Collisions in MD5

... and how to use them for fun and profit.

Mar. 10, 2009

Antoine Delignat-Lavaud
Computer Science Department,
École Normale Supérieure de Cachan

Outline

1. **Hash functions and their uses**
   - What is a hash function ?
   - The Merkle-Damgård construction
   - Message-Digest algorithm 5 (MD5)
2. **Differential cryptanalysis of MD5**
   - Wang's differential path
   - Deriving a sufficient conditions set
   - Building the collision
3. **Conclusion**

What is a hash function ?

Hash function

Let Σ, Ω be two finite alphabets and n a positive integer. A hash function f is a map :

$$f : \Sigma^* \longrightarrow \Omega^n$$

What is a hash function ?

Hash function

Let Σ, Ω be two finite alphabets and n a positive integer. A hash function f is a map :

$$f : \Sigma^* \longrightarrow \Omega^n$$

Cryptographic hash functions

In cryptography, a hash function is used to compute the *signature* of an input. As such, it is expected to be :

What is a hash function ?

Hash function

Let Σ, Ω be two finite alphabets and n a positive integer. A hash function f is a map :

$$f : \Sigma^* \longrightarrow \Omega^n$$

Cryptographic hash functions

In cryptography, a hash function is used to compute the *signature* of an input. As such, it is expected to be :

1. Easy to compute for any input

What is a hash function ?

Hash function

Let Σ, Ω be two finite alphabets and n a positive integer. A hash function f is a map :

$$f : \Sigma^* \longrightarrow \Omega^n$$

Cryptographic hash functions

In cryptography, a hash function is used to compute the *signature* of an input. As such, it is expected to be :

1. Easy to compute for any input
2. Preimage resistant (given $s \in \Omega^n$, it is hard to find $\omega \in \Sigma^*$ such that $f(\omega) = s$)

What is a hash function ?

Hash function

Let Σ, Ω be two finite alphabets and n a positive integer. A hash function f is a map :

$$f : \Sigma^* \longrightarrow \Omega^n$$

Cryptographic hash functions

In cryptography, a hash function is used to compute the *signature* of an input. As such, it is expected to be :

1. Easy to compute for any input
2. Preimage resistant (given $s \in \Omega^n$, it is hard to find $\omega \in \Sigma^*$ such that $f(\omega) = s$)
3. Second preimage resistant (given $\omega_1 \in \Sigma^*$, it is hard to find $\omega_2 \neq \omega_1$ such that $f(\omega_1) = f(\omega_2)$)

What is a hash function ?

Hash function

Let Σ, Ω be two finite alphabets and n a positive integer. A hash function f is a map :

$$f : \Sigma^* \longrightarrow \Omega^n$$

Cryptographic hash functions

In cryptography, a hash function is used to compute the *signature* of an input. As such, it is expected to be :

1. Easy to compute for any input
2. Preimage resistant (given $s \in \Omega^n$, it is hard to find $\omega \in \Sigma^*$ such that $f(\omega) = s$)
3. Second preimage resistant (given $\omega_1 \in \Sigma^*$, it is hard to find $\omega_2 \neq \omega_1$ such that $f(\omega_1) = f(\omega_2)$)
4. Collision resistant (it is hard to find $\omega_1, \omega_2 \in \Sigma^*$ such that $f(\omega_1) = f(\omega_2)$)

What does “hard” mean ?

What does “hard” mean ?

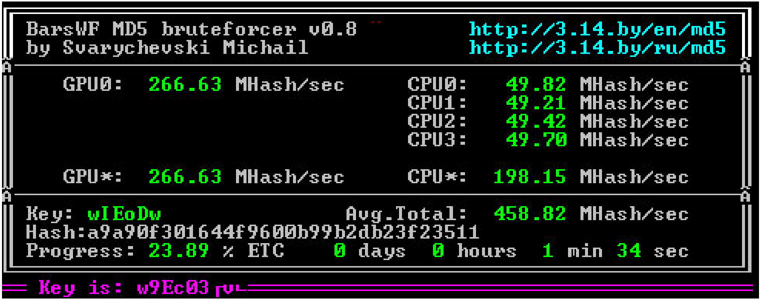
- Birthday attack : $\mathcal{O}(|\Omega|^{\frac{n}{2}})$

What does "hard" mean ?

