HASH FUNCTIONS

Hashing

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

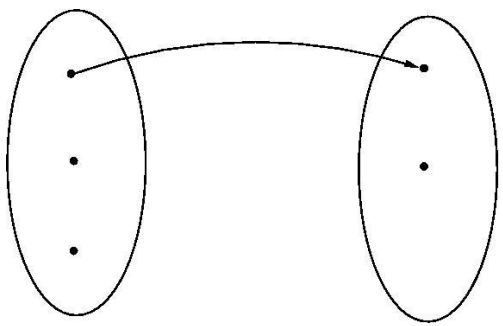
Their primary purpose is collision-resistant data compression, but they have many other purposes and properties as well.

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

Collision resistance (CR)

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

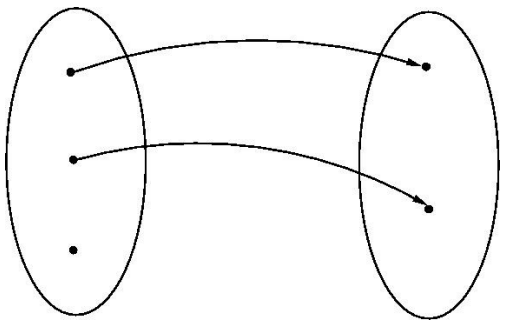
If $|D| > 2^n$ then the pigeonhole principle tells us that there must exist a collision for h.



Collision resistance (CR)

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

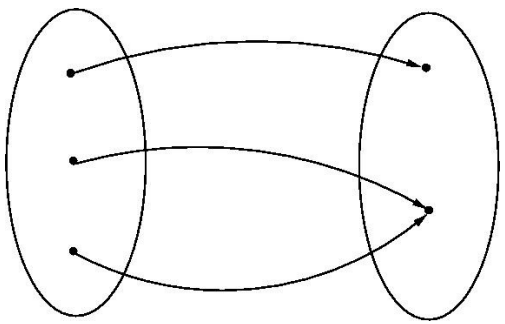
If $|D| > 2^n$ then the pigeonhole principle tells us that there must exist a collision for h.



Collision resistance (CR)

Definition: A collision for a function $h : D \to \{0,1\}^n$ 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 $H : \mathrm{Keys}(H) \times D \to R$ of functions, meaning for each $K \in \mathrm{Keys}(H)$ 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)
procedure Initialize	If $(x_1 = x_2)$ then return false
$K \xleftarrow{\$} \mathrm{Keys}(H)$	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

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

Collision-resistance

Game CR_H	**procedure Finalize**(x_1, x_2)
procedure Initialize	If $(x_1 = x_2)$ then return false
$K \xleftarrow{\$} \mathrm{Keys}(H)$	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: $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher.
Let H: $\{0,1\}^k \times \{0,1\}^{2n} \to \{0,1\}^n$ be defined by

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

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

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

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

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

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 $A(K)$

$x_1 \leftarrow 0^{128}1^{128}$; $x_2[2] \leftarrow 0^{128}$; $x_2[1] \leftarrow E_K^{-1}(E_K(x_1[1]) \oplus x_1[2] \oplus x_2[2])$

return x_1, x_2

Then $\mathbf{Adv}^{\mathrm{cr}}_H(A) = 1$ 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 $E\colon\ \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$ be a blockcipher. Let D be the set of all strings whose length is a positive multiple of l.

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

Alg $H(K, M)$

$M[1]M[2]\ldots M[n] \leftarrow M$
$C[0] \leftarrow 0^l$
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 $\mathbf{Adv}^{\mathrm{cr}}_H(A) = 1$.

Keyless hash functions

We say that H: $\mathsf{Keys}(H) \times D \to R$ is keyless if $\mathsf{Keys}(H) = \{\varepsilon\}$ 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, SHA3, ... are keyless.

