A Survey on Hash Functions and Collision Attacks

Yan Dai and Ziqian Zhong

January 10, 2020

History [1]

- 1990. MD4 published.
- 1992. MD5 published.
- 1993. SHA-0 published.
- 1995. SHA-1 published. H. Dobbertin reports collisions in MD4 within seconds.
- 1996. H. Dobbertin reports collisions in the MD5 compression function.
- 2004. X. Wang's team reports collisions in MD5 in a few hours.
- 2005. X. Wang's team reports a 2^{69}-cost collision attack on SHA-1.
- 2007. Stevens et al. show chosen-prefix collision attacks on MD5.
- 2017. Stevens et al. report collisions in SHA-1.
- 2019. G. Leurent et al. report chosen-prefix collision attacks in SHA-1 are only a few times more expensive than identical ones.
- 2020. G. Leurent et al. report the first chosen-prefix collision attack on SHA-1. *(unpublished manuscript)*

[1] https://crypto.stackexchange.com/questions/60640/does-shattered-actually-show-sha-1-signed-certificates-are-unsafe

Structure

Following these works, we studied properties of hash functions and collision attacks on hash functions. The survey is consisted of three parts.

Structure

Following these works, we studied properties of hash functions and collision attacks on hash functions. The survey is consisted of three parts.

- In the first part, we introduce hash functions and their properties. We also prove two theorems about the properties of hash functions.

Structure

Following these works, we studied properties of hash functions and collision attacks on hash functions. The survey is consisted of three parts.

- In the first part, we introduce hash functions and their properties. We also prove two theorems about the properties of hash functions.
- In the second part, we focus on the constructions of hash functions.

Structure

Following these works, we studied properties of hash functions and collision attacks on hash functions. The survey is consisted of three parts.

- In the first part, we introduce hash functions and their properties. We also prove two theorems about the properties of hash functions.
- In the second part, we focus on the constructions of hash functions.
- In the last part, we study the collision attacks made on MD5 and SHA-1, and demostrate a MD5 collision attack.

Definition

Definition

A hash funtion *is a function mapping a space $\mathcal{M}$ to another space $\mathcal{T}$. Here, we call the space $\mathcal{M}$ as* message space, *and space $\mathcal{T}$ as* digest space*.*

Properties

Definition

For a hash function or a family of hash functions $F : \mathcal{M} \to \mathcal{T}$, we can define the following three properties of it:

1. ***Preimage resistant** (onewayness): Randomly given $t = F(m)$, there's only negligible chance for an efficient algorithm to find m' so that $F(m') = t$.*
2. ***Second preimage resistant**: Randomly given m, there's only negligible chance for an efficient algorithm to find $m' \neq m$ so that $F(m') = F(m)$.*
3. ***Collision resistant**: There's only negligible chance for an efficient algorithm to find two different m, m'-s so that $F(m') = F(m)$.*

Theorem 1

Theorem

If a hash function $H : \mathcal{M} \to \mathcal{T}$ is second preimage resistant, where $|\mathcal{M}|$ is infinite and $|\mathcal{T}|$ is finite, it must be one-way.

Theorem 1

Theorem

If a hash function $H : \mathcal{M} \to \mathcal{T}$ is second preimage resistant, where $|\mathcal{M}|$ is infinite and $|\mathcal{T}|$ is finite, it must be one-way.

Couterexample without the resitriction on the size of $|\mathcal{M}|$ and $|\mathcal{T}|$:

$$T(x) = \begin{cases} 0 \| x[k+1:n] & , x[1:k] = 0^k \\ 1 \| G(x) & , \text{otherwise} \end{cases}$$

Theorem 2

Theorem

If a hash function $H : \mathcal{M} \to \mathcal{T}$ is collision resistant, it must be second preimage resistant.

Merkle-Damgård Paradigm

- Introduced by R.C. Merkle in 1979.
-
$$t_i = \begin{cases} IV, & i = 0 \\ f(m_i, t_{i-1}), & 1 \le i \le \ell \end{cases}$$
$$F(m_1 \| m_2 \| \cdots \| m_\ell, IV) = t_\ell$$
- Proved independently by R.C. Merkle and I. Damgård in 1989.