- Birthday attack : $\mathcal{O}(|\Omega|^{\frac{n}{2}})$
- Brute force can be effective ! (up to 1 billion hashes per second on a desktop PC)

What does "hard" mean ?

- Birthday attack : $\mathcal{O}(|\Omega|^{\frac{n}{2}})$
- Brute force can be effective ! (up to 1 billion hashes per second on a desktop PC)
- MD5 : $\Omega = \{0, 1\}$, $n = 128$ is too low for current processing power.



Consequences

- MD5-hashed password are easy to crack : at most 2 days for a 68^8 keyspace using \$500 worth of hardware, a mere 2 more days to crack UNIX's `$1$-crypt` function

Consequences

- MD5-hashed password are easy to crack : at most 2 days for a 68^8 keyspace using \$500 worth of hardware, a mere 2 more days to crack UNIX's `$1$-crypt` function
- Derived authentication methods at risk (e.g. CRAM-MD5)

Consequences

- MD5-hashed password are easy to crack : at most 2 days for a 68^8 keyspace using \$500 worth of hardware, a mere 2 more days to crack UNIX's `$1$-crypt` function
- Derived authentication methods at risk (e.g. CRAM-MD5)
- Random collisions, not very significant.

Consequences

- MD5-hashed password are easy to crack : at most 2 days for a 68^8 keyspace using $500 worth of hardware, a mere 2 more days to crack UNIX's `$1$-crypt` function
- Derived authentication methods at risk (e.g. CRAM-MD5)
- Random collisions, not very significant.
- But we want collisions on *meaningful* data, $< 2^{64}$ calls to the MD5.

The Merkle-Damgård construction

Goals

- Compress input : variable length $\rightarrow$ fixed length

The Merkle-Damgård construction

Goals

- Compress input : variable length $\rightarrow$ fixed length
- Balence strength and simplicity

The Merkle-Damgård construction

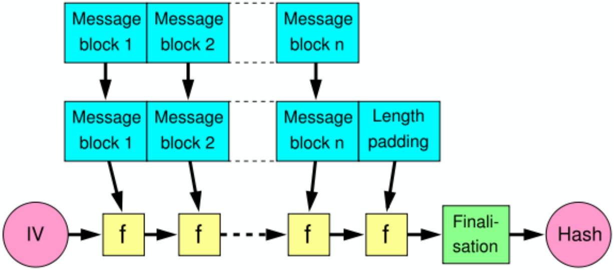
Goals

- Compress input : variable length $\rightarrow$ fixed length
- Balence strength and simplicity
- Strong “avalanche” effect

The Merkle-Damgård construction

Goals

- Compress input : variable length $\rightarrow$ fixed length
- Balence strength and simplicity
- Strong "avalanche" effect



The Merkle-Damgård construction

Construction

- $f : \{0,1\}^n \times \{0,1\}^m \longrightarrow \{0,1\}^n$ is the compression function.
- m is the block size, n the digest size
- IV is a fixed initialization vector
- Length padding is critical for the security of the construction

Properties

- Proven strength : f fix-start collision resistant and fix-start preimage resistant implies cryptographic strength

The Merkle-Damgård construction

Construction

- $f : \{0,1\}^n \times \{0,1\}^m \longrightarrow \{0,1\}^n$ is the compression function.
- m is the block size, n the digest size
- IV is a fixed initialization vector
- Length padding is critical for the security of the construction

Properties

- Proven strength : f fix-start collision resistant and fix-start preimage resistant implies cryptographic strength
- Convenient : single function for the whole process

The Merkle-Damgård construction

Construction

- $f : \{0,1\}^n \times \{0,1\}^m \longrightarrow \{0,1\}^n$ is the compression function.
- m is the block size, n the digest size
- IV is a fixed initialization vector
- Length padding is critical for the security of the construction

Properties

- Proven strength : f fix-start collision resistant and fix-start preimage resistant implies cryptographic strength
- Convenient : single function for the whole process
- Can wreak havoc if compression function has collisions

Message-Digest algorithm 5 (MD5)

