Collisions in MD5
... and how to use them for fun and profit.

Mar. 10, 2009
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 $\Sigma, \Omega$ be two finite alphabets and $n$ a positive integer. A hash function $f$ is a map:

$$f : \Sigma^* \rightarrow \Omega^n$$
What is a hash function?

**Hash function**

Let $\Sigma, \Omega$ 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:

- Easy to compute for any input
- Preimage resistant (given $s \in \Omega^n$, it is hard to find $\omega \in \Sigma^*$ such that $f(\omega) = s$)
- Second preimage resistant (given $\omega_1 \in \Sigma^*$, it is hard to find $\omega_2 \neq \omega_1$ such that $f(\omega_1) = f(\omega_2)$)
- Collision resistant (it is hard to find $\omega_1, \omega_2 \in \Sigma^*$ such that $f(\omega_1) = f(\omega_2)$)
What is a hash function?

**Hash function**

Let $\Sigma, \Omega$ be two finite alphabets and $n$ a positive integer. A hash function $f$ is a map:

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

**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 $\Sigma, \Omega$ be two finite alphabets and $n$ a positive integer. A hash function $f$ is a map:

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

**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 $\Sigma, \Omega$ be two finite alphabets and $n$ a positive integer. A hash function $f$ is a map:

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

**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 $\Sigma, \Omega$ 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?

• Birthday attack: $O(\Omega n^2)$
• 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.
### What does “hard” mean?

- Birthday attack: $\mathcal{O}(\sqrt{|\Omega|})$
What does “hard” mean?

- Birthday attack: $O(|\Omega|^n)$
- Brute force can be effective! (up to 1 billion hashes per second on a desktop PC)
What does “hard” mean?

- Birthday attack: $O(|\Omega|^n)$
- 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.

---

**BarsWF MD5 bruteforcer v0.8**
by Svarychevski Michail

- GPU0: 266.63 MHash/sec
- CPU0: 49.82 MHash/sec
- CPU1: 49.21 MHash/sec
- CPU2: 49.42 MHash/sec
- GPU*: 266.63 MHash/sec
- CPU*: 198.15 MHash/sec

Key: wIeODw  Avg.Total: 458.82 MHash/sec
Hash:a9a90f301644f9600b99b2db23f23511
Progress: 23.89 % ETC  0 days 0 hours 1 min 34 sec

---

**Key is: w9Ec03ru**
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 $\texttt{\$1\$-crypt}$ 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 → fixed length
The Merkle-Damgård construction

Goals

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

Outline

Hash functions and their uses
  - What is a hash function?
  - The Merkle-Damgård construction
  - Message-Digest algorithm 5 (MD5)

Differential cryptanalysis of MD5
  - Wang’s differential path
  - Deriving a sufficient conditions set
  - Building the collision

Conclusion
The Merkle-Damgård construction

Goals

- Compress input: variable length → fixed length
- Balance strength and simplicity
- Strong “avalanche” effect
The Merkle-Damgård construction

Goals

- Compress input: variable length → fixed length
- Balance strength and simplicity
- Strong “avalanche” effect
The Merkle-Damgård construction

**Construction**

- $f : \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}^n$ 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

- $f : \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}^n$ 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

- \( f : \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}^n \) 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
Collisions in MD5
Antoine Delignat-Lavaud

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

Algorithm parameters

Non-linear function

\[
F_1(X, Y, Z) = (X \land Y) \lor (\neg X \land Z)
\]
\[
F_2(X, Y, Z) = (X \land Z) \lor (Y \land \neg Z)
\]
\[
F_3(X, Y, Z) = X \oplus Y \oplus Z
\]
\[
F_4(X, Y, Z) = Y \oplus (X \lor \neg Z)
\]

\[K_{i,k} \text{ “nothing up my sleeve” constants}\]

\[
K_{i,k} = \lfloor 2^{32} \mid \sin(4 \ast (k - 1) + i) \rfloor
\]

Initialization vector

\[
A_0 = 0x67452301
\]
\[
B_0 = 0xEFCDAB89
\]
\[
C_0 = 0x98BADCFE
\]
\[
D_0 = 0x10325476
\]
Differential cryptanalysis of MD5

Differential Cryptanalysis

- Family 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

Wang’s differential path
- Deriving a sufficient conditions set
- Building the collision

Conclusion
Differential cryptanalysis of MD5

Differential Cryptanalysis

- Family 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 \(\Delta_i\) 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

- Family 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 \(\Delta_i\) 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
Differential cryptanalysis of MD5

Differential Cryptanalysis

- Family 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 \(\Delta_i\) 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 \(IHV_k\):

\[
\ldots \xrightarrow{M_k} IVH_k \xrightarrow{M_0} IVH_{k+1} \xrightarrow{M_1} IHV_{k+1} \ldots
\]

\[
\ldots \xrightarrow{M_k} IVH_k \xrightarrow{M'_0} IVH'_{k+1} \xrightarrow{M'_1} IHV'_{k+1} \ldots
\]
How to differentiate smartly