MD5 Message-digest Algorithm

- Designed by Ronald Rivest in 1992.
- Mapping a message of length less than 2^{64} to 128 bits.
- Little Endian.

MD5 Message-digest Algorithm

- Designed by Ronald Rivest in 1992.
- Mapping a message of length less than 2^{64} to 128 bits.
- Little Endian.
- Three steps:
  1. Padding.
  2. Partitioning.
  3. Processing.

MD5 Message-digest Algorithm (cont'd)

- Initial vector:

$$IV = (67452301_{16}, EFCDAB89_{16}, 98BADCFE_{16}, 10325476_{16})$$

- 64 steps, split into 4 rounds consisting of 16 steps each.
- Non-linear function:

$$f_t(X,Y,Z) = \begin{cases} (X \wedge Y) \oplus (\overline{X} \wedge Z), & 0 \leq t < 16 \\ (Z \wedge X) \oplus (\overline{Z} \wedge Y), & 16 \leq t < 32 \\ X \oplus Y \oplus Z, & 32 \leq t < 48 \\ Y \oplus (X \vee \overline{Z}), & 48 \leq t < 64 \end{cases}$$

- Expand message:

$$W_t = \begin{cases} m_t, & 0 \leq t < 16 \\ m_{(1+5t) \bmod 16}, & 16 \leq t < 32 \\ m_{(5+3t) \bmod 16}, & 32 \leq t < 48 \\ m_{7t \bmod 16}, & 48 \leq t < 64 \end{cases}$$

MD5 Message-digest Algorithm (cont'd)

- Addtion constant: $\mathrm{AC}_t = \lfloor 2^{32} \cdot |\sin(t+1)| \rfloor$.
- Rotation constant: RC_t.
- In the t-th step, Q_{t+1} is computed as:

$$
\begin{aligned}
F_t &= f_t(Q_t, Q_{t-1}, Q_{t-2}) \\
T_t &= F_t + Q_{t-3} + \mathrm{AC}_t + W_t \\
R_t &= T_t \lll \mathrm{RC}_t \\
Q_{t+1} &= Q_t + R_t
\end{aligned}
$$

- $\mathrm{MD5Compress}(B, t) = (a + Q_{61}, b + Q_{64}, c + Q_{63}, d + Q_{62})$.

Secure Hash Algorithm 1

- Designed by United States National Security Agency in 1995.
- Mapping a message of length less than 2^{64} to 160 bits.
- **Big** Endian.

Secure Hash Algorithm 1

- Designed by United States National Security Agency in 1995.
- Mapping a message of length less than 2^{64} to 160 bits.
- **Big** Endian.
- Three steps:
  1. Padding.
  2. Partitioning.
  3. Processing.

Secure Hash Algorithm 1 (cont'd)

- Expand message: For all $16 \leq i < 80$,

$$m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) \lll 1$$

- Boolean function φ_i, see Table 2.1.
- Constant K_i, see Table 2.1.
- For all $0 \leq i < 80$:

$$A_{i+1} = (A_i \lll 5) + \varphi_i(A_{i-1}, A_{i-2} \lll 2, A_{i-3} \lll 2) + (A_{i-4} \lll 2) + K_i + m_i$$

- Calculate $\mathrm{SHA1Compress}(B, t)$ as:

$$(a + A_{80}, b + A_{79}, c + (A_{78} \lll 2), d + (A_{77} \lll 2), e + (A_{76} \lll 2))$$

Secure Hash Algorithm 1 (cont'd)

Table: Boolean functions and constants in SHA-1 Compression Function

Step i	$\varphi_i(X, Y, Z)$	K_i
$0 \leq i < 20$	$\varphi_{\text{IF}} = (X \wedge Y) \vee (\overline{X} \wedge Z)$	`0x5A827999`
$20 \leq i < 40$	$\varphi_{\text{XOR}} = X \oplus Y \oplus Z$	`0x6ED9EBA1`
$40 \leq i < 60$	$\varphi_{\text{MAJ}} = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$	`0x8F1BBCDC`
$60 \leq i < 80$	$\varphi_{\text{XOR}} = X \oplus Y \oplus Z$	`0xCA62C1D6`

Definition

Definition

For some chosen initial sequence A (usually consisted of some blocks), the identical prefix attack *needs to find two sequences of equal length B and B' (usually consisted of some blocks), so that $H(A\|B) = H(A\|B')$.*

Definition

For some chosen initial sequences A and A' with equal length (usually consisted of some blocks), the chosen prefix attack *needs to find two sequences of equal length B and B' (usually consisted of some blocks), so that $H(A\|B) = H(A'\|B')$.*

Methodology

- Differential Analysis.