SHA1

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

$V \leftarrow$ SHF1(5A827999 $\|$ 6ED9EBA1 $\|$ 8F1BBCDC $\|$ CA62C1D6 , M)

return V

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

$y \leftarrow$ shapad(M)

Parse y as $M_1 \| M_2 \| \cdots \| M_n$ where $|M_i| = 512$ $(1 \le i \le n)$

$V \leftarrow$ 67452301 $\|$ EFCDAB89 $\|$ 98BADCFE $\|$ 10325476 $\|$ C3D2E1F0

for $i = 1, \ldots, n$ do $V \leftarrow$ shf1($K, M_i \| V$)

return V

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

$d \leftarrow (447 - |M|) \bmod 512$

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

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

return y

shf1

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

Parse B as $W_0 \,\|\, W_1 \,\|\, \cdots \,\|\, W_{15}$ where $|W_i| = 32$ $(0 \le i \le 15)$

Parse V as $V_0 \,\|\, V_1 \,\|\, \cdots \,\|\, V_4$ where $|V_i| = 32$ $(0 \le i \le 4)$

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

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 $(0 \le t \le 19)$ then $f \leftarrow (B \wedge C) \vee ((\neg B) \wedge D)$

  if $(20 \le t \le 39$ OR $60 \le t \le 79)$ then $f \leftarrow B \oplus C \oplus D$

  if $(40 \le t \le 59)$ then $f \leftarrow (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$

  $temp \leftarrow \text{ROTL}^5(A) + f + E + W_t + L_t$

  $E \leftarrow D$; $D \leftarrow C$; $C \leftarrow \text{ROTL}^{30}(B)$; $B \leftarrow A$; $A \leftarrow temp$

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

$V \leftarrow V_0 \,\|\, V_1 \,\|\, V_2 \,\|\, V_3 \,\|\, V_4$; return V

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 (SSL)

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

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

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

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

But we will revisit this when we consider dictionary attacks!

Compare-by-hash

- 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 $F_A = F_B$

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

Problem: Transmission could take forever, particularly if the link is slow (DSL).

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 $h_A = H(F_B)$.

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

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

Virus protection

An executable may be available at lots of sites $S_1, S_2, \ldots, S_N$. 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 $H : \{0,1\}^k \times D \to \{0,1\}^n$ 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 $i = 1, \ldots, q$ do $y_i \xleftarrow{\$} \{0,1\}^n$
if $\exists i, j$ $(i \neq j$ and $y_i = y_j)$ then $\mathsf{COLL} \leftarrow \mathsf{true}$

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

Birthday attack

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

adversary $A(K)$
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 FAIL

Analysis of birthday attack

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

adversary $A(K)$
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 FAIL

What is the probability that this attack finds a collision?

adversary $A(K)$
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 $y_i = y_j)$ 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	Adversary A

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

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

Are the two collision probabilities the same?

Analysis of birthday attack

Birthday	Adversary A

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

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

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

Analysis of birthday attack

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

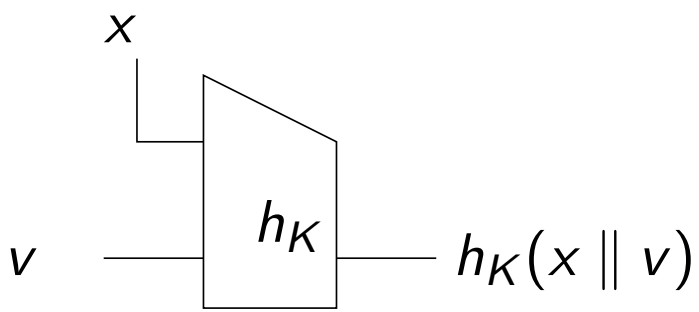
T_B is the number of trials to find collisions via a birthday attack.

