(Second) Preimage Attacks on (Reduced) SHA-0/1

Christophe De Cannière and Christian Rechberger

ENS, Chaire France Telecom
Katholieke Universiteit Leuven
Graz University of Technology

January 8, 2008
Outline

1. Background
   - SHA-0/1
   - Collision Attacks

2. (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity

3. Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks

4. Example and Final Remarks
Outline

1. Background
   - SHA-0/1
   - Collision Attacks

2. (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity

3. Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks

4. Example and Final Remarks
Hash Function

- **Input:** message $m$ of arbitrary length
- **Output:** hash value $h(m)$ of fixed length $n$
- Fixed, publicly known function (no secret parameters)
- Sufficiently efficient
SHA-0/1 Hash Function

- Iterative hash function.
- 512-bit message blocks $m_j$.
- 160-bit chaining variable $H_j$.
- 160-bit hash value $h(m)$.
- Padding, MD-strengthening.
SHA-0/1 Hash Function

- Iterative hash function.
- 512-bit message blocks $m_j$.
- 160-bit chaining variable $H_j$.
- 160-bit hash value $h(m)$.
- Padding, MD-strengthening.
SHA-0/1 Compression Function

\[ H_{j-1} \]

\[ A_0 \quad B_0 \quad C_0 \quad D_0 \quad E_0 \]

\[ m_j \]

\[ W_0 \]

\[ \ldots \]

\[ W_{15} \]

five 32-bit state variables
SHA-0/1 Compression Function

\[ H_{j-1} \]

\[ W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \ll 0/1 \]
SHA-0/1 Compression Function

\[ H_{j-1} \]

\[
\begin{array}{cccccc}
A_0 & B_0 & C_0 & D_0 & E_0 \\
A_1 & B_1 & C_1 & D_1 & E_1 \\
\end{array}
\]

step transformation

\[ m_j \]

\[
\begin{array}{cccccc}
W_0 \\
\vdots \\
W_{15} \\
W_{16} \\
\vdots \\
W_{79} \\
\end{array}
\]
SHA-0/1 Compression Function

\[
H_{j-1}
\]

\[
\begin{array}{cccccc}
A_0 & B_0 & C_0 & D_0 & E_0 \\
A_1 & B_1 & C_1 & D_1 & E_1 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
A_{79} & B_{79} & C_{79} & D_{79} & E_{79} \\
A_{80} & B_{80} & C_{80} & D_{80} & E_{80} \\
\end{array}
\]

\[
\begin{array}{cccccc}
W_0 & \cdots & W_{15} \\
W_{16} & \cdots & W_{79} \\
\end{array}
\]

\[
m_j
\]
SHA-0/1 Compression Function

\[
\begin{array}{cccccc}
A_0 & B_0 & C_0 & D_0 & E_0 \\
A_1 & B_1 & C_1 & D_1 & E_1 \\
A_{79} & B_{79} & C_{79} & D_{79} & E_{79} \\
A_{80} & B_{80} & C_{80} & D_{80} & E_{80} \\
\end{array}
\]
SHA-0/1 Compression Function

\[ H_{j-1} \]

\[
\begin{array}{cccccc}
A_0 & B_0 & C_0 & D_0 & E_0 \\
A_1 & B_1 & C_1 & D_1 & E_1 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
A_{79} & B_{79} & C_{79} & D_{79} & E_{79} \\
A_{80} & B_{80} & C_{80} & D_{80} & E_{80} \\
\end{array}
\]

\[ m_j \]

\[
\begin{array}{cccccc}
W_0 \\
\cdots \\
W_{15} \\
W_{16} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
W_{79} \\
\end{array}
\]
SHA Step Transformation
SHA Step Transformation

\[
\begin{align*}
A_i & \quad B_i & \quad C_i & \quad D_i & \quad E_i \\
5 & \quad f & \quad K_i & \quad W_i \\
A_{i+1} & \quad B_{i+1} & \quad C_{i+1} & \quad D_{i+1} & \quad E_{i+1}
\end{align*}
\]
F-Function