Methodology

- Differential Analysis.
- Meet in the Middle.

Methodology

- Differential Analysis.
- Meet in the Middle.
- Multiple Message Blocks.

MD5 Collision Attack by Wang et al.

- Two messages consisting of two blocks each.

MD5 Collision Attack by Wang et al.

- Two messages consisting of two blocks each.
- Differential paths.
  - Binary signed digit representation:

$$\Delta x = \sum_{i=0}^{31} k_i 2^i$$

  - Difference in IHVs:

$$\delta\mathrm{IHV}_{k+1} = (2^{31}, 2^{31} + 2^{25}, 2^{31} + 2^{25}, 2^{31} + 2^{25})$$

  - Difference in first blocks:

$$\delta m_4 = 2^{31}, \delta m_{11} = 2^{15}, \delta m_{14} = 2^{31}$$

MD5 Collision Attack by Wang et al. (cont'd)

- Sufficient conditions:

Symbol	Condition on $Q_t[i]$	direct or indirect
.	none	direct
0	$Q_t[i] = 0$	direct
1	$Q_t[i] = 1$	direct
^	$Q_t[i] = Q_{t-1}[i]$	indirect
!	$Q_t[i] = \overline{Q_{t-1}[i]}$	indirect

MD5 Collision Attack by Wang et al. (cont'd)

- Sufficient conditions:

Symbol	Condition on $Q_t[i]$	direct or indirect
.	none	direct
0	$Q_t[i] = 0$	direct
1	$Q_t[i] = 1$	direct
^	$Q_t[i] = Q_{t-1}[i]$	indirect
!	$Q_t[i] = \overline{Q_{t-1}[i]}$	indirect

- Multi-message modification.

An Example of Multiple Modification

Suppose that $Q_{17}[33]$ is now 1 instead of 0, we can correct it by modifying $m_1, m_3, \cdots, m_5$:

1. $\hat{m}_1 \leftarrow m_1 + 2^{26}$, getting a new $\hat{Q}_2$.
2. $\hat{m}_2 \leftarrow ((Q_3 - \hat{Q}_2) \gg 17) - Q_{-1} - F(\hat{Q}_2, Q_1, Q_0) - \mathrm{AC}_2$.
3. $\hat{m}_3 \leftarrow ((Q_4 - Q_3) \gg 22) - Q_0 - F(Q_3, \hat{Q}_2, Q_1) - \mathrm{AC}_3$.
4. $\hat{m}_4 \leftarrow ((Q_5 - Q_4) \gg 7) - Q_1 - F(Q_4, Q_3, \hat{Q}_2) - \mathrm{AC}_4$.
5. $\hat{m}_5 \leftarrow ((Q_6 - Q_5) \gg 12) - \hat{Q}_2 - F(Q_5, Q_4, Q_3) - \mathrm{AC}_5$.

An Example of Multiple Modification

Suppose that $Q_{17}[33]$ is now 1 instead of 0, we can correct it by modifying $m_1, m_3, \cdots, m_5$:

1. $\hat{m}_1 \leftarrow m_1 + 2^{26}$, getting a new $\hat{Q}_2$.
2. $\hat{m}_2 \leftarrow ((Q_3 - \hat{Q}_2) \gg 17) - Q_{-1} - F(\hat{Q}_2, Q_1, Q_0) - \mathrm{AC}_2$.
3. $\hat{m}_3 \leftarrow ((Q_4 - Q_3) \gg 22) - Q_0 - F(Q_3, \hat{Q}_2, Q_1) - \mathrm{AC}_3$.
4. $\hat{m}_4 \leftarrow ((Q_5 - Q_4) \gg 7) - Q_1 - F(Q_4, Q_3, \hat{Q}_2) - \mathrm{AC}_4$.
5. $\hat{m}_5 \leftarrow ((Q_6 - Q_5) \gg 12) - \hat{Q}_2 - F(Q_5, Q_4, Q_3) - \mathrm{AC}_5$.