Compression functions

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

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



The MD transform

Design principle: To build a CR hash function

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

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

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

MD setup

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

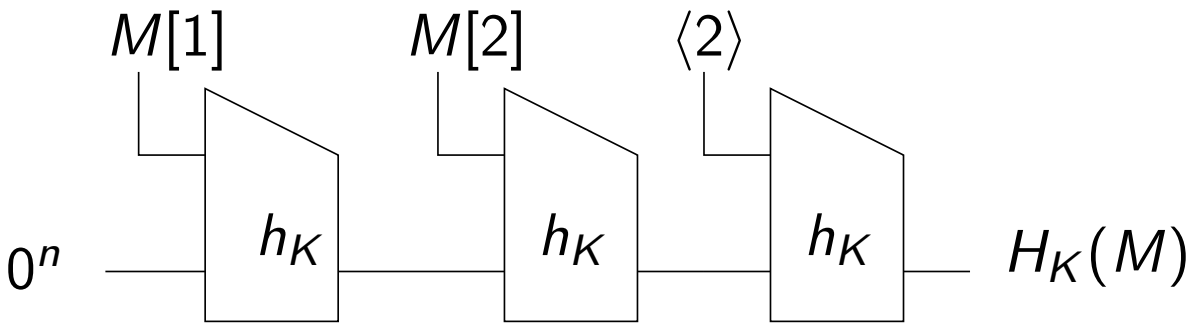
- $\|M\|_b$ be the number of b-bit blocks in M, and write $M = M[1] \ldots M[\ell]$ where $\ell = \|M\|_b$.
- $\langle i \rangle$ denote the b-bit binary representation of $i \in \{0, \ldots, 2^b - 1\}$.
- D be the set of all strings of at most $2^b - 1$ blocks, so that $\|M\|_b \in \{0, \ldots, 2^b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

MD transform

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

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

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.

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.

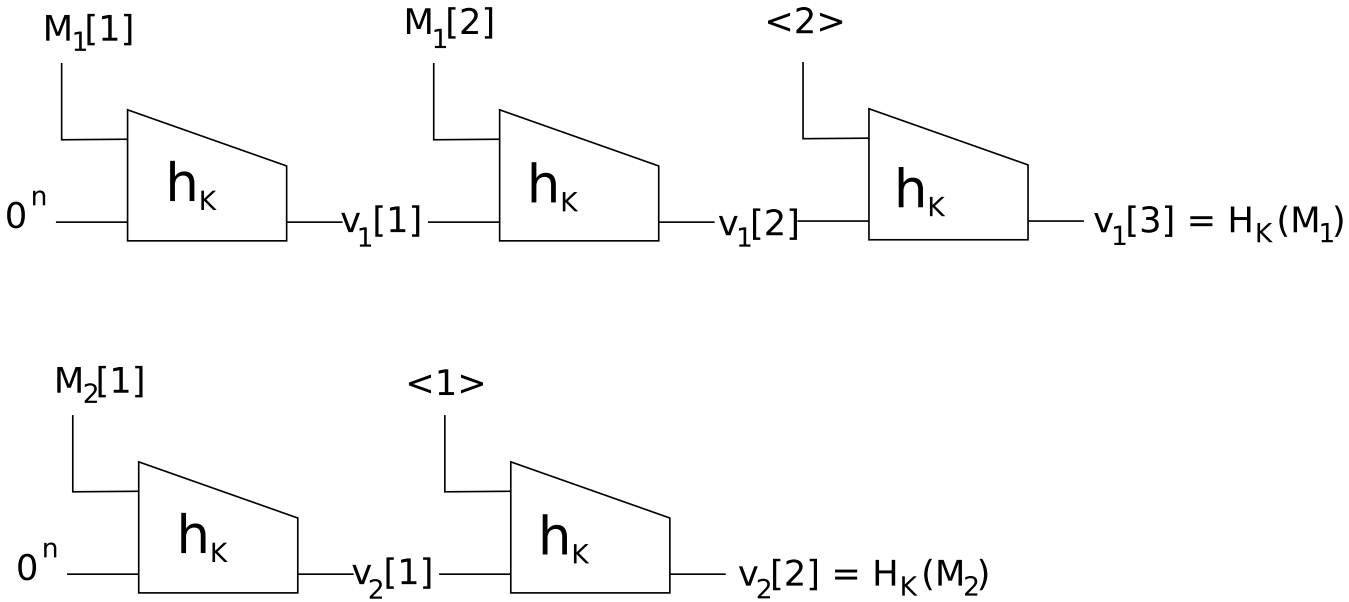
Implication:

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

How A_h works

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.

