# Hash functions

CS-466: Applied Cryptography
Adam O'Neill

Adapted from http://cseweb.ucsd.edu/~mihir/cse107/

# Setting the Stage

- Today we will study a second lower-level primitive, hash functions.

# Setting the Stage

- Today we will study a second lower-level primitive, hash functions.
- Hash functions like MD5, SHA1, SHA256 are used pervasively.

# Setting the Stage

- Today we will study a second lower-level primitive, hash functions.
- Hash functions like MD5, SHA1, SHA256 are used pervasively.
- Primary purpose is data compression, but they have many other uses and are often treated like a "magic wand" in protocol design.

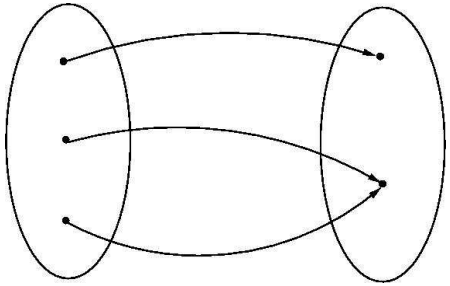
# Collision Resistance

**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 $x_1 \neq x_2$.

# Collision Resistance

**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 $x_1 \neq x_2$.

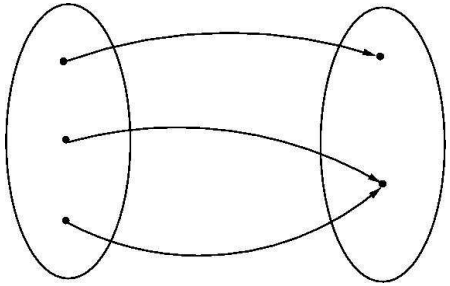
If $|D| > 2^n$ then the pigeonhole principle tells us that there must exist a collision for $h$.



# Collision Resistance

**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.

# The Game

Suppose $\text{Keys}(H) = \{\varepsilon\}$ then can $H$ be CR-secure?

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 $\text{CR}_H$ | **procedure Finalize**$(x_1, x_2)$ |
|---|---|
| **procedure Initialize** | 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 $(H_K(x_1) = H_K(x_2))$ |

adversary gets the key. check collision valid

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

We say that $H$ is CR-secure (or just CR) if for any "practical" $A$, $\mathbf{Adv}^{\text{cr}}_H(A)$ is "small".

# 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

(e.g. Let $E$ = AES, $n = 128$)

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

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

Claim. $H$ is not CR.

Want $x_1[1]\,x_1[2]$, $x_2[1]\,x_2[2]$ which are unequal. st.

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

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

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

# Keyless Hash Functions

We say that $H$: $\text{Keys}(H) \times D \to R$ 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, SHA3, ... are keyless.

Although a formal definition of CR doesn't make sense (or is not achievable) for keyless hash functions, we can still use them in reductions.

# SHA1

Secure Hash Algorithm.

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

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

return $V$

160 bits.

---

**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 $\ell$ be the 64-bit binary representation of $|M|$

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

return $y$

# Underlying Compression Function

**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

- Hashing before digitally signing.

# Applications

- Hashing before digitally signing.
- Primitive in cryptographic protocols.

# Applications

- Hashing before digitally signing.
- Primitive in cryptographic protocols.
- Tool for security applications.