Description

- Merkle-Damgård based

Message-Digest algorithm 5 (MD5)

Description

- Merkle-Damgård based
- $n = 128$, $m = 512$

Message-Digest algorithm 5 (MD5)

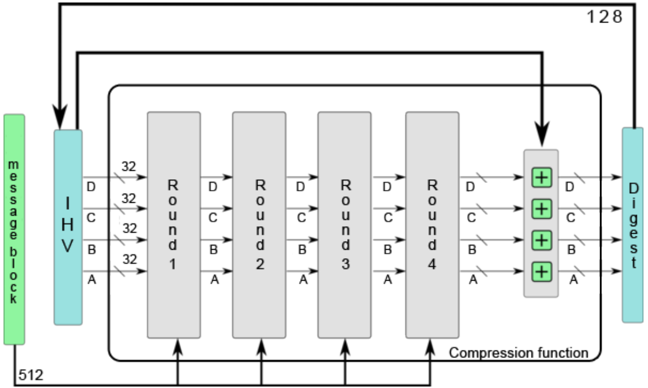
Description

- Merkle-Damgård based
- $n = 128$, $m = 512$
- Designed by Ronald Rivest in 1991

Message-Digest algorithm 5 (MD5)

Description

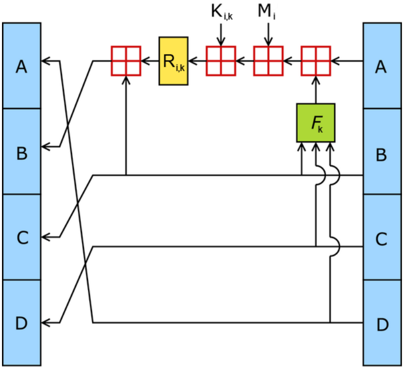
- Merkle-Damgård based
- $n = 128$, $m = 512$
- Designed by Ronald Rivest in 1991



A round of MD5

Description

- One different non-linear function $F_{k \in [1,4]}$ per round
- 16 operations per round on 32-bit slices $M_{i \in [1,16]}$ of the 512 bit input block.
- A constant $K_{i,k}$ is added at each round and a left bit rotation $R_{i,k}$ is applied



Algorithm parameters

Non-linear function

$$F_1(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$F_2(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$F_3(X,Y,Z) = X \oplus Y \oplus Z$$
$$F_4(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

$K_{i,k}$ "nothing up my sleeve" constants

$$K_{i,k} = \lfloor 2^{32} |\sin(4 * (k-1) + i)| \rfloor$$

Initialization vector

$$\begin{aligned} A_0 &= \texttt{0x67452301} \\ B_0 &= \texttt{0xEFCDAB89} \\ C_0 &= \texttt{0x98BADCFE} \\ D_0 &= \texttt{0x10325476} \end{aligned}$$

Differential cryptanalysis of MD5

Differential Cryptanalysis

- Familly of cryptanalysis methods
- Known as early as 1974 by the NSA, published 15 years later !
- Explore how variations in the input translate to the output, "tickle attack"

Differential path and collisions

- Message value (M, M') unimportant, only difference $\Delta M = M' - M$ matters

Differential cryptanalysis of MD5

Differential Cryptanalysis

- Familly of cryptanalysis methods
- Known as early as 1974 by the NSA, published 15 years later !
- Explore how variations in the input translate to the output, “tickle attack”

Differential path and collisions

- Message value (M,M') unimportant, only difference $\Delta M = M' - M$ matters
- We search for a sequence of differences Δ_i such that $\exists i, \Delta_i = 0$, i.e such that the difference eventually disappear after an unspecified number of compressions

Differential cryptanalysis of MD5

Differential Cryptanalysis

- Familly of cryptanalysis methods
- Known as early as 1974 by the NSA, published 15 years later !
- Explore how variations in the input translate to the output, “tickle attack”

Differential path and collisions

- Message value (M, M') unimportant, only difference $\Delta M = M' - M$ matters
- We search for a sequence of differences Δ_i such that $\exists i, \Delta_i = 0$, i.e such that the difference eventually disappear after an unspecified number of compressions
- This sequence of differentials is a roadmap to find the collision