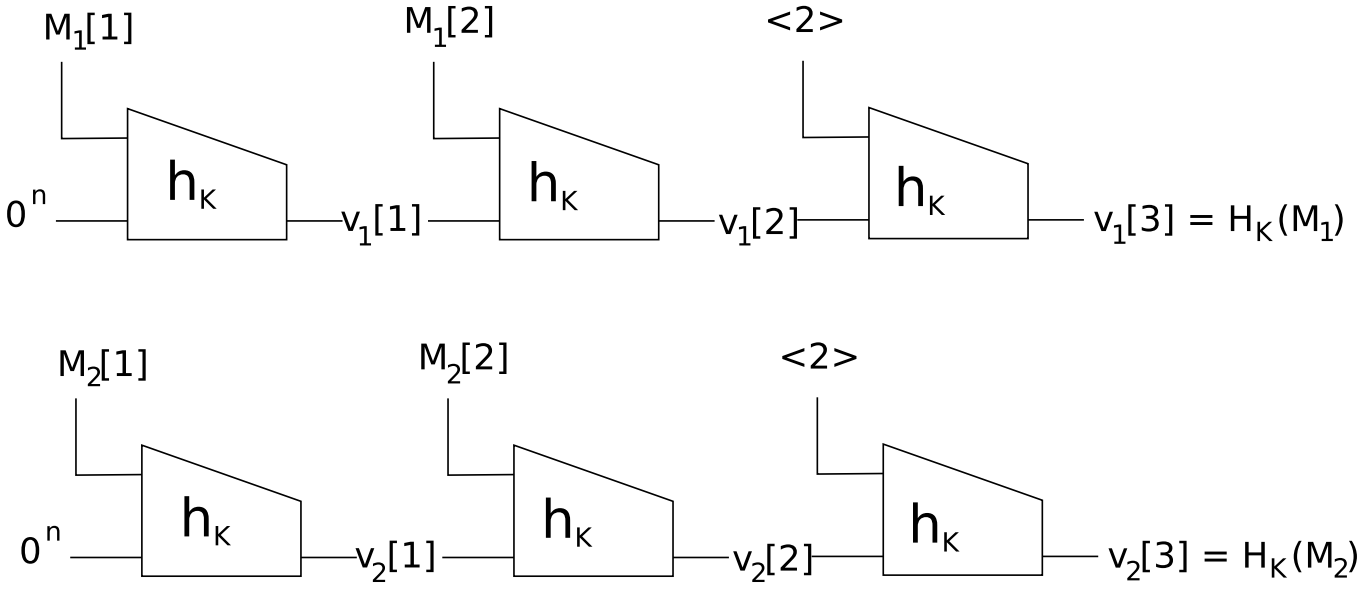
Case 1: $\|M_1\|_b \neq \|M_2\|_b$



Let $x_1 = \langle 2 \rangle || V_1[2]$ and $x_2 = \langle 1 \rangle || V_2[1]$. Then

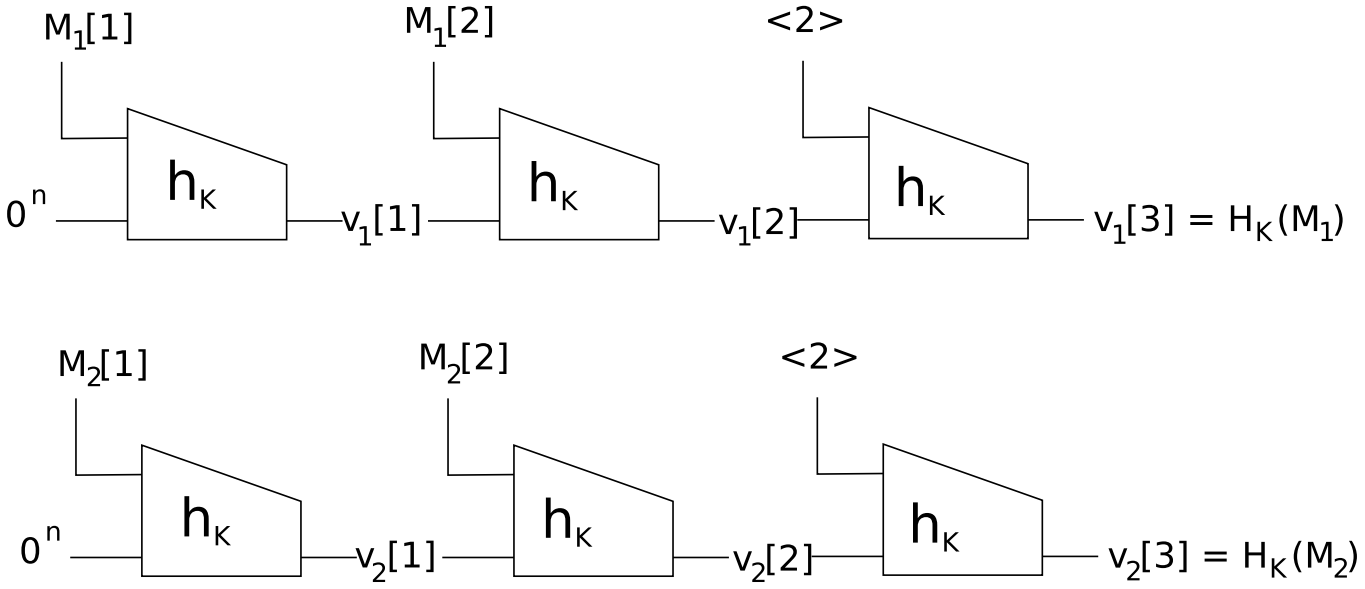
- $h_K(x_1) = h_K(x_2)$ because $H_K(M_1) = H_K(M_2)$.
- But $x_1 \neq x_2$ because $\langle 1 \rangle \neq \langle 2 \rangle$.