# Applications

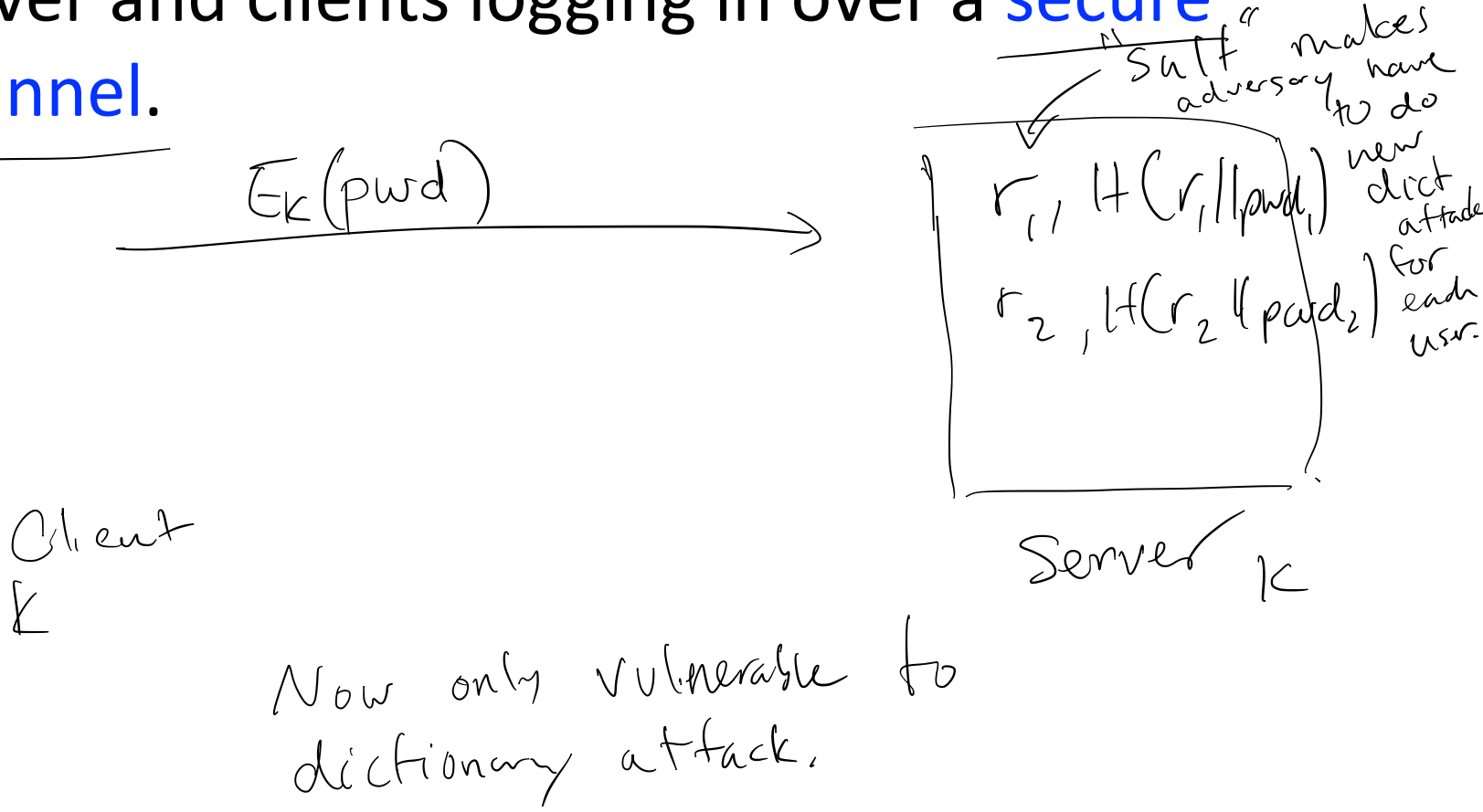
- Hashing before digitally signing.
- Primitive in cryptographic protocols.
- Tool for security applications.
- Tool for non-security applications.

# Applications

- Hashing before digitally signing.
- Primitive in cryptographic protocols.
- Tool for security applications.
- Tool for non-security applications.
- Let's see some examples…