Antoine Delignat-Lavaud

Differential cryptanalysis of MD5

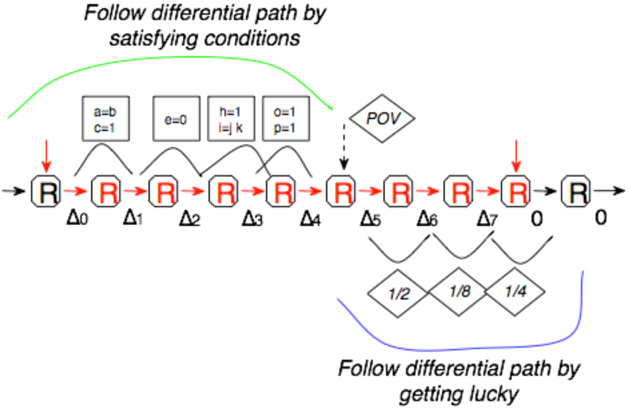
Differential Cryptanalysis

- Familly of cryptanalysis methods
- Known as early as 1974 by the NSA, published 15 years later !
- Explore how variations in the input translate to the output, “tickle attack”

Differential path and collisions

- Message value (M, M') unimportant, only difference $\Delta M = M' - M$ matters
- We search for a sequence of differences Δ_i such that $\exists i, \Delta_i = 0$, i.e such that the difference eventually disappear after an unspecified number of compressions
- This sequence of differentials is a roadmap to find the collision
- ... but they’re hard to find

Exploiting a differential path

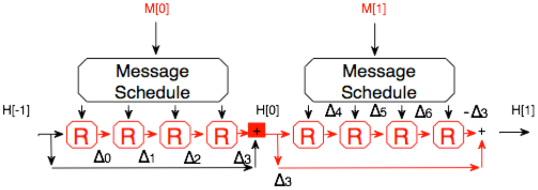


Multi-bloc differential path

Wang et al. differential path construction

We consider a more general problem : find $(M_0, M_0'), (M_1, M_1')$ such that we have for any IHV_k :

$$
\begin{array}{ccccccccc}
\ldots & \xrightarrow{M_k} & IVH_k & \xrightarrow{M_0} & IVH_{k+1} & \xrightarrow{M_1} & IHV_{k+1} & \xrightarrow{\cdots} \\
 & & = & & \neq & & = & \\
\ldots & \xrightarrow{M_k} & IVH_k & \xrightarrow{M_0'} & IVH_{k+1}' & \xrightarrow{M_1'} & IHV_{k+1} & \xrightarrow{\cdots}
\end{array}
$$



How to differentiate smartly

Differential notations

- M' denotes the collistion dual message of M
- $\Delta X = X' - X$ where $-$ denotes integer modular difference
- Applies to 32-bit components, e.g $\Delta IHV = (\Delta A, \Delta B, \Delta C, \Delta D)$
- $+2^{15} - 2^{8}$ means bit 15 flipped from 0 to 1 and bit 8 flipped from 1 to 0

Wang's collision path

$\delta m_4 = +2^{31}, \quad \delta m_{11} = +2^{15}, \quad \delta m_{14} = +2^{31}, \quad \delta m_i = 0, i \notin \{4, 11, 14\}$