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



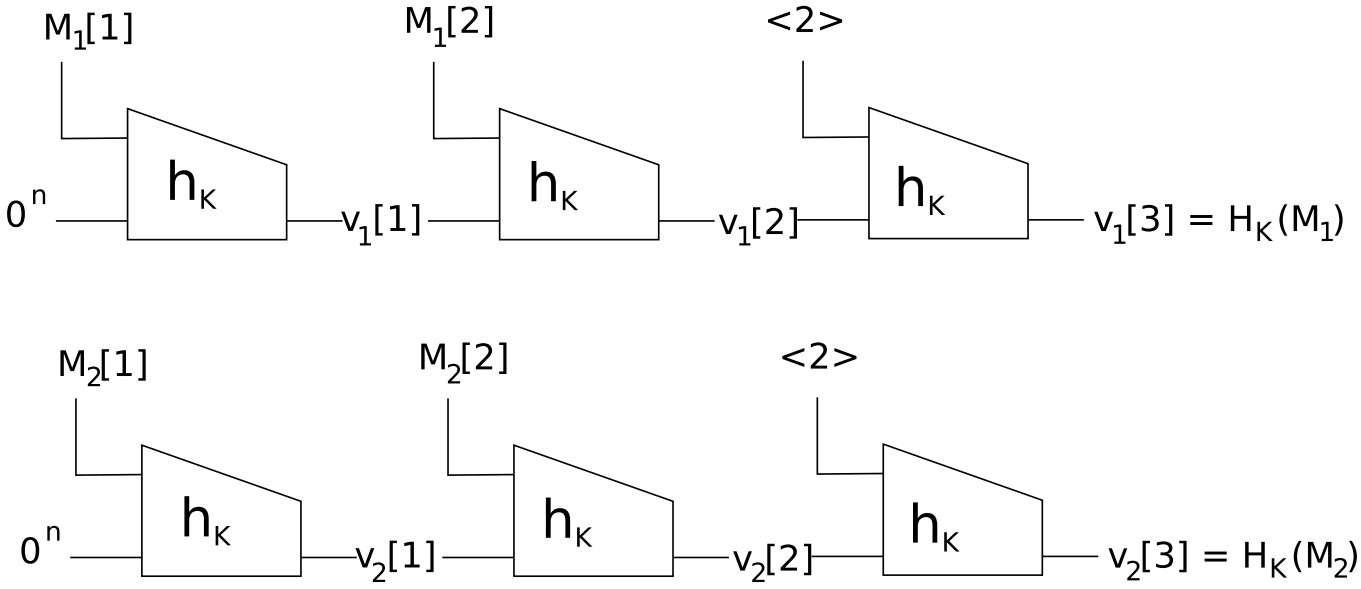
$x_1 \leftarrow \langle 2 \rangle || V_1[2]$; $x_2 \leftarrow \langle 2 \rangle || V_2[2]$
If $x_1 \neq x_2$ then return x_1, x_2

