq \langle 2 \rangle$ .

## Case 2: $\|M_1\|_b = \|M_2\|_b$

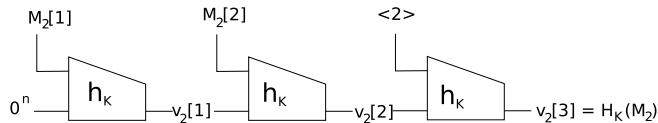
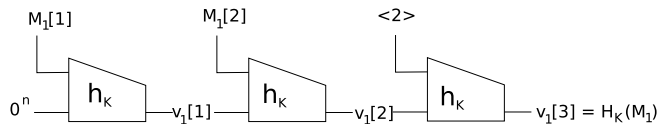


$x_1 \leftarrow \langle 2 \rangle \| v_1[2]$ ;  $x_2 \leftarrow \langle 2 \rangle \| v_2[2]$

If  $x_1 \neq x_2$  then return  $x_1, x_2$



## Case 2: $\|M_1\|_b = \|M_2\|_b$

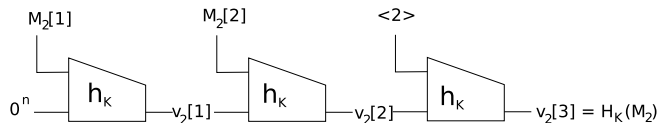
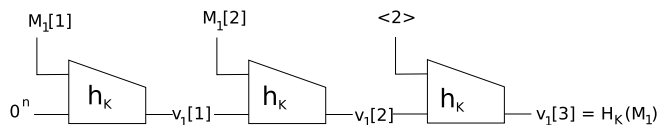


$x_1 \leftarrow \langle 2 \rangle \| V_1[2]$ ;  $x_2 \leftarrow \langle 2 \rangle \| V_2[2]$

If  $x_1 \neq x_2$  then return  $x_1, x_2$

Else //  $V_1[2] = V_2[2]$

## Case 2: $\|M_1\|_b = \|M_2\|_b$



$x_1 \leftarrow \langle 2 \rangle \| V_1[2]$ ;  $x_2 \leftarrow \langle 2 \rangle \| V_2[2]$

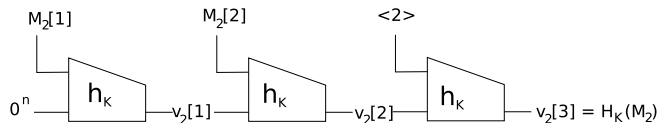
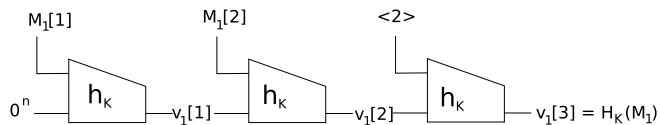
If  $x_1 \neq x_2$  then return  $x_1, x_2$

Else //  $V_1[2] = V_2[2]$

$x_1 \leftarrow M_1[2] \| V_1[1]$ ;  $x_2 \leftarrow M_2[2] \| V_2[1]$

If  $x_1 \neq x_2$  then return  $x_1, x_2$

## Case 2: $\|M_1\|_b = \|M_2\|_b$



$x_1 \leftarrow \langle 2 \rangle \| V_1[2]$ ;  $x_2 \leftarrow \langle 2 \rangle \| V_2[2]$

If  $x_1 \neq x_2$  then return  $x_1, x_2$

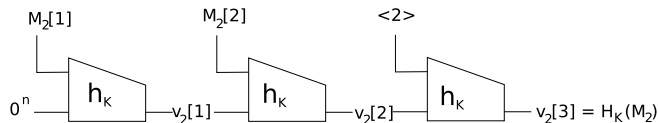
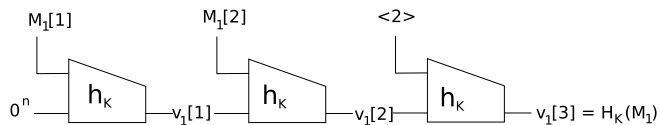
Else //  $V_1[2] = V_2[2]$

$x_1 \leftarrow M_1[2] \| V_1[1]$ ;  $x_2 \leftarrow M_2[2] \| V_2[1]$

If  $x_1 \neq x_2$  then return  $x_1, x_2$

Else //  $V_1[1] = V_2[1]$

## Case 2: $\|M_1\|_b = \|M_2\|_b$



$x_1 \leftarrow \langle 2 \rangle \| V_1[2]$ ;  $x_2 \leftarrow \langle 2 \rangle \| V_2[2]$

If  $x_1 \neq x_2$  then return  $x_1, x_2$

Else //  $V_1[2] = V_2[2]$

$x_1 \leftarrow M_1[2] \| V_1[1]$ ;  $x_2 \leftarrow M_2[2] \| V_2[1]$

If  $x_1 \neq x_2$  then return  $x_1, x_2$

Else //  $V_1[1] = V_2[1]$

$x_1 \leftarrow M_1[1] \| 0^n$ ;  $x_2 \leftarrow M_2[1] \| 0^n$

Return  $x_1, x_2$

## Exercise

Let  $h: \mathcal{K} \times \{0, 1\}^{2b} \rightarrow \{0, 1\}^b$  be a compression function. Define  $H: \mathcal{K} \times \{0, 1\}^{4b} \rightarrow \{0, 1\}^b$  as follows:

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

$M_1 \parallel M_2 \leftarrow M$

$V_1 \leftarrow h(K, M_1); V_2 \leftarrow h(K, M_2)$

$V \leftarrow h(K, V_1 \parallel V_2)$

return  $V$

Above, by  $M_1 \parallel M_2 \leftarrow M$ , we mean that  $M_1$  is the first  $2b$  bits of  $M$  and  $M_2$  is the rest, so that  $|M_1| = |M_2| = 2b$ .

Show that if  $h$  is collision-resistant then so is  $H$ . Do this by stating and proving an analogue of the Theorem above.

# 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

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

**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  $A$

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

Return  $x_1\|v_1, x_2\|v_2$



## 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

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

The compression function of SHA1 is underlain in this way by a block cipher  $E : \{0, 1\}^{512} \times \{0, 1\}^{160} \rightarrow \{0, 1\}^{160}$ .

The compression function of SHA256 is underlain in this way by a block cipher  $E : \{0, 1\}^{512} \times \{0, 1\}^{256} \rightarrow \{0, 1\}^{256}$ .

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

Can we do better than birthday if we do exploit the structure?

Ideally not, but functions have fallen short!

# Cryptanalytic attacks against hash functions

1993,1996 : collisions found in compression function md5 of MD5 in  $2^{16}$  time [dBBo,Do]. But these did not yield collisions for MD5.

2004, 2005 : collisions found in RIPEMD, SHA0 [WaFeLaYu].

2005, 2006: collisions can be found in MD5 in 1 minute [WaFeLaYu, LeWadW, Kl]

2015: collisions found in compression function sha1 of SHA1 [SKP]. But these do not yield collisions for SHA1.

2017 : collisions found for SHA1 in  $2^{63.1}$  time [SBKAM]  
<https://shattered.io/>.

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

# SHA1 collision

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

**SHattered**  
The first concrete collision attack against SHA-1  
<https://shattered.io>

**CWI**  
Marc Stevens  
Pierre Karpman

**Google**  
Elie Bursztein  
Ange Albertini  
Yarik Markov

**SHattered**  
The first concrete collision attack against SHA-1  
<https://shattered.io>

**CWI**  
Marc Stevens  
Pierre Karpman

**Google**  
Elie Bursztein  
Ange Albertini  
Yarik Markov

```
└─ sha1sum *.pdf
38762cf7f55934b34d179ae6a4c80cadccb7f0a 1.pdf
38762cf7f55934b34d179ae6a4c80cadccb7f0a 2.pdf
└─ /tmp/sha1
└─ sha256sum *.pdf
2bb787a73e37352f92383abe7e2902936d1059ad9f1ba6daaa9c1e58ee6970d0 1.pdf
d4488775d29bdef7993367d541064dbdda50d383f89f0aa13a6ff2e0894ba5ff 2.pdf
```



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

[Ars Technica](#) - Feb 23, 2017

The most well-known **collision** occurred sometime around 2010 against the MD5 hash algorithm, which is even weaker than **SHA1**. A piece of ...



## Watershed **SHA1** collision just broke the WebKit repository, others ...

[Ars Technica](#) - Feb 24, 2017

Thursday's watershed attack on the widely used **SHA1** hashing function has claimed its first casualty: the version control system used by the ...



## **SHA-1** collision can break SVN code repositories

[PCWorld](#) - Feb 27, 2017

A recently announced **SHA-1** collision attack has the potential to break code repositories that use the Subversion (SVN) revision control system.

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

These are based on the MD paradigm we discussed above.

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

**Submissions:** 64

**Round 1:** 51

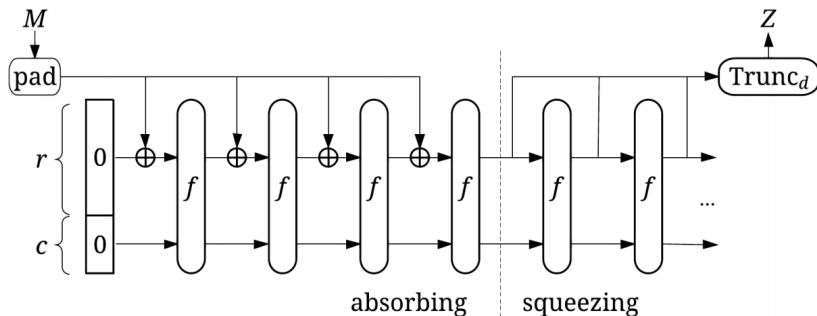
**Round 2:** 14: BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grostl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, Skein.

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

**SHA3:** 1: Keccak



# SHA3: 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 SHA1 and SHA2.

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