# Password Verification

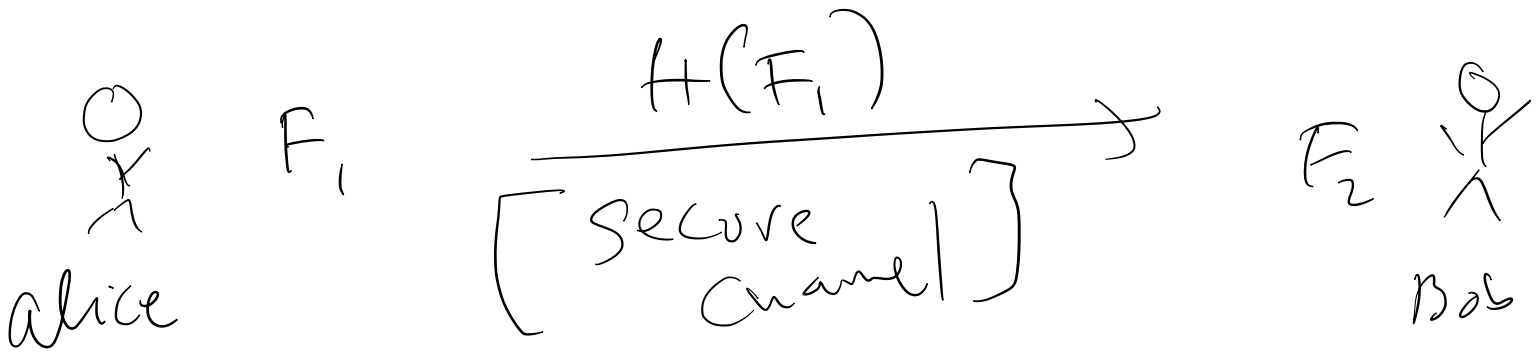
- Consider a password file stored on a remote server and clients logging in over a secure channel.




Now only vulnerable to dictionary attack.

# Compare-by-Hash

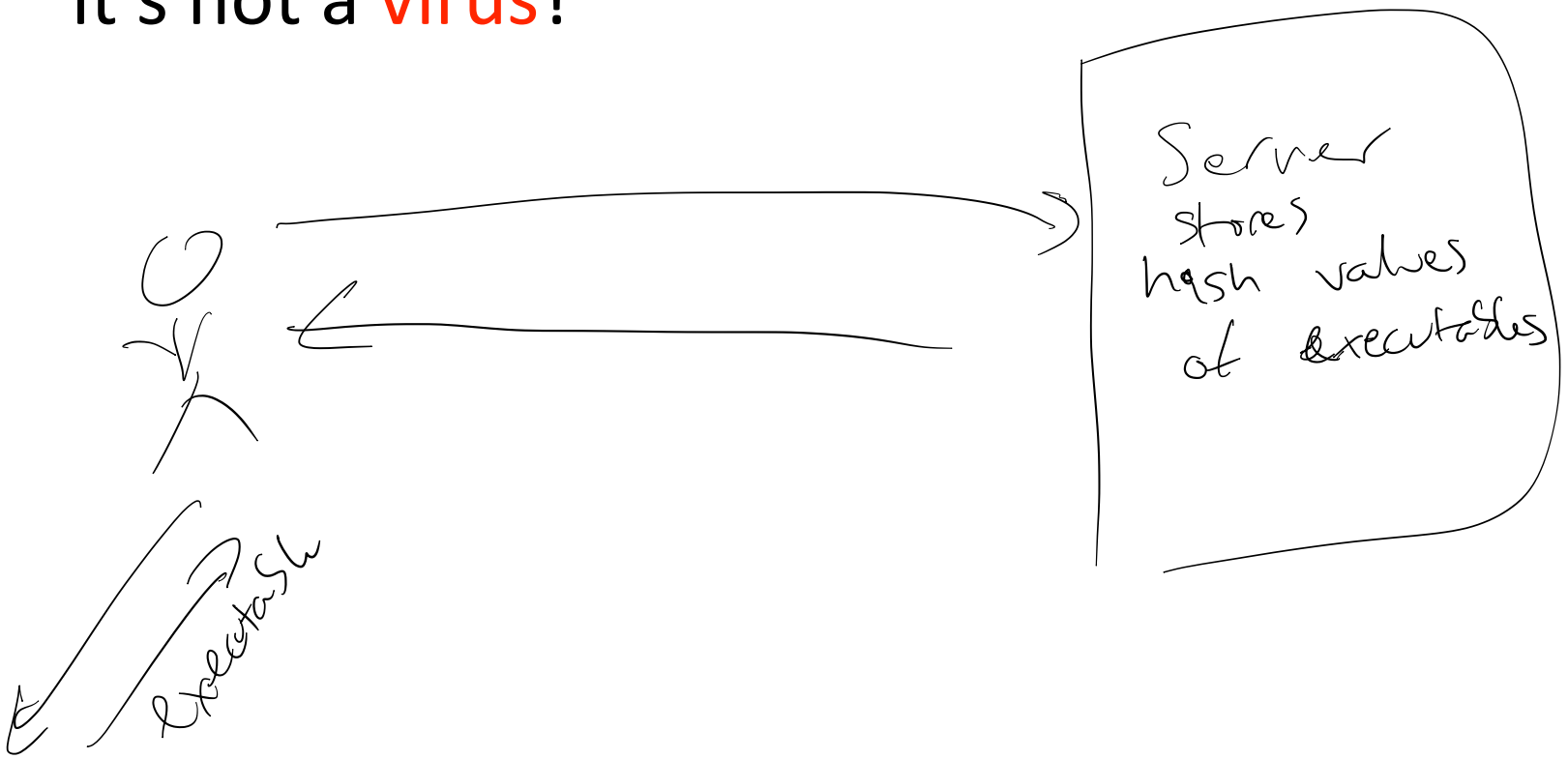
- Suppose two parties each have a large file and want to know if they have the same file.