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



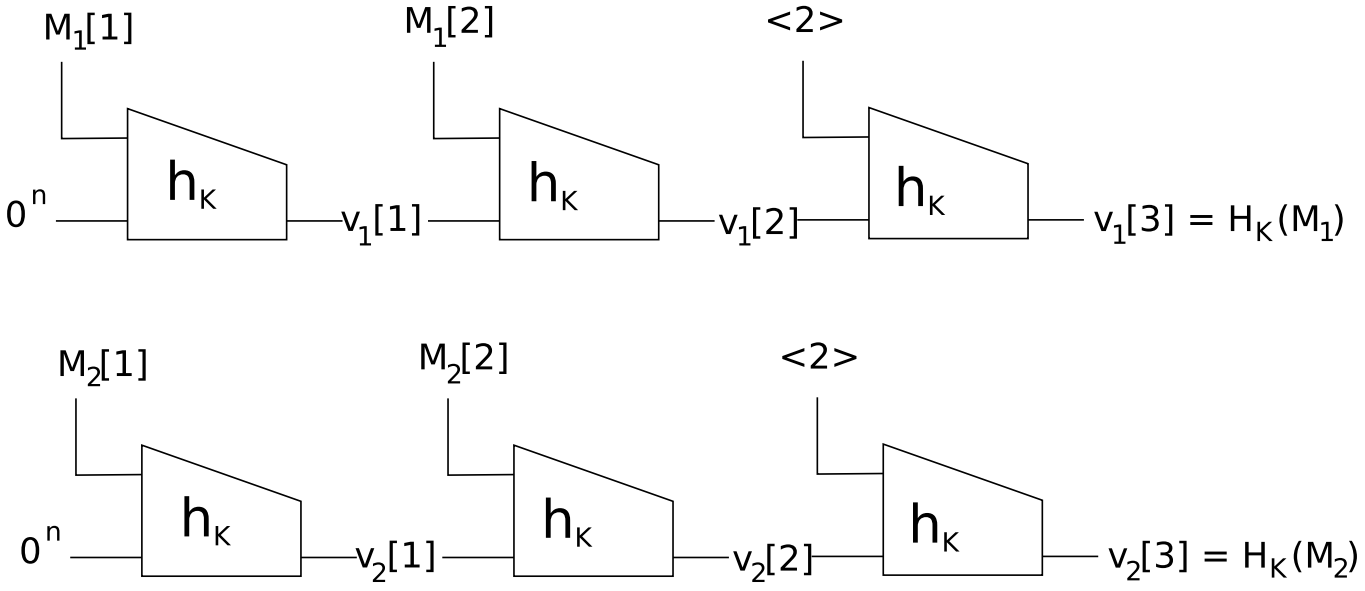
$x_1 \leftarrow \langle 2 \rangle || V_1[2]$; $x_2 \leftarrow \langle 2 \rangle || V_2[2]$
If $x_1 \neq x_2$ then return x_1, x_2
Else // $V_1[2] = V_2[2]$