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
$0-3$	$-$	$-$	$-$	$-$	$\cdot$
4	$-$	$-$	2^{31}	2^{31}	7
5	$+2^6 \ldots +2^{21}, -2^{22}$	$2^{11}+2^{19}$	$-$	$2^{11}+2^{19}$	12
6	$-2^6+2^{23}+2^{31}$	$-2^{10}-2^{14}$	$-$	$-2^{10}-2^{14}$	17
7	$+2^0 \ldots +2^4, -2^5, +2^6 \ldots +2^{10}$ $-2^{11}, -2^{23} \ldots -2^{25}, +2^{26} \ldots +2^{31}$	$-2^2+2^5+2^{10}$ $+2^{16}-2^{25}-2^{27}$	$-$	$-2^2+2^5+2^{10}$ $+2^{16}-2^{25}-2^{27}$	22
8	$+2^0+2^{15}-2^{16}+2^{17}$ $+2^{18}+2^{19}-2^{20}-2^{23}$	$2^6+2^8+2^{10}$ $+2^{16}-2^{24}+2^{31}$	$-$	$2^8+2^{10}+2^{16}$ $-2^{24}+2^{31}$	7
9	$-2^0+2^1+2^6+2^7-2^8-2^{31}$	$2^0+2^6-2^{20}$ $-2^{23}+2^{26}+2^{31}$	$-$	$2^0-2^{20}+2^{26}$	12
10	$-2^{12}+2^{13}+2^{31}$	$2^0+2^6+2^{13}-2^{23}$	$-$	$2^{13}-2^{27}$	17
11	$+2^{30}+2^{31}$	-2^0-2^8	2^{15}	$-2^8-2^{17}-2^{23}$	22
12	$+2^7-2^8, +2^{13} \ldots +2^{18}, -2^{19}+2^{31}$	$2^7+2^{17}+2^{31}$	$-$	$2^0+2^6+2^{17}$	7
13	$-2^{24}+2^{25}+2^{31}$	$-2^{13}+2^{31}$	$-$	-2^{12}	12
14	$+2^{31}$	$2^{18}+2^{31}$	2^{31}	$2^{18}-2^{30}$	17
15	$+2^3-2^{15}+2^{31}$	$2^{25}+2^{31}$	$-$	$-2^7-2^{13}+2^{25}$	22
16	$-2^{29}+2^{31}$	2^{31}	$-$	2^{24}	5
17	$+2^{31}$	2^{31}	$-$	$-$	9
18	$+2^{31}$	2^{31}	2^{15}	2^3	14
19	$+2^{17}+2^{31}$	2^{31}	$-$	-2^{29}	20
20	$+2^{31}$	2^{31}	$-$	$-$	5
21	$+2^{31}$	2^{31}	$-$	$-$	9
22	$+2^{31}$	2^{31}	$-$	2^{17}	14
23	$-$	$-$	2^{31}	$-$	20
24	$-$	2^{31}	$-$	$-$	5
25	$-$	$-$	2^{31}	$-$	9
$26-33$	$-$	$-$	$-$	$-$	$\cdot$
34	$-$	$-$	2^{15}	2^{15}	16
35	$\delta Q_{35} = 2^{31}$	2^{31}	2^{31}	$-$	23
36	$\delta Q_{36} = 2^{31}$	$-$	$-$	$-$	4
37	$\delta Q_{37} = 2^{31}$	2^{31}	2^{31}	$-$	11
$38-49$	$\delta Q_t = 2^{31}$	2^{31}	$-$	$-$	$\cdot$
50	$\delta Q_{50} = 2^{31}$	$-$	2^{31}	$-$	15
$51-59$	$\delta Q_t = 2^{31}$	2^{31}	$-$	$-$	$\cdot$
60	$\delta Q_{60} = 2^{31}$	$-$	2^{31}	$-$	6
61	$\delta Q_{61} = 2^{31}$	2^{31}	2^{15}	2^{15}	10
62	$\delta Q_{62} = 2^{31}+2^{25}$	2^{31}	$-$	$-$	15
63	$\delta Q_{63} = 2^{31}+2^{25}$	2^{31}	$-$	$-$	21
64	$\delta Q_{64} = 2^{31}+2^{25}$	$\times$	$\times$	$\times$	$\times$

$\delta m_4 = -2^{31}, \quad \delta m_{11} = -2^{15}, \quad \delta m_{14} = -2^{31}, \quad \delta m_i = 0, i \notin \{4, 11, 14\}$