- Bitwise boolean function \( f \) changes every 20 steps:

\[
\begin{align*}
    i = 0, \ldots, 19: & \quad f_{\text{IF}} = (B \land C) \oplus (\neg B \land D) \\
    i = 20, \ldots, 39: & \quad f_{\text{XOR}} = B \oplus C \oplus D \\
    i = 40, \ldots, 59: & \quad f_{\text{MAJ}} = (B \land C) \oplus (B \land D) \oplus (C \land D) \\
    i = 60, \ldots, 79: & \quad f_{\text{XOR}} = B \oplus C \oplus D
\end{align*}
\]
SHA Step Function (Recursive in $A_i$)

- All state variables can be expressed as a function of $A_i$
SHA Step Function (Recursive in $A_i$)

- All state variables can be expressed as a function of $A_i$
SHA Compression Function (Recursive in $A_i$)

From now on, we only consider state variables $A_i$. 
Outline

1 Background
   ■ SHA-0/1
   ■ Collision Attacks

2 (Second) Preimage Attack on Reduced SHA-0
   ■ General Ideas
   ■ Basic Technique
   ■ Complexity

3 Improvements
   ■ Getting Rid of Those Carries
   ■ Using More Blocks
   ■ Using Even More Blocks

4 Example and Final Remarks
Collision Search Attack

- **Goal:**
  
  Find two different messages with the same hash value
Collision Search Attack

- **Goal:**
  Find two different messages with the same hash value

\[ h(\text{message 1}) = 92B8CD94 \neq h(\text{message 2}) = 92B8CD94 \]
Differential Cryptanalysis: Not All Bits Are Equal

- Limit search space to pairs of messages whose bits are related throughout the hash computation.
- Depending on their position, bits of $A_i$ and $W_i$ depend on $m_j$ in a more or less complex way.

![Diagram showing the relationship between $A$, $m_j$, and $W$ with easy and hard sections.]
Bottom Part of Characteristic

- Requirement of (near-)collision imposes restrictions in last 5 steps of the “hard” part.
Requirement of (near-)collision imposes restrictions in last 5 steps of the “hard” part.

→ **Stage 1**: impose differences in “easier” parts, which have the highest possible probability to propagate to desired difference in “hard” part.
Bottom Part of Characteristic

- Requirement of (near-)collision imposes restrictions in last 5 steps of the “hard” part.
  - Stage 1: impose differences in “easier” parts, which have the highest possible probability to propagate to desired difference in “hard” part.

- Nice sparse char. because of:
  - limited bit-interaction
  - uniformity of linearized SHA-1
  - two-block collision
Top Part of Characteristic

- Difference in second part of $W$ determines difference in first part of $W$. 

```
\begin{array}{c}
\text{-4} \\
\text{16} \\
\text{30} \\
\text{80}
\end{array}
```
Top Part of Characteristic

- Difference in second part of $W$ determines difference in first part of $W$.

→ **Stage 2**: find generalized characteristic which connects the difference in $W$ to the desired difference in $A$. 
Top Part of Characteristic

- Difference in second part of $W$ determines difference in first part of $W$.
  - **Stage 2**: find generalized characteristic which connects the difference in $W$ to the desired difference in $A$.
- Because of tight restrictions, characteristic needs to exploit nonlinearity.
  - Not so easy to find. [DCR06]
Finding a Message Pair

- **Stage 3:** construct message pair following the characteristic for first 20+ steps.
Finding a Message Pair

- **Stage 3**: construct message pair following the characteristic for first 20+ steps.
- **Stage 4**: if conditions in next few steps are not fulfilled, try to fix them.
  → Boomerangs, clusters, ... [JP07, MRR07].
Finding a Message Pair

- **Stage 3**: construct message pair following the characteristic for first 20+ steps.
- **Stage 4**: if conditions in next few steps are not fulfilled, try to fix them.
  → Boomerangs, clusters, ... [JP07, MRR07].
- **Stage 5**: check if characteristic is followed in the last part. If not, try again with different pair.
## Achievements

- **2004:**
  - 80-step SHA-0: collision found [Jou04]
  - 53-step SHA-1: better than birthday [OR04], [BC04]

- **2005:**
  - 58-step SHA-1: collision found [WYY05]
  - 80-step SHA-1: first $2^{69}$, then $2^{63}$ hash evaluations [WYY05]

- **2006:**
  - 64-step SHA-1: collision found [DCR06]

- **2007:**
  - 70-step SHA-1: collision found [DCRM07]
  - 80-step SHA-1: $\approx 2^{60}$ hash evaluations [MRR07]
Achievements