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



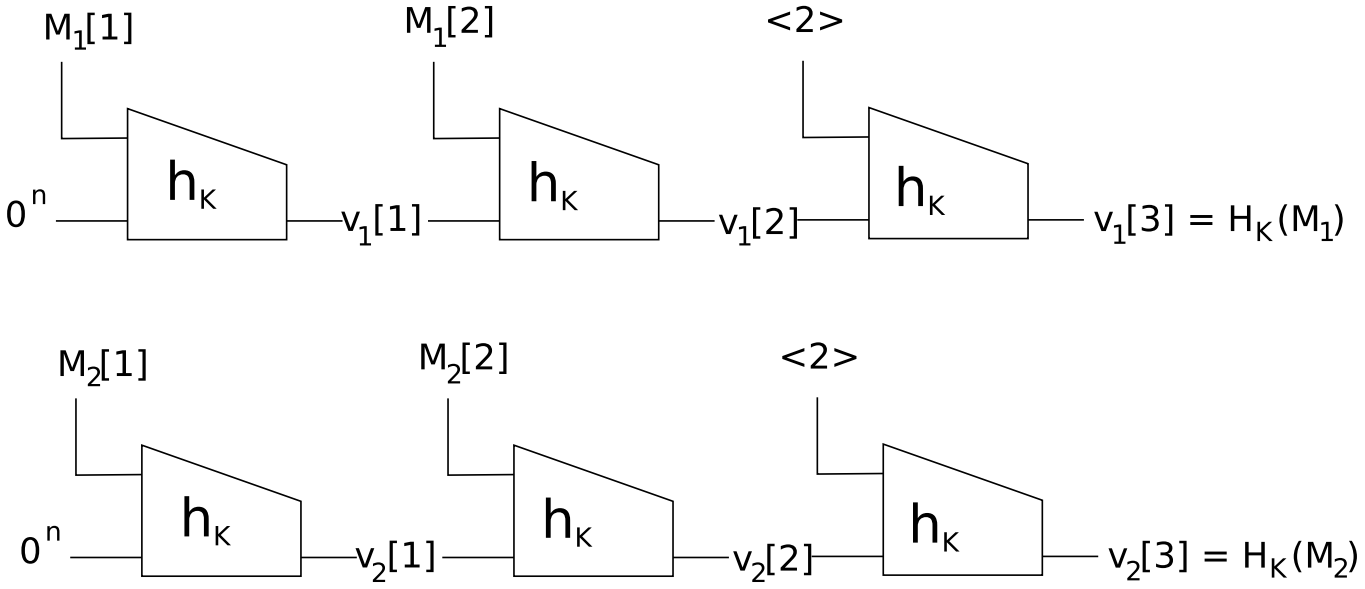
$x_1 \leftarrow \langle 2 \rangle || V_1[2]$; $x_2 \leftarrow \langle 2 \rangle || V_2[2]$
If $x_1 \neq x_2$ then return x_1, x_2
Else // $V_1[2] = V_2[2]$
    $x_1 \leftarrow M_1[2] || V_1[1]$; $x_2 \leftarrow M_2[2] || V_2[1]$
    If $x_1 \neq x_2$ then return x_1, x_2

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: $\mathcal{K} \times \{0,1\}^{2b} \to \{0,1\}^b$ be a compression function. Define H: $\mathcal{K} \times \{0,1\}^{4b} \to \{0,1\}^b$ 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 $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let us design keyless compression function

$$h : \{0,1\}^{b+n} \to \{0,1\}^n$$

by

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

Is H collision resistant?

How are compression functions designed?

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

$$h : \{0,1\}^{b+n} \to \{0,1\}^n$$

by

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

Is H collision resistant?

NO!

adversary A

Pick some x_1, x_2, v_1 with $x_1 \neq x_2$

$y \leftarrow E_{x_1}(v_1)$; $v_2 \leftarrow E^{-1}_{x_2}(y)$

return $x_1 \parallel v_1, x_2 \parallel v_2$

Then

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

How are compression functions designed?

Let $E : \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Keyless compression function

$$h : \{0,1\}^{b+n} \to \{0,1\}^n$$

may be designed as

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

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

Cryptanalytic attacks

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

When	Against	Time	Who
1993,1996	md5	2^{16}	[dBBo,Do]
2005	RIPEMD	2^{18}	
2004	SHA0	2^{51}	[JoCaLeJa]
2005	SHA0	2^{40}	[WaFeLaYu]
2005	SHA1	2^{69}	[WaYiYu]
2012	SHA1	$2^{60} - 2^{65}$	[St]
2005,2006	MD5	1 minute	[WaFeLaYu,LeWadW,KI]

md5 is the compression function of MD5
SHA0 is an earlier, weaker version of SHA1

Security of MD5

MD5 is used in 720 different places in Microsoft Windows OS.

What can current attacks do against MD5?

- Find 2 random-looking messages that only differ in 3 bits (boring)
- Find two PDF documents whose hashes collide (more exciting)
- Find two Win32 executables whose hashes collide (very exciting)
- Break deployed cryptographic protocols (very exciting)

Finding collisions

How do attacks work in reality against MD5? Examples:

- Find 2 random-looking messages that only differ in 3 bits
  Cochran's code for MD5:
  `http://www.cs.colorado.edu/~jrblack/md5toolkit.tar.gz`
  Work's in a few minutes on laptop...try it!