t	ΔQ_t (BSDR of δQ_t)	δF_t	δw_t	δT_t	RC_t
-3	$+2^{31}$	$\times$	$\times$	$\times$	$\times$
-2	$+2^{25}+2^{31}$	$\times$	$\times$	$\times$	$\times$
-1	$-2^{25}+2^{26}+2^{31}$	$\times$	$\times$	$\times$	$\times$
0	$+2^{25}+2^{31}$	2^{31}	$-$	$-$	7
1	$+2^{25}+2^{31}$	2^{31}	$-$	2^{25}	12
2	$+2^5+2^{25}+2^{31}$	2^{25}	$-$	$2^{31}+2^{26}$	17
3	$-2^5-2^6+2^7-2^{11}+2^{12}$ $-2^{16} \ldots -2^{20}, +2^{21}$ $-2^{25} \ldots -2^{29}, +2^{30}+2^{31}$	$-2^{11}-2^{21}+2^{25}$ $-2^{27}+2^{31}$	$-$	$-2^{11}-2^{21}-2^{26}$	22
4	$+2^1+2^2+2^3-2^4+2^5$ $-2^{25}+2^{26}+2^{31}$	$2^1-2^3-2^{18}$ $+2^{26}+2^{30}$	2^{31}	$2^1+2^2-2^{18}$ $+2^{25}+2^{26}+2^{30}$	7
5	$+2^0-2^6+2^7+2^8-2^9$ $-2^{10}-2^{11}+2^{12}+2^{31}$	$-2^4-2^5-2^8-2^{20}$ $-2^{25}-2^{26}+2^{28}+2^{30}$	$-$	$-2^4-2^8-2^{20}$ $-2^{26}+2^{28}-2^{30}$	12
6	$+2^{16}-2^{17}+2^{20}-2^{21}+2^{31}$	$2^3-2^5-2^{10}-2^{11}$ $-2^{16}-2^{21}-2^{25}$	$-$	$2^3-2^{10}-2^{21}-2^{31}$	17
7	$+2^6+2^7+2^8-2^9$ $+2^{27}-2^{28}+2^{31}$	$2^{16}-2^{27}+2^{31}$	$-$	$-2^1+2^5+2^{16}$ $+2^{25}-2^{27}$	22
8	$-2^{15}+2^{16}-2^{17}+2^{23}$ $+2^{24}+2^{25}-2^{26}+2^{31}$	$-2^6+2^{16}+2^{25}$	$-$	$2^0+2^8+2^9$ $+2^{16}+2^{25}-2^{31}$	7
9	$-2^0+2^1, -2^6 \ldots -2^8, +2^9+2^{31}$	$2^0+2^{16}-2^{26}+2^{31}$	$-$	$2^0-2^{20}-2^{26}$	12
10	$+2^{12}+2^{31}$	2^6+2^{31}	$-$	-2^{27}	17
11	$+2^{31}$	2^{31}	-2^{15}	$-2^{17}-2^{23}$	22
12	$-2^7, +2^{13} \ldots +2^{18}-2^{19}+2^{31}$	$2^{17}+2^{31}$	$-$	$2^0+2^6+2^{17}$	7
13	$-2^{24} \ldots -2^{29}, +2^{30}+2^{31}$	$-2^{13}+2^{31}$	$-$	-2^{12}	12
14	$+2^{31}$	$2^{18}+2^{30}$	2^{31}	$2^{18}+2^{30}$	17
15	$+2^3+2^{15}+2^{31}$	$-2^{25}+2^{31}$	$-$	$-2^7-2^{13}-2^{25}$	22
16	$-2^{29}+2^{31}$	2^{31}	$-$	2^{24}	5
17	$+2^{31}$	2^{31}	$-$	$-$	9
18	$+2^{31}$	2^{31}	-2^{15}	2^3	14
19	$+2^{17}+2^{31}$	2^{31}	$-$	-2^{29}	20
20	$+2^{31}$	2^{31}	$-$	$-$	5
21	$+2^{31}$	2^{31}	$-$	$-$	9
22	$+2^{31}$	2^{31}	$-$	2^{17}	14
23	$-$	$-$	2^{31}	$-$	20
24	$-$	2^{31}	$-$	$-$	5
25	$-$	$-$	2^{31}	$-$	9
$26-33$	$-$	$-$	$-$	$-$	$\cdot$
34	$-$	$-$	-2^{15}	-2^{15}	16
35	$\delta Q_{35} = 2^{31}$	2^{31}	2^{31}	$-$	23
36	$\delta Q_{36} = 2^{31}$	$-$	$-$	$-$	4
37	$\delta Q_{37} = 2^{31}$	2^{31}	2^{31}	$-$	11
$38-49$	$\delta Q_t = 2^{31}$	2^{31}	$-$	$-$	$\cdot$
50	$\delta Q_{50} = 2^{31}$	$-$	2^{31}	$-$	15
$51-59$	$\delta Q_t = 2^{31}$	2^{31}	$-$	$-$	$\cdot$
60	$\delta Q_{60} = 2^{31}$	$-$	2^{31}	$-$	6
61	$\delta Q_{61} = 2^{31}$	2^{31}	-2^{15}	-2^{15}	10
62	$\delta Q_{62} = 2^{31} - 2^{25}$	2^{31}	$-$	$-$	15
63	$\delta Q_{63} = 2^{31} - 2^{25}$	2^{31}	$-$	$-$	21
64	$\delta Q_{64} = 2^{31} - 2^{25}$	$\times$	$\times$	$\times$	$\times$