Successful as there is no condition on Q_2!

Improvements

- Improvements by V. Klima by using:
  $m_t = ((Q_{t+1} - Q_t) \gg \mathrm{RC}_t) - f_t(Q_t, Q_{t-1}, Q_{t-2}) - Q_{t-3} - \mathrm{AC}_t$

Improvements

- Improvements by V. Klima by using:
  $m_t = ((Q_{t+1} - Q_t) \gg \mathrm{RC}_t) - f_t(Q_t, Q_{t-1}, Q_{t-2}) - Q_{t-3} - \mathrm{AC}_t$
- Improved multi-message modification technique by V. Klima.

Improvements

- Improvements by V. Klima by using:
  $m_t = ((Q_{t+1} - Q_t) \gg \mathrm{RC}_t) - f_t(Q_t, Q_{t-1}, Q_{t-2}) - Q_{t-3} - \mathrm{AC}_t$
- Improved multi-message modification technique by V. Klima.
- Improvements by M. Stevens by using $Q_{t+1} - Q_t = R_t = T_t \ll \mathrm{RC}_t$.

Improvements

- Improvements by V. Klima by using:
  $m_t = ((Q_{t+1} - Q_t) \gg \mathrm{RC}_t) - f_t(Q_t, Q_{t-1}, Q_{t-2}) - Q_{t-3} - \mathrm{AC}_t$
- Improved multi-message modification technique by V. Klima.
- Improvements by M. Stevens by using $Q_{t+1} - Q_t = R_t = T_t \ll \mathrm{RC}_t$.
- Tunnels by V. Klima.

Improvements

- Improvements by V. Klima by using:
  $m_t = ((Q_{t+1} - Q_t) \gg \mathrm{RC}_t) - f_t(Q_t, Q_{t-1}, Q_{t-2}) - Q_{t-3} - \mathrm{AC}_t$
- Improved multi-message modification technique by V. Klima.
- Improvements by M. Stevens by using $Q_{t+1} - Q_t = R_t = T_t \ll \mathrm{RC}_t$.
- Tunnels by V. Klima.
- More differential paths constructed by M. Stevens.

An Example of the Improvements by M. Stevens

1. δR_4 should be -2^6 instead of 2^6.
2. $\Delta T_4 = -2^{31}$.
3. $T_4[31] = 1$.
4. $Q_4[6] = Q_5[6] = 0,\ Q_5[5] = 1$.

An Example Tunnel: Q_9-tunnel

Table: Conditions for Q_9, Q_{10} and Q_{11}

t	Conditions on $Q_t[31], Q_t[30], \cdots, Q_t[0]$
9	11111011 ...10000 0.1^1111 00111101
10	0111.... 0..11111 1101...0 01....00
11	0010.... ....0001 1100...0 11....10

An Example Tunnel: Q_9-tunnel

Table: Conditions for Q_9, Q_{10} and Q_{11}

t	Conditions on $Q_t[31], Q_t[30], \cdots, Q_t[0]$
9	11111011 ...10000 0.1^1111 00111101
10	0111.... 0..11111 1101...0 01....00
11	0010.... ....0001 1100...0 11....10

- Only $m_8, m_9, m_{10}, m_{11}, m_{12}$ will be affected.

An Example Tunnel: Q_9-tunnel

Table: Conditions for Q_9, Q_{10} and Q_{11}

t	Conditions on $Q_t[31], Q_t[30], \cdots, Q_t[0]$
9	11111011 ...10000 0.1^1111 00111101
10	0111.... 0..11111 1101...0 01....00
11	0010.... ....0001 1100...0 11....10

- Only $m_8, m_9, m_{10}, m_{11}, m_{12}$ will be affected.
- If $Q_{11}[22] = 1$, $F_{11}[22] = f_{11}(Q_{11}[22], Q_{10}[22], Q_9[22]) = (Q_{11}[22] \wedge Q_{10}[22]) \oplus (\overline{Q_{11}}[22] \wedge Q_9[22]) = Q_{10}[22]$!