Differential notations

- $M'$ denotes the collision dual message of $M$
- $\Delta X = X' - X$ where $-\cdot$ 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
Collisions in MD5

Antoine Delignat-Lavaud

Outline

Hash functions and their uses
What is a hash function?
The Merkle-Damgård construction
Message-Digest algorithm 5 (MD5)

Differential cryptanalysis of MD5
Wang’s differential path
Deriving a sufficient conditions set
Building the collision

Conclusion
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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>State condition $Q_t[i]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>none</td>
</tr>
<tr>
<td>0</td>
<td>$Q_t[i] = 0$</td>
</tr>
<tr>
<td>1</td>
<td>$Q_t[i] = 1$</td>
</tr>
<tr>
<td>^</td>
<td>$Q_t[i] = Q_{t-1}[i]$</td>
</tr>
<tr>
<td>!</td>
<td>$Q_t[i] = \neg Q_{t-1}[i]$</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>State condition $Q_t[i]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>none</td>
</tr>
<tr>
<td>0</td>
<td>$Q_t[i] = 0$</td>
</tr>
<tr>
<td>1</td>
<td>$Q_t[i] = 1$</td>
</tr>
<tr>
<td>^</td>
<td>$Q_t[i] = Q_{t-1}[i]$</td>
</tr>
<tr>
<td>!</td>
<td>$Q_t[i] = \neg Q_{t-1}[i]$</td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>State condition $Q_t[i]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>none</td>
</tr>
<tr>
<td>0</td>
<td>$Q_t[i] = 0$</td>
</tr>
<tr>
<td>1</td>
<td>$Q_t[i] = 1$</td>
</tr>
<tr>
<td>^</td>
<td>$Q_t[i] = Q_{t-1}[i]$</td>
</tr>
<tr>
<td>!</td>
<td>$Q_t[i] = \neg Q_{t-1}[i]$</td>
</tr>
</tbody>
</table>
Automated sufficient conditions derivation

- Contract sufficient conditions to control output of non-linear $F_i$ function

Simplified algorithm
SC algorithm

Automated sufficient conditions derivation

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

Simplified algorithm
## Automated sufficient conditions derivation

- Contract 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

- Find candidate $\Delta F_i$ that satisfies input differential with highest probability to maintain the path
- Set “chaining” differentials to prevent carries
- Derive conditions to control $\Delta F_i$ from first to last bit. If a contradiction arises, backtrack
Collisions in MD5

Antoine Delignat-Lavaud

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

Automated sufficient conditions derivation

- Contract 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
## SC algorithm

### Automated sufficient conditions derivation

- Contract 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 $\Delta F_i$ that satisfies input differential with highest probability to maintain the path
SC algorithm

Automated sufficient conditions derivation

- Contract 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 $\Delta F_i$ that satisfies input differential with highest probability to maintain the path
- Set “chaining” differentials to prevent carries
Automated sufficient conditions derivation

- Contract 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 $\Delta F_i$ that satisfies input differential with highest probability to maintain the path
- Set “chaining” differentials to prevent carries
- Derive conditions to control $\Delta F_i$ from first to last bit. If a contradiction arises, backtrack
An exemple

Sample path

- \( F(X, Y, Z) = (X \land Y) \lor (\neg X \land Z) \)
- 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 \( \Delta F \) 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 $\Delta IVH_k = (0, \Delta B, \Delta B, \Delta B)$
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

<table>
<thead>
<tr>
<th>Chosen prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Can’t eliminate any $IVH_k$</td>
</tr>
<tr>
<td>• Can eliminate $\Delta IVH_k = (0, \Delta B, \Delta B, \Delta B)$</td>
</tr>
<tr>
<td>• Birthday attach on previous block can lead to $IHV$ of the correct form</td>
</tr>
<tr>
<td>• Few changes in the blocks before the collision block allow collisions on meaningful data</td>
</tr>
</tbody>
</table>
Building the collision

<table>
<thead>
<tr>
<th>Chosen prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Can’t eliminate any $ IVH_k $</td>
</tr>
<tr>
<td>• Can eliminate $ \Delta IVH_k = (0, \Delta B, \Delta B, \Delta B) $</td>
</tr>
<tr>
<td>• Birthday attach on previous block can lead to $ IHV $ of the correct form</td>
</tr>
<tr>
<td>• Few changes in the blocks before the collision block allow collisions on meaningful data</td>
</tr>
<tr>
<td>• Only a few bits are changed in the two 512-bit collision blocks</td>
</tr>
</tbody>
</table>
Building the collision

Collision algorithm

- Birthday attack previous blocks at the least suspicious place have a good $\Delta IHV$
Building the collision

Collision algorithm

- Birthday attack previous blocks at the least suspicious place have a good $\Delta IHV$
- Chose arbitrary block pair that meets all sufficient conditions for the first round.
Building the collision

Collision algorithm

- Birthday attack previous blocks at the least suspicious place have a good $\Delta IHV$
- Chose arbitrary block pair that meets all sufficient 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 $\Delta IHV$
- Chose arbitrary block pair that meets all sufficient 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