1. Useful even in non-security contexts.
2. Seems to provide some measure of security (but dictionary attack still applies)

# Virus Protection

- Suppose you download an executable from somewhere on the Internet. How do you know it's not a virus?



# Birthday Attack

Let: $H: D \to \{0,1\}^n$ be a hash function.

Consider for some integer parameter $q$:

Adversary $A$

For $i = 1$ to $q$ do:

$x_i \xleftarrow{\$} D$

$y_i \leftarrow H(x_i)$

If $\exists i_1, i_2 \in [q]$ s.t.

$(H(x_{i_1}) = H(x_{i_2}) \wedge x_{i_1} \neq x_{i_2})$

then return $(x_{i_1}, x_{i_2})$

Else return $\perp$.

# Analysis

Assume that $H$ is regular meaning

$$\forall y \in \{0,1\}^n \quad |H^{-1}(y)| = \frac{|D|}{2^n}$$

Then $$Pr[H(x_i) = y] = Pr[x_i \in H^{-1}(y)]$$

$$= \frac{1}{2^n}$$

then $$Adv^{CR}_H(A) = C(2^n, q) \geq \frac{3q(q-1)}{2^n}$$

$\Rightarrow$ need n-bit output for n/2-bit secury

e.g. for 80-bit sec. we need 160 bit output

e.g. SHA1 has 160 bit. output
SHA256 has 256 bit output

| 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 | $2^{256}$ |

output length

time to find collision by bday attack

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

non-generic attacks!
(bday attack is generic)

| 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] |

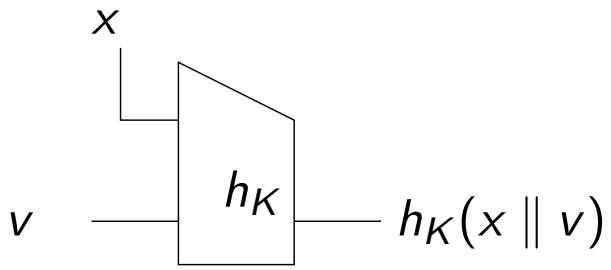
md5 is the compression function of MD5
SHA0 is an earlier, weaker version of SHA1

# 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

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



Merkle-Damgaard

# 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.

b is called "capacity"

# MD Setup

e.g. 512 for SHA1

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.

# 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

- The nice property of the MD transform is that it preserves collision-resistance (CR).

# MD preserves CR

- The nice property of the MD transform is that it preserves collision-resistance (CR).
- If we start with a CR fixed input-length compression function we end up with a CR hash function taking unbounded-length inputs.

at least for practical purposes.