An Example Tunnel: Q_9-tunnel

Table: Conditions for Q_9, Q_{10} and Q_{11}

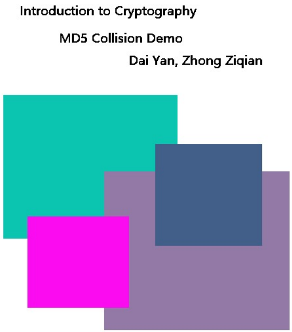
t	Conditions on $Q_t[31], Q_t[30], \cdots, Q_t[0]$
9	11111011 ...10000 0.1^1111 00111101
10	0111.... 0..11111 1101...0 01....00
11	0010.... ....0001 1100...0 11....10

- Only $m_8, m_9, m_{10}, m_{11}, m_{12}$ will be affected.
- If $Q_{11}[22] = 1$, $F_{11}[22] = f_{11}(Q_{11}[22], Q_{10}[22], Q_9[22]) = (Q_{11}[22] \wedge Q_{10}[22]) \oplus (\overline{Q_{11}}[22] \wedge Q_9[22]) = Q_{10}[22]$!
- If $Q_{10}[22] = 0$, $F_{10}[22] = f_{10}(Q_{10}[22], Q_9[22], Q_8[22]) = (Q_{10}[22] \wedge Q_9[22]) \oplus (\overline{Q_{10}}[22] \wedge Q_8[22]) = Q_8[22]$!

Collision Finding Algorithm

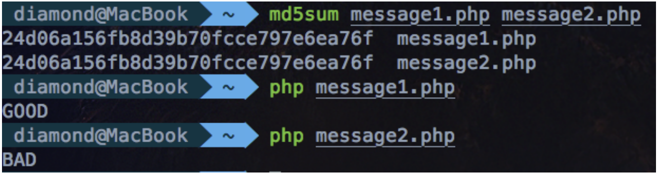
1. Randomly choose $Q_1, Q_3, Q_4, \cdots, Q_{16}$ fulfilling conditions, and then calculate $m_0, m_6, m_7, \cdots, m_{15}$.
2. Repeat choosing Q_{17} fulfilling conditions, until $Q_{18}, Q_{19}, Q_{20}, Q_{21}$ are fulfilling conditions.
3. Use tunnels $\mathcal{T}(Q_9, m_{10}), \mathcal{T}(Q_9, m_9)$ and $\mathcal{T}(Q_{10}, m_{10})$, until conditions on $Q_{22}, Q_{23}, \cdots, Q_{64}, T_{22}, T_{34}$ and the IHV-conditions for the next block are fulfilled.

A Demo



MD5 digest value: `ec040305fffbf4c3e7aeed84bffc77ed`, found in 35 seconds on `Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz` (single thread).

Another Demo



Two PHP files of the following format:

`PADDING if (message1 == message1) GOOD else BAD`

`PADDING if (message2 == message1) GOOD else BAD`

MD5 digest value: `24d06a156fb8d39b70fcce797e6ea76f`, found in 20 seconds on `Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz` (single thread).

Local collisions

- Considering cancel a difference of 2^b introduced in m_t.

Local collisions

- Considering cancel a difference of 2^b introduced in m_t.
- Chabaud et al.: introduce differences $\{2^{(b+5) \bmod 32}, 2^b, 2^{(b+30) \bmod 32}, 2^{(b+30) \bmod 32}, 2^{(b+30) \bmod 32}\}$ (without signs) to $\{m_{t+1}, m_{t+2}, m_{t+3}, m_{t+4}, m_{t+5}\}$, correspondingly. Work with some probability.

Local collisions

- Considering cancel a difference of 2^b introduced in m_t.
- Chabaud et al.: introduce differences $\{2^{(b+5) \bmod 32}, 2^b, 2^{(b+30) \bmod 32}, 2^{(b+30) \bmod 32}, 2^{(b+30) \bmod 32}\}$ (without signs) to $\{m_{t+1}, m_{t+2}, m_{t+3}, m_{t+4}, m_{t+5}\}$, correspondingly. Work with some probability.
- We call this introducing a local collision on bit b at position t. Different local collisions can overlap and interact with each other.