Exploiting Wang's path

Sufficient conditions

- Once a valid path is found (*Wang did it "by hand", relying only on intuition !*), we must build a pair of blocks that follows it

Symbol	State condition $Q_t[i]$
.	none
0	$Q_t[i] = 0$
1	$Q_t[i] = 1$
^	$Q_t[i] = Q_{t-1}[i]$
!	$Q_t[i] = \neg Q_{t-1}[i]$

Exploiting Wang's path

Sufficient conditions

- Once a valid path is found (*Wang did it "by hand", relying only on intuition !*), we must build a pair of blocks that follows it
- Sufficient set of bit conditions for the path to hold on a block derived from path

Symbol	State condition $Q_t[i]$
.	none
0	$Q_t[i] = 0$
1	$Q_t[i] = 1$
^	$Q_t[i] = Q_{t-1}[i]$
!	$Q_t[i] = \neg Q_{t-1}[i]$

Exploiting Wang's path

Sufficient conditions

- Once a valid path is found (*Wang did it "by hand", relying only on intuition !*), we must build a pair of blocks that follows it
- Sufficient set of bit conditions for the path to hold on a block derived from path
- Wang proposed a set of conditions derived by hand, she made mistakes

Symbol	State condition $Q_t[i]$
.	none
0	$Q_t[i] = 0$
1	$Q_t[i] = 1$
^	$Q_t[i] = Q_{t-1}[i]$
!	$Q_t[i] = \neg Q_{t-1}[i]$

SC algorithm

Automated sufficient conditions derivation

- Contruct sufficient conditions to control output of non-linear F_i function

Simplified algorithm

SC algorithm

Automated sufficient conditions derivation

- Contruct sufficient conditions to control output of non-linear F_i function
- Construct conditions to control carry length

Simplified algorithm

SC algorithm

Automated sufficient conditions derivation

- Contruct sufficient conditions to control output of non-linear F_i function
- Construct conditions to control carry length
- Rotations are still handled by hand, or by a SAT solver

Simplified algorithm

SC algorithm

Automated sufficient conditions derivation

- Contruct sufficient conditions to control output of non-linear F_i function
- Construct conditions to control carry length
- Rotations are still handled by hand, or by a SAT solver
- Differentials in the outermost rounds are examined first

Simplified algorithm

SC algorithm

Automated sufficient conditions derivation

- Contruct sufficient conditions to control output of non-linear F_i function
- Construct conditions to control carry length
- Rotations are still handled by hand, or by a SAT solver
- Differentials in the outermost rounds are examined first

Simplified algorithm

- Find candidate ΔF_i that satisfies input differential with highest probability to maintain the path

SC algorithm

Automated sufficient conditions derivation

- Contruct sufficient conditions to control output of non-linear F_i function
- Construct conditions to control carry length
- Rotations are still handled by hand, or by a SAT solver
- Differentials in the outermost rounds are examined first

Simplified algorithm

- Find candidate ΔF_i that satisfies input differential with highest probability to maintain the path
- Set “chaining” differentials to prevent carries

SC algorithm

Automated sufficient conditions derivation

- Contruct sufficient conditions to control output of non-linear F_i function
- Construct conditions to control carry length
- Rotations are still handled by hand, or by a SAT solver
- Differentials in the outermost rounds are examined first

