HASH FUNCTIONS
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 ...
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 $h$. 

![Diagram showing a collision]
**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 \( h \).
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 $H : \text{Keys}(H) \times D \to R$ of functions, meaning for each $K \in \text{Keys}(H)$ we have a map $H_K : D \to R$ defined by $H_K(x) = H(K, x)$.

Let

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

<table>
<thead>
<tr>
<th>Game $\text{CR}_H$</th>
<th>procedure $\text{Finalize}(x_1, x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>procedure Initialize</strong></td>
<td>If $(x_1 = x_2)$ then return false</td>
</tr>
<tr>
<td>$K \leftarrow $ \text{Keys}(H)$</td>
<td>If $(x_1 \notin D$ or $x_2 \notin D)$ then return false</td>
</tr>
<tr>
<td>Return $K$</td>
<td>Return $(H_K(x_1) = H_K(x_2))$</td>
</tr>
</tbody>
</table>

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

$$\text{Alg } H(K, x[1]x[2])$$
$$y \leftarrow E_K(E_K(x[1]) \oplus x[2]); \text{ Return } y$$

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

$$
\textbf{Alg } H(K, x[1]x[2])
$$

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

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

**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** \( H(K, x[1]x[2]) \)

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

**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** \( A(K) \)

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

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:

\[
\textbf{Alg } H(K, M)
\]

\[
\]

\[
C[0] \leftarrow 0^l
\]

For $i = 1, \ldots, n$ do

\[
B[i] \leftarrow E(K, C[i - 1] \oplus M[i]); \quad C[i] \leftarrow E(K, B[i] \oplus M[i])
\]

Return $C[n]

Show that $H$ is not CR by giving an efficient adversary $A$ such that $\text{Adv}_{H}^{\text{cr}}(A) = 1$. 

Mihir Bellare

UCSD
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, SHA3, ... are keyless.
SHA1

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

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

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

\[
y \leftarrow \text{shapad}(M)
\]
Parse y as \( M_1 \parallel M_2 \parallel \cdots \parallel M_n \) where \( |M_i| = 512 \) (1 ≤ i ≤ n)
\[
V \leftarrow 67452301 \parallel \text{EFCDAB89} \parallel \text{98BADCFE} \parallel \text{10325476} \parallel \text{C3D2E1F0}
\]
for \( i = 1, \ldots, n \) do \( V \leftarrow \text{shf1}(K, M_i \parallel V) \)
return V

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

\[
d \leftarrow (447 - |M|) \mod 512
\]
Let \( \ell \) be the 64-bit binary representation of |M|
\[
y \leftarrow M \parallel 1 \parallel 0^d \parallel \ell  \quad // \quad |y| \text{ is a multiple of 512}
\]
return y
**Alg** \( \text{shf1}(K, B \parallel V) \)  
// \(|K| = 128, |B| = 512\) and \(|V| = 160\)

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

Parse \( V \) as \( V_0 \parallel V_1 \parallel \cdots \parallel V_4 \) 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 \( f \leftarrow (B \land C) \lor (\lnot B) \land D\)

if \((20 \leq t \leq 39\) \(\text{OR} 60 \leq t \leq 79)\) then \( f \leftarrow B \oplus C \oplus D\)

if \((40 \leq t \leq 59)\) then \( f \leftarrow (B \land C) \lor (B \land D) \lor (C \land D)\)

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

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

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

\( V \leftarrow V_0 \parallel V_1 \parallel V_2 \parallel V_3 \parallel V_4; \text{return } V \)
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)

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

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

\[ A^{PW} \quad PW \quad B^{PW} \]

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 $F_A = F_B$

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

**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 $h_A = H(F_B)$.

\[ A^{F_A} \xrightarrow{h_A} B^{F_B} \]

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 $S_1, S_2, \ldots, S_N$. 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.
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 $i = 1, \ldots, q$ do $y_i \leftarrow \{0, 1\}^n$
if $\exists i, j$ ($i \neq j$ and $y_i = y_j$) then $\text{COLL} \leftarrow \text{true}$

$$\Pr[\text{COLL}] = C(2^n, q) \approx \frac{q^2}{2^{n+1}}$$
Let $H : \{0, 1\}^k \times D \rightarrow \{0, 1\}^n$.