Disturbance Vector

We denote the difference of the two messages words (without signs) be $\{\mathrm{DW}_t\}_{t=0}^{79}$ and we define the *disturbance vector* $\{\mathrm{DV}_t\}_{t=0}^{79}$, where the 2^b bit of DV_t is 1 if and only if there is a local collision starting in position t in bit b.

Disturbance Vector

We denote the difference of the two messages words (without signs) be $\{\mathrm{DW}_t\}_{t=0}^{79}$ and we define the *disturbance vector* $\{\mathrm{DV}_t\}_{t=0}^{79}$, where the 2^b bit of DV_t is 1 if and only if there is a local collision starting in position t in bit b.
By definition, we have

$$\begin{aligned}\mathrm{DW}_t =& \mathrm{DV}_t \oplus (\mathrm{DV}_{t-1} \gg 5) \oplus \mathrm{DV}_{t-2} \oplus (\mathrm{DV}_{t-3} \gg 30) \oplus \\ & (\mathrm{DV}_{t-4} \gg 30) \oplus (\mathrm{DV}_{t-5} \gg 30)\end{aligned}$$

Disturbance Vector

We denote the difference of the two messages words (without signs) be $\{\mathrm{DW}_t\}_{t=0}^{79}$ and we define the *disturbance vector* $\{\mathrm{DV}_t\}_{t=0}^{79}$, where the 2^b bit of DV_t is 1 if and only if there is a local collision starting in position t in bit b.
By definition, we have

$$\begin{aligned}\mathrm{DW}_t =& \mathrm{DV}_t \oplus (\mathrm{DV}_{t-1} \gg 5) \oplus \mathrm{DV}_{t-2} \oplus (\mathrm{DV}_{t-3} \gg 30) \oplus \\ & (\mathrm{DV}_{t-4} \gg 30) \oplus (\mathrm{DV}_{t-5} \gg 30)\end{aligned}$$

Therefore, we'll also have

$$\mathrm{DV}_i = (\mathrm{DV}_{i-3} \oplus \mathrm{DV}_{i-8} \oplus \mathrm{DV}_{i-14} \oplus \mathrm{DV}_{i-16}) \ll 1$$

Disturbance Vector

At first glance, three additional restrictions are posed for the sequence DV:

Disturbance Vector

At first glance, three additional restrictions are posed for the sequence DV:

1. $\mathrm{DV}_{-5} = \mathrm{DV}_{-4} = \mathrm{DV}_{-3} = \mathrm{DV}_{-2} = \mathrm{DV}_{-1} = 0$. This is the natural result, as initial states can never be modified.

Disturbance Vector

At first glance, three additional restrictions are posed for the sequence DV:

1. $\mathrm{DV}_{-5} = \mathrm{DV}_{-4} = \mathrm{DV}_{-3} = \mathrm{DV}_{-2} = \mathrm{DV}_{-1} = 0$. This is the natural result, as initial states can never be modified.
2. $\mathrm{DV}_{75} = \mathrm{DV}_{76} = \mathrm{DV}_{77} = \mathrm{DV}_{78} = \mathrm{DV}_{79} = 0$. This is also necessary for the local collision to be complete.

Disturbance Vector

At first glance, three additional restrictions are posed for the sequence DV:

1. $\mathrm{DV}_{-5} = \mathrm{DV}_{-4} = \mathrm{DV}_{-3} = \mathrm{DV}_{-2} = \mathrm{DV}_{-1} = 0$. This is the natural result, as initial states can never be modified.
2. $\mathrm{DV}_{75} = \mathrm{DV}_{76} = \mathrm{DV}_{77} = \mathrm{DV}_{78} = \mathrm{DV}_{79} = 0$. This is also necessary for the local collision to be complete.
3. For every bit b and $i = 0, 1 \cdots 15$, at most one of $\mathrm{DV}_i[b]$ and $\mathrm{DV}_{i+1}[b]$ is non-zero. The boolean function (namely *IF*) in the first steps makes it impossible to modify these two positions together.

Disturbance Vector

These restrictions are all removed by Wang et al..

Disturbance Vector

These restrictions are all removed by Wang et al..

- The first and the second restriction: constructing separately on the first few steps.

Disturbance Vector

These restrictions are all removed by Wang et al..

- The first and the second restriction: constructing separately on the first few steps.
- The third restriction: using a two-block collision instead of using a single-block one.