- Find 2 Win32 executables whose hashes collide
  Swiss group:
  `http://www.win.tue.nl/hashclash/SoftIntCodeSign/`
  Takes 2 days on a Playstation 3

Status of SHA1

No collisions for SHA1 have yet been publicly announced.

But collisions were found for the compression function sha1 of SHA1 in 2015.

We expect collisions for SHA1 to be found and announced soon. Google, Microsoft and Mozilla browsers will stop accepting SHA1-based certificates in 2017.

Exercise

You are using SHA1 in your product. Your boss is worried and asks you to explain what is going on. The theorem we proved shows that if sha1 is collision resistant then so is SHA1. How does that mesh with the above, namely that collisions have been found for sha1 but not for SHA1, and what are the implications of the attack for the security of SHA1? Should SHA1 be moved out of your product? What would you recommend to your boss in that regard?

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

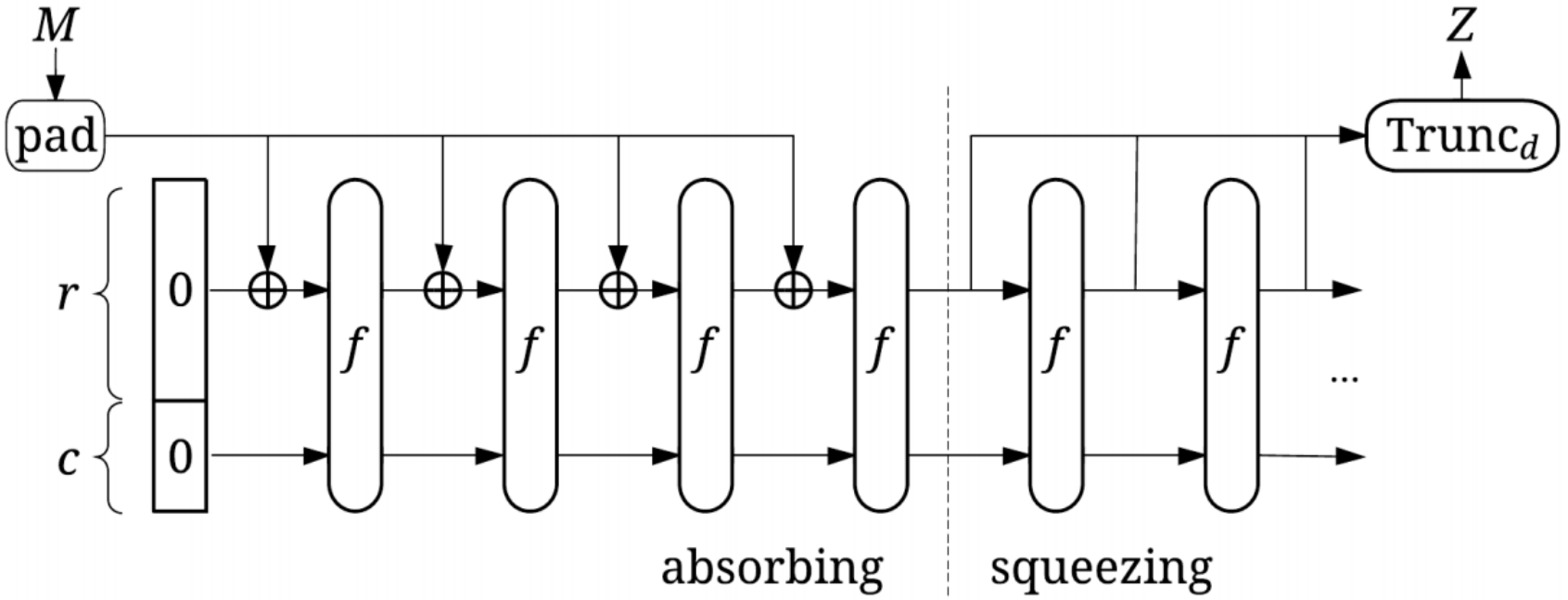
Round 1: 51

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: The Sponge construction



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

SHA3 does not use the MD paradigm used by SHA1 and SHA2.

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