# MD preserves CR

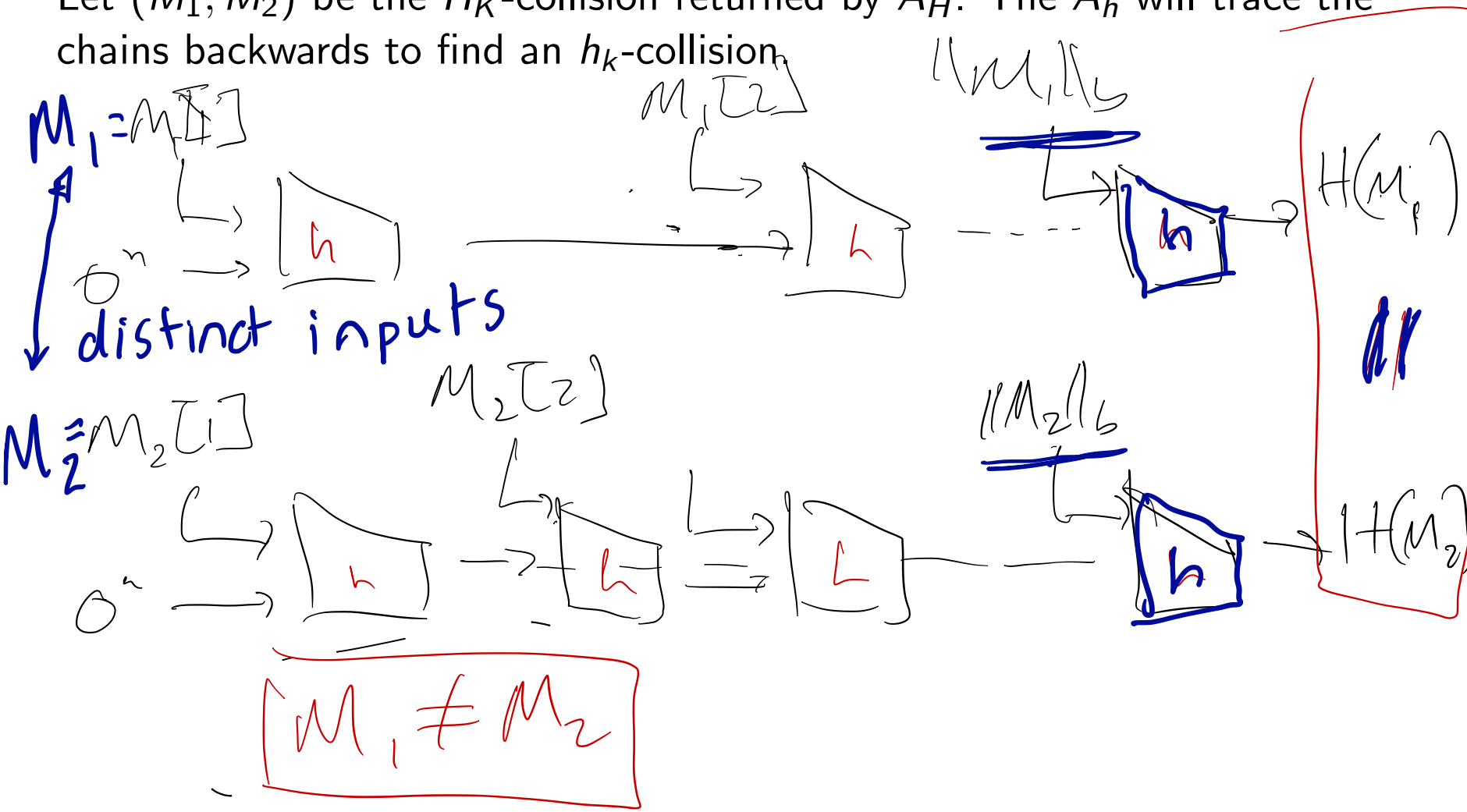
- The nice property of the MD transform is that it preserves collision-resistance (CR).
- If we start with a CR fixed input-length compression function we end up with a CR hash function taking unbounded-length inputs.
- There is no need to cryptanalyze the latter. The only way to break it is to break the compression function.

$M_1$ has $a_1$ blocks $M_2$ has $a_2$ blocks

# Proof Idea

$a_1 + a_2 + 2$ computations of $h$.