- **2004:**
  - 80-step SHA-0: collision found [Jou04]
  - 53-step SHA-1: better than birthday [OR04], [BC04]

- **2005:**
  - 58-step SHA-1: collision found [WYY05]
  - 80-step SHA-1: first $2^{69}$, then $2^{63}$ hash evaluations [WYY05]

- **2006:**
  - 64-step SHA-1: collision found [DCR06]

- **2007:**
  - 70-step SHA-1: collision found [DCRM07]
  - 80-step SHA-1: $\approx 2^{60}$ hash evaluations [MRR07]

**Question:** Can we somehow use this for (2nd) preimage attacks?
Outline

1. Background
   - SHA-0/1
   - Collision Attacks

2. (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity

3. Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks

4. Example and Final Remarks
Second Preimage Attack

**Goal:**
Given a message, find a different message which produces the same hash value

\[
\text{h} = 92B8CD94 = 92B8CD94
\]
Second Preimage Attack

- **Goal:**
  Given a message, find a different message which produces the same hash value

\[ h_{92B8CD94} \neq h_{?} \]

\[ 92B8CD94 = 92B8CD94 \]
How to generate second preimages?
How to generate second preimages?

1. **Idea 1:**
   Try to apply characteristic from collision search attack to given message [WZW05].
How to generate second preimages?

1. **Idea 1:**
   Try to apply characteristic from collision search attack to given message [WZW05].
   -> **Problem:** low success rate
How to generate second preimages?

1. **Idea 1:**
   Try to apply characteristic from collision search attack to given message [WZW05].

   → **Problem:** low success rate

**Collision Attack** 2nd Preimage Attack
How to generate second preimages?

1. **Idea 1:** Try to apply characteristic from collision search attack to given message [WZW05].

   → **Problem:** low success rate

**Collision Attack**

1. Apply special difference to special message $m$.  

**2nd Preimage Attack**
How to generate second preimages?

1. **Idea 1:**
   Try to apply characteristic from collision search attack to given message [WZW05].

   → **Problem:** low success rate

**Collision Attack**

1. Apply special difference to special message $m$.

2. Check for collision.
How to generate second preimages?

1. **Idea 1:**
   Try to apply characteristic from collision search attack to given message [WZW05].
   
   → **Problem:** low success rate

**Collision Attack**

1. Apply special difference to special message $m$.

2. Check for collision.

3. If not, try with different special message.
How to generate second preimages?

1 Idea 1: Try to apply characteristic from collision search attack to given message [WZW05].

→ Problem: low success rate

Collision Attack

1 Apply special difference to special message $m$.
2 Check for collision.
3 If not, try with different special message.

2nd Preimage Attack

1 Apply special difference to given message $m$. 
How to generate second preimages?

1. **Idea 1:**
   Try to apply characteristic from collision search attack to given message [WZW05].
   → **Problem:** low success rate

**Collision Attack**

1. Apply special difference to special message $m$.
2. Check for collision.
3. If not, try with different special message.

**2nd Preimage Attack**

1. Apply special difference to given message $m$.
2. Check for collision.
How to generate second preimages?

1. **Idea 1:**
   Try to apply characteristic from collision search attack to given message [WZW05].
   
   → **Problem:** low success rate

**Collision Attack**

1. Apply special difference to special message $m$.
2. Check for collision.
3. If not, try with different special message.

**2nd Preimage Attack**

1. Apply special difference to given message $m$.
2. Check for collision.
3. If not, too bad...
How to generate second preimages?

1. **Idea 1:**
   Try to apply characteristic from collision search attack to given message [WZW05].
   → **Problem:** low success rate

**Collision Attack**

1. Apply special difference to special message $m$.
2. Check for collision.
3. If not, try with different special message.

**2nd Preimage Attack**

1. Apply special difference to given message $m$.
2. Check for collision.
3. If not, try with different special difference?
How to generate second preimages?

2 Idea 2: Try to use differential characteristics to correct parts of the hash value of a (chosen) message.
How to generate second preimages?

**Idea 2:**
Try to use differential characteristics to correct parts of the hash value of a (chosen) message.

**Preimage Attack**
1. Compute hash value for special message $m$. 
How to generate second preimages?

2 Idea 2: Try to use differential characteristics to correct parts of the hash value of a (chosen) message.

Preimage Attack

1. Compute hash value for special message $m$.
2. Try to correct (parts of it) by applying special differences.
How to generate second preimages?