Disturbance Vector

These restrictions are all removed by Wang et al..

- The first and the second restriction: constructing separately on the first few steps.
- The third restriction: using a two-block collision instead of using a single-block one.

With these restrictions removed, much better disturbance vectors can be found and thus lowering the attacking complexity.

Disturbance Vector and Differential Path

While disturbance vector determined the modified positions in the late steps, we still need to construct the first steps in the differential path to lead to the disturbance vector.

Disturbance Vector and Differential Path

While disturbance vector determined the modified positions in the late steps, we still need to construct the first steps in the differential path to lead to the disturbance vector.
A differential path consists of bit conditions - restrictions posed to both the initial bits and the modified bits.

Disturbance Vector and Differential Path

While disturbance vector determined the modified positions in the late steps, we still need to construct the first steps in the differential path to lead to the disturbance vector.
A differential path consists of bit conditions - restrictions posed to both the initial bits and the modified bits.
Notice that only adding restrictions to the difference positions is not efficient enough, and making such a path non-linear can facilitate the control on the messages.

Disturbance Vector and Differential Path

While disturbance vector determined the modified positions in the late steps, we still need to construct the first steps in the differential path to lead to the disturbance vector.
A differential path consists of bit conditions - restrictions posed to both the initial bits and the modified bits.
Notice that only adding restrictions to the difference positions is not efficient enough, and making such a path non-linear can facilitate the control on the messages.
Bit conditions can exist not only for the working states, but also for the messages.

Selecting Disturbance Vector

S. Manuel noticed that all interesting disturbance vectors found by his search and previously shown in the literature, belong to two classes.

Selecting Disturbance Vector

S. Manuel noticed that all interesting disturbance vectors found by his search and previously shown in the literature, belong to two classes.

- $\mathrm{I}(i, b)$: $\mathrm{DV}_i = \mathrm{DV}_{i+1} = \cdots = \mathrm{DV}_{i+14} = 0$ and $\mathrm{DV}_{i+15} = 2^b$.

Selecting Disturbance Vector

S. Manuel noticed that all interesting disturbance vectors found by his search and previously shown in the literature, belong to two classes.

- $\mathrm{I}(i, b)$: $\mathrm{DV}_i = \mathrm{DV}_{i+1} = \cdots = \mathrm{DV}_{i+14} = 0$ and $\mathrm{DV}_{i+15} = 2^b$.
- $\mathrm{II}(i, b)$: $\mathrm{DV}_{i+1} = \mathrm{DV}_{i+3} = 2^{(b+31) \bmod 32}$, $\mathrm{DV}_{i+15} = 2^b$, and $\mathrm{DV}_i = \mathrm{DV}_{i+2} = \mathrm{DV}_{i+4} = \mathrm{DV}_{i+5} = \cdots = \mathrm{DV}_{i+14} = 0$

Selecting Disturbance Vector

It's complicated to analyze the exact impact of every interaction. Try to trace the success probability of all possible differential paths instead.

Selecting Disturbance Vector

It's complicated to analyze the exact impact of every interaction. Try to trace the success probability of all possible differential paths instead. *Optimal joint local-collision analysis*, Stevens. A meet-in-the-middle approach is used and is accelerated by combining equivalent states.

Selecting Disturbance Vector

It's complicated to analyze the exact impact of every interaction. Try to trace the success probability of all possible differential paths instead. *Optimal joint local-collision analysis*, Stevens. A meet-in-the-middle approach is used and is accelerated by combining equivalent states. $\mathrm{II}(52, 0)$ is shown to be the most efficient.

Selecting Differential Path

The first differential path is hand-crafted by Wang et al..

Selecting Differential Path

The first differential path is hand-crafted by Wang et al..
Automatically searches are proposed by Yajima et al. and Stevens et al..

Selecting Differential Path

The first differential path is hand-crafted by Wang et al.. Automatically searches are proposed by Yajima et al. and Stevens et al.. Their methods are also based on a meet-in-the-middle approach. The path with the most number of degrees of freedom is chosen.

Collision Search Process

The favorable input differences can be computed with *optimal joint local-collision analysis*. The first block will lead the difference to one such difference and the second block will lead it to zero.