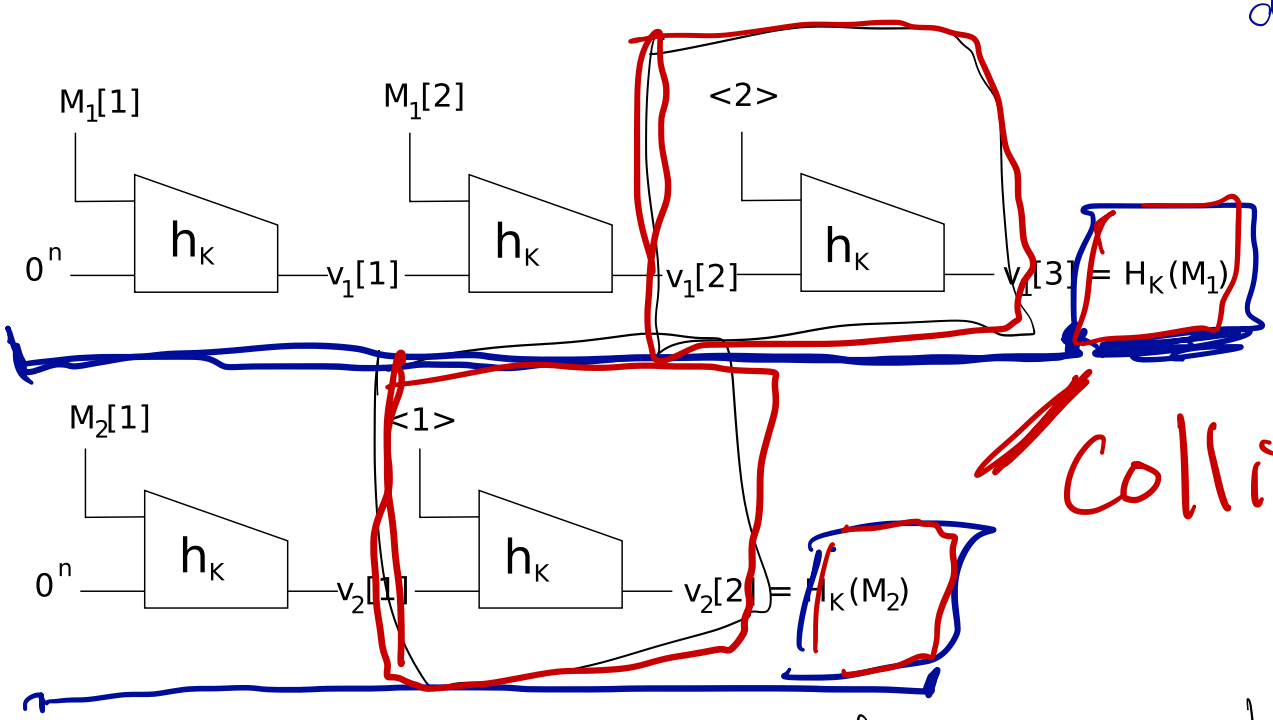
Let $(M_1, M_2)$ be the $H_K$-collision returned by $A_H$. The $A_h$ will trace the chains backwards to find an $h_k$-collision.



$|M_1|_b \neq |M_2|_b$

# Case 1 —

the length in b-bit blocks of $M_1$ and $M_2$ is different.