**adversary $A(K)$**

for $i = 1, \ldots, q$ 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 $\text{FAIL}$
Analysis of birthday attack

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

**adversary** $A(K)$

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

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

We have dropped things that don’t much affect the advantage and focused on success probability. So we want to know what is $\Pr[\text{COLL}]$.

Let $H_K^{-1}(y) = \{x \in D : H_K(x) = y\}$ and assume the size of this set is $|H_K^{-1}(y)| = |D|/2^n$ for all $y$. Then we are throwing $q$ balls into $N = 2^n$ bins, and $\Pr[\text{COLL}] = \binom{N}{q} \approx \frac{q^2}{N}$. 
### Birthday attack times

<table>
<thead>
<tr>
<th>Function</th>
<th>$n$</th>
<th>$T_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD4</td>
<td>128</td>
<td>$2^{64}$</td>
</tr>
<tr>
<td>MD5</td>
<td>128</td>
<td>$2^{64}$</td>
</tr>
<tr>
<td>SHA1</td>
<td>160</td>
<td>$2^{80}$</td>
</tr>
<tr>
<td>SHA2-256</td>
<td>256</td>
<td>$2^{128}$</td>
</tr>
<tr>
<td>SHA2-512</td>
<td>512</td>
<td>$2^{256}$</td>
</tr>
<tr>
<td>SHA3-256</td>
<td>256</td>
<td>$2^{128}$</td>
</tr>
<tr>
<td>SHA3-512</td>
<td>512</td>
<td>$2^{256}$</td>
</tr>
</tbody>
</table>

$T_B$ is the number of trials to find collisions via a birthday attack.
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 $M = M[1] \ldots M[\ell]$ 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 $h : \{0, 1\}^k \times \{0, 1\}^{b+n} \rightarrow \{0, 1\}^n$.

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 $i = 1, \ldots, m + 1$ 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.
**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

$$\text{Adv}_{H}^{\text{cr}}(A_H) \leq \text{Adv}_{h}^{\text{cr}}(A_h)$$

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:**

$h \text{ CR} \quad \Rightarrow \quad \text{Adv}_{h}^{\text{cr}}(A_h) \text{ small}$

$\Rightarrow \quad \text{Adv}_{H}^{\text{cr}}(A_H) \text{ small}$

$\Rightarrow \quad H \text{ CR}$
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 $x_1 = \langle 2 \rangle \| V_1[2] \text{ and } x_2 = \langle 1 \rangle \| V_2[1]$. 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: \( \| M_1 \|_b = \| M_2 \|_b \)

\[
\begin{align*}
& h_K \\
& h_K \\
& h_K
\end{align*}
\]

\( v_1[1] \rightarrow v_1[2] \rightarrow v_1[3] = H_K(M_1) \)

\[
\begin{align*}
& h_K \\
& h_K \\
& h_K
\end{align*}
\]

\( v_2[1] \rightarrow v_2[2] \rightarrow v_2[3] = H_K(M_2) \)

\[
\begin{align*}
M_2[1] & \quad M_2[2] \quad <2> \\
M_1[1] & \quad M_1[2] \quad <2>
\end{align*}
\]

\[
\begin{align*}
\langle 2 \rangle \| V_1[2] ; \ & x_1 \leftarrow \langle 2 \rangle \| V_2[2] \\
\text{If } x_1 & \neq x_2 \text{ then return } x_1, x_2
\end{align*}
\]
Case 2: $\| M_1 \|_b = \| M_2 \|_b$

\[
\begin{align*}
& M_1^{[1]} \\
& h_k \quad v_1^{[1]} \\
& h_k \quad v_1^{[2]} \\
& h_k \quad v_1^{[3]} = H_K(M_1)
\end{align*}
\]

\[
\begin{align*}
& M_2^{[1]} \\
& h_k \quad v_2^{[1]} \\
& h_k \quad v_2^{[2]} \\
& h_k \quad v_2^{[3]} = H_K(M_2)
\end{align*}
\]