Collision Search Process

The favorable input differences can be computed with *optimal joint local-collision analysis*. The first block will lead the difference to one such difference and the second block will lead it to zero.

The attack is performed with two stages.

Collision Search Process

The favorable input differences can be computed with *optimal joint local-collision analysis*. The first block will lead the difference to one such difference and the second block will lead it to zero.

The attack is performed with two stages.

- The first stage finds a message block that satisfy the bit relations up to round 33. This stage is the most complicated and need to be speeded up using message modification techniques described below.

Collision Search Process

The favorable input differences can be computed with *optimal joint local-collision analysis*. The first block will lead the difference to one such difference and the second block will lead it to zero.
The attack is performed with two stages.

- The first stage finds a message block that satisfy the bit relations up to round 33. This stage is the most complicated and need to be speeded up using message modification techniques described below.
- In the second stage, the message block is extended and verified.

Message Modification Techniques

Neutral bit: a set of bits in the message can be flipped while preserving the bit conditions with high probability up to round n.

Message Modification Techniques

Neutral bit: a set of bits in the message can be flipped while preserving the bit conditions with high probability up to round n.
Boomerangs: a few bits that flip only a state bit when being flipped together with high probability up to some round n.

Message Modification Techniques

Automatic search for neutral bits and boomerangs.

Message Modification Techniques

Automatic search for neutral bits and boomerangs.
Thousands of messages that can pass the first rounds are first randomly generated. Neutral bits and boomerangs are verified in such a set. Only ones with success probability at least 0.9 are kept.

Message Modification Techniques

Automatic search for neutral bits and boomerangs.

Thousands of messages that can pass the first rounds are first randomly generated. Neutral bits and boomerangs are verified in such a set. Only ones with success probability at least 0.9 are kept.

Boomerang' candidates can be found by negating inner states and computing the corresponding change to message words.

Message Modification Techniques

Automatic search for neutral bits and boomerangs.

Thousands of messages that can pass the first rounds are first randomly generated. Neutral bits and boomerangs are verified in such a set. Only ones with success probability at least 0.9 are kept.

Boomerang' candidates can be found by negating inner states and computing the corresponding change to message words.

Neutral bits' candidates can be found by expressing bit-relations as linear equations and explicitly constructing a solution.

Case Study: the First Full SHA-1 Collision Attack

Announced by Stevens et al. in 2017. Around $2^{63.1}$ SHA-1 computations.

Case Study: the First Full SHA-1 Collision Attack

Announced by Stevens et al. in 2017. Around $2^{63.1}$ SHA-1 computations. Disturbance vector $\mathrm{II}(52, 0)$.

Case Study: the First Full SHA-1 Collision Attack

Announced by Stevens et al. in 2017. Around $2^{63.1}$ SHA-1 computations. Disturbance vector $\mathrm{II}(52, 0)$.Target difference set size 192.

Case Study: the First Full SHA-1 Collision Attack

Announced by Stevens et al. in 2017. Around $2^{63.1}$ SHA-1 computations.
Disturbance vector $\mathrm{II}(52, 0)$.Target difference set size 192.
Differential path is constructed with a meet-in-the-middle approach and a SAT construction approach.

Case Study: the First Full SHA-1 Collision Attack

Announced by Stevens et al. in 2017. Around $2^{63.1}$ SHA-1 computations.
Disturbance vector $\mathrm{II}(52, 0)$.Target difference set size 192.
Differential path is constructed with a meet-in-the-middle approach and a SAT construction approach.
Extra tricks introduced.

Case Study: the First Full SHA-1 Collision Attack

Announced by Stevens et al. in 2017. Around $2^{63.1}$ SHA-1 computations.
Disturbance vector $\mathrm{II}(52, 0)$.Target difference set size 192.
Differential path is constructed with a meet-in-the-middle approach and a SAT construction approach.
Extra tricks introduced.
First block computation on CPUs, second block computation on GPUs.

Conclusion

In this survey, we studied properties of hash functions and proved two theorems about hash function. We also showed how an identical prefix collision for MD5 and SHA-1 can be produced with differential cryptanalysis. We hope that the method used to analyse these two hash functions can provide useful insights on analyzing and designing hash functions.

Thank You!