Simplified algorithm

- Find candidate ΔF_i that satisfies input differential with highest probability to maintain the path
- Set “chaining” differentials to prevent carries
- Derive conditions to control ΔF_i from first to last bit. If a contradiction arises, backtrack

An exemple

Sample path

- $F(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
- We want $\Delta Q_{i-1} = \Delta Q_{i-2} = 0$, $\Delta Q_{i-3} = 2^5$, $\Delta F(Q_{i-1}, Q_{i-2}, Q_{i-3}) = 2^7$
- State equation :$\Delta F(Q_{i-1}, Q_{i-2}, Q_{i-3})$ is $R_j(\Delta Q_i - \Delta Q_{i-1}) - \Delta M_i - \Delta K_i - \Delta Q_{i-4}$
- $\Delta F(Q_{i-1}, Q_{i-2}, Q_{i-3}) = 2^7$ is impossible (no differential on 8th bit)
- So we add a bit differential in position 8 by expanding carry in $\Delta Q_{i-3} = 2^5$
- We add conditions $Q_{i-3}[1_6, 1_7, 0_8]$
- Now we have differentials in bit 6 and 7 that ΔF hasn't. Fortunately, F doesn't have differentials if bits 6 and 7 are set.
- Furthermore, $Q_{i-3}[0_8]$ yields $\Delta F(Q_{i-1}, Q_{i-2}, Q_{i-3}) = 2^7$, so we have our conditions.

Building the collision

Chosen prefix

- Can't eliminate any IVH_k

Building the collision

Chosen prefix

- Can't eliminate any IVH_k
- Can eliminate $\Delta IVH_k = (0, \Delta B, \Delta B, \Delta B)$

Antoine Delignat-Lavaud

Building the collision

Chosen prefix

- Can't eliminate any IVH_k
- Can eliminate $\Delta IVH_k = (0, \Delta B, \Delta B, \Delta B)$
- Birthday attach on previous block can lead to *IHV* of the correct form

Building the collision

Chosen prefix

- Can't eliminate any IVH_k
- Can eliminate $\Delta IVH_k = (0, \Delta B, \Delta B, \Delta B)$
- Birthday attach on previous block can lead to IHV of the correct form
- Few changes in the blocks before the collision block allow collisions on meaningful data

Building the collision

Chosen prefix

- Can't eliminate any IVH_k
- Can eliminate $\Delta IVH_k = (0, \Delta B, \Delta B, \Delta B)$
- Birthday attach on previous block can lead to *IHV* of the correct form
- Few changes in the blocks before the collision block allow collisions on meaningful data
- Only a few bits are changed in the two 512-bit collision blocks

Building the collision

Collision algorithm

- Birthday attack previous blocks at the least suspicious place have a good ΔIHV

Building the collision

Collision algorithm

- Birthday attack previous blocks at the least suspicious place have a good ΔIHV
- Chose arbitrary block pair that meets all suficient conditions for the first round.

Building the collision

Collision algorithm

- Birthday attack previous blocks at the least suspicious place have a good ΔIHV
- Chose arbitrary block pair that meets all suficient conditions for the first round.
- Apply compression function while sufficient conditions are met.

Building the collision

Collision algorithm

- Birthday attack previous blocks at the least suspicious place have a good ΔIHV
- Chose arbitrary block pair that meets all suficient conditions for the first round.
- Apply compression function while sufficient conditions are met.
- If a condition is not met in a relatively deep state of the function, try to patch the block you're building using message modification (precomputed modification that do not broke previous conditions for this path) or tunneling (backtrack to the first neutral bit and pray)

An impressive breach

Random-looking versus constructed

- Back in 1991, MD5 was designed using intuition rather than theory
- Using simple techniques and intuition, it was possible to find weak diffusion paths and exploit them
- Rivest has learned his lesson, SHA-3 candidate MD6 is proven secure against differential attacks
- Sequential approach replaced by parallel, tree-based scheme

Serious security implications

- Integrity bypassed in a minute
- Digital signature no longer to be trusted
- Fortunately complex enough to discourage real world attacks

References

M. Stevens
On collisions for MD5.
Eindhoven University, 2007

Xiaoyun Wang, Hongbo Yu
How to Break MD5 and Other Hash Functions.
Shandong University, 2005