$x_1 \leftarrow \langle 2\rangle \| V_1^{[2]} \rangle; \quad x_2 \leftarrow \langle 2\rangle \| V_2^{[2]} \rangle$

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

Else // $V_1^{[2]} = V_2^{[2]}$
Case 2: \( \| M_1 \|_b = \| M_2 \|_b \)

\[
\begin{align*}
x_1 & \leftarrow \langle 2 \rangle || V_1[2] ; x_2 \leftarrow \langle 2 \rangle || V_2[2] \\
\text{If } x_1 \neq x_2 \text{ then return } x_1, x_2 \\
& \quad x_1 \leftarrow M_1[2] || V_1[1] ; x_2 \leftarrow M_2[2] || V_2[1] \\
& \quad \text{If } x_1 \neq x_2 \text{ then return } x_1, x_2
\end{align*}
\]
Case 2: \[ \| M_1 \|_b = \| M_2 \|_b \]

If \( x_1 \neq x_2 \) then return \( x_1, x_2 \)


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

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 $x_1, x_2$


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

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 $x_1, x_2$
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** $H(K, M)$

1. $M_1 \parallel M_2 \leftarrow M$
2. $V_1 \leftarrow h(K, M_1)$; $V_2 \leftarrow h(K, M_2)$
3. $V \leftarrow h(K, V_1 \parallel V_2)$
4. 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 design a keyless compression function

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

by

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

Is $H$ collision resistant?
How are compression functions designed?

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

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

by

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

Is $H$ collision resistant?

NO!

**adversary**  $A$

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

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

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

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

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 $E : \{0, 1\}^{512} \times \{0, 1\}^{160} \rightarrow \{0, 1\}^{160}$. 
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

<table>
<thead>
<tr>
<th>When</th>
<th>Against</th>
<th>Time</th>
<th>Who</th>
</tr>
</thead>
<tbody>
<tr>
<td>1993,1996</td>
<td>md5</td>
<td>$2^{16}$</td>
<td>[dBBo,Do]</td>
</tr>
<tr>
<td>2005</td>
<td>RIPEMD</td>
<td>$2^{18}$</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>SHA0</td>
<td>$2^{51}$</td>
<td>[JoCaLeJa]</td>
</tr>
<tr>
<td>2005</td>
<td>SHA0</td>
<td>$2^{40}$</td>
<td>[WaFeLaYu]</td>
</tr>
<tr>
<td>2005</td>
<td>SHA1</td>
<td>$2^{69}$</td>
<td>[WaYiYu]</td>
</tr>
<tr>
<td>2012</td>
<td>SHA1</td>
<td>$2^{60} - 2^{65}$</td>
<td>[St]</td>
</tr>
<tr>
<td>2005,2006</td>
<td>MD5</td>
<td>1 minute</td>
<td>[WaFeLaYu,LeWadW,Kl]</td>
</tr>
</tbody>
</table>

md5 is the compression function of MD5
SHA0 is an earlier, weaker version of SHA1
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 \textit{(boring)}
- Find two PDF documents whose hashes collide \textit{(more exciting)}
- Find two Win32 executables whose hashes collide \textit{(very exciting)}
- Break deployed cryptographic protocols \textit{(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:
  Takes 2 days on a PlayStation 3
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.
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?
National Institute for Standards and Technology (NIST) held a world-wide competition to develop a new hash function standard.

Contest webpage:

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


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.

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