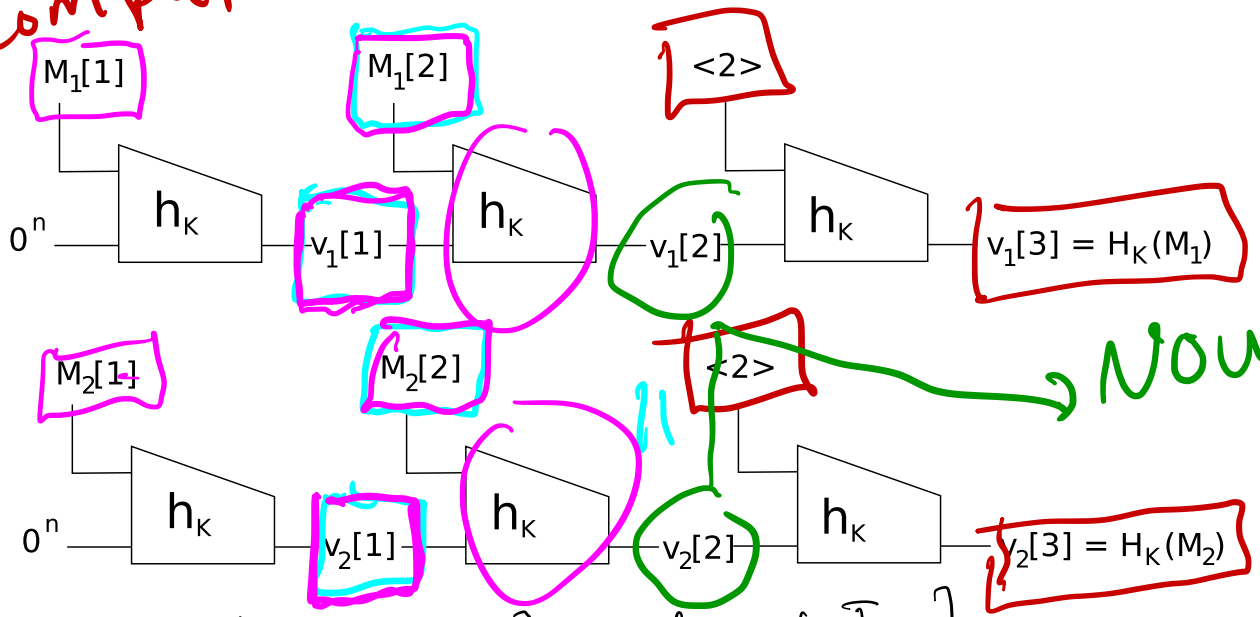

collide!

We have a collision in h in the last invocations in each computation.

# Case 2

$\|M_1\|_b = \|M_2\|_b$

lines up block-for-block h computations




NOW LOOK HERE

Look at $V_1[2]$ and $V_2[2]$.

- If they are different, we have a collision in h
- If they are the same, go back one invocation of h and repeat.

Eventually we find a collision or $M_1 = M_2$, which is a contradiction

# Compression from Blockcipher?

key length = $b$ block length = $n$

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

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

SHA 1
$b = 512$
$n = 160$

by

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

Is $H$ collision resistant?

No! We use the fact that $E_K$ is efficiently invertible given $K$ (i.e. $E_K^{-1}$ is efficiently computable)

Adversary $A$:
$y \leftarrow E(K, x)$ where $K, X$ are arbitrary.
$x' \leftarrow E^{-1}(K', y)$ where $K' \neq K$
ref $(K||x, K'||x')$

how SHA1's compression function works

# A Better Way

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

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

compression function

may be designed as

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

Davies–Meyer compression function

prevents previous attack

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

hard to solve for $v$ or $v'$

Davies-Meyer

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

$E_x(v) \oplus v = E_{x'}(v') \oplus v'$

SHA1 output length 160

Recall:

# Non-Generic Attacks

| 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

SHAttered. 2017

NIST competition

# SHA3 Competition

**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.

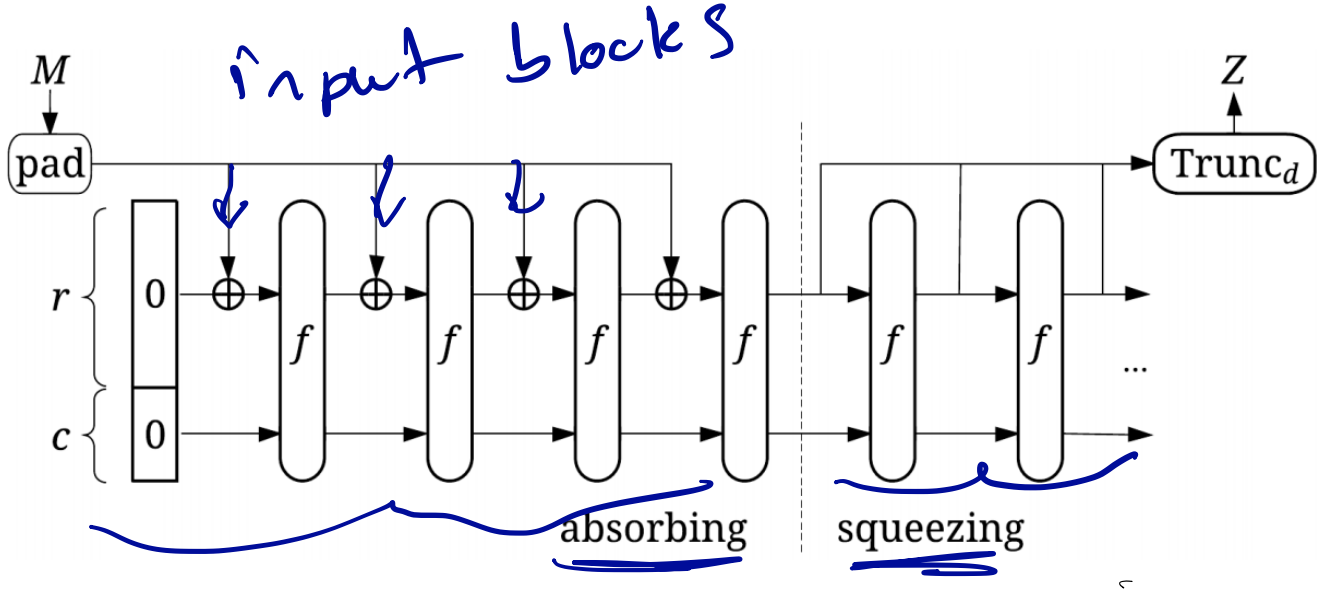
**Finalists:** 5: BLAKE, Grostl, JH, Keccak, Skein.

**SHA3:** 1: Keccak

doesn't use MD.

uses "sponge functions"

# Winner: The Sponge Construction

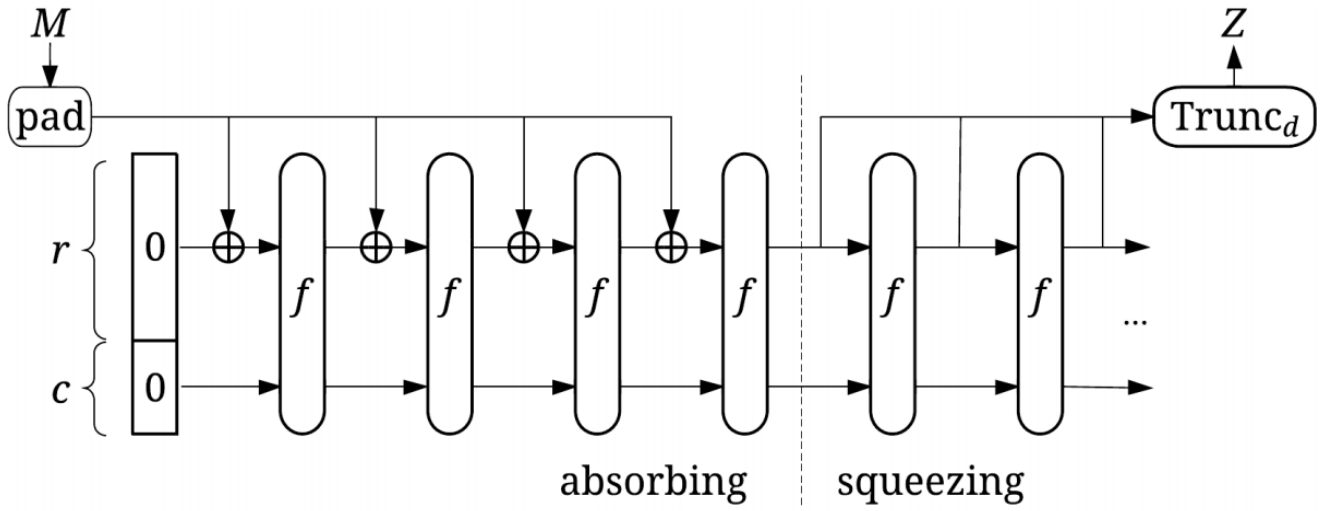



$f$: $\{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a (public, invertible!) permutation.
$d$ is the number of output bits, and $c = 2d$.

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.

# Winner: 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.