2 Idea 2:  
Try to use differential characteristics to correct parts of the hash value of a (chosen) message.

Preimage Attack

1 Compute hash value for special message $m$.
2 Try to correct (parts of it) by applying special differences.
3 If not successful, try with different special message.
How to generate second preimages?

Idea 2:
Try to use differential characteristics to correct parts of the hash value of a (chosen) message.

Preimage Attack

1. Compute hash value for special message \( m \).
2. Try to correct (parts of it) by applying special differences.
3. If not successful, try with different special message.

→ Seems to work quite well if one can find many highly probable differential paths for the same special message [Leu08], [Rec08].
How to generate second preimages?

3 **Idea 3:** Turn the problem around.
How to generate second preimages?

3 Idea 3: Turn the problem around.

Instead of trying to find a message which produces the correct hash value after being expanded and fed through several iterations of the state update transformation;

→ Start from state variables which produce the correct hash value, and try to modify them such that the expanded words satisfy the linear recursion.
How to generate second preimages?

3 Idea 3: Turn the problem around.

Instead of trying to find a message which produces the correct hash value after being expanded and fed through several iterations of the state update transformation;

→ Start from state variables which produce the correct hash value, and try to modify them such that the expanded words satisfy the linear recursion.

Why?
Flipping a Bit in the Message

<table>
<thead>
<tr>
<th>$i$</th>
<th>$A_i$</th>
<th>$W_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>00001111010010111100001111100001</td>
<td>11111000010100101110110101011</td>
</tr>
<tr>
<td>-3</td>
<td>01000000110010010101000111011000</td>
<td>0101010110100110110110111111110001</td>
</tr>
<tr>
<td>-2</td>
<td>01100010111010110110110011111111010</td>
<td>1011110111110100111011111111111001</td>
</tr>
<tr>
<td>-1</td>
<td>11101111110011010110110001001</td>
<td>001110000100010010011001100011001</td>
</tr>
<tr>
<td>0</td>
<td>01100111010001010001001100000001</td>
<td>01110010011010110011010000101001</td>
</tr>
<tr>
<td>1</td>
<td>100110000000110111101000000010</td>
<td>10101011010011011011011011111100</td>
</tr>
<tr>
<td>2</td>
<td>1011101001101100010101001010010001</td>
<td>1011110111110100111011111111111001</td>
</tr>
<tr>
<td>3</td>
<td>11010100110111000100110101010110111</td>
<td>00110001000100100110010110011000110</td>
</tr>
<tr>
<td>4</td>
<td>10010001001111101000101100011111</td>
<td>001010011001101111111111010100001</td>
</tr>
<tr>
<td>5</td>
<td>010101110100000101110011110000001</td>
<td>1011110111111011010011111011111001</td>
</tr>
<tr>
<td>6</td>
<td>10100111000001001101101000011100</td>
<td>001001100001001101100110000101010</td>
</tr>
<tr>
<td>7</td>
<td>10110001111100010011110110000000</td>
<td>11010011100110100110000111100011001</td>
</tr>
<tr>
<td>8</td>
<td>1010011011101010101010101010100010</td>
<td>10001111111011100000101100110110100</td>
</tr>
<tr>
<td>9</td>
<td>110100011100001110111010111010000</td>
<td>01111111010100100110110111001001</td>
</tr>
<tr>
<td>10</td>
<td>010101011101101101100111101010100</td>
<td>111101010100001011011000010101011001</td>
</tr>
<tr>
<td>11</td>
<td>1110011111111101101100011011011101</td>
<td>00100110111010101101100011001100111</td>
</tr>
<tr>
<td>12</td>
<td>011010100001001001010100011101011011001100010</td>
<td>10001111111011100000101100110110100</td>
</tr>
<tr>
<td>13</td>
<td>11101010011000001000010011001010111</td>
<td>011111111111110111011010000110110</td>
</tr>
<tr>
<td>14</td>
<td>11011000111000110110101110110111011</td>
<td>101000001010001001111111101100101</td>
</tr>
<tr>
<td>15</td>
<td>010010001010010011001100010000011</td>
<td>00011010111011111111100000011001</td>
</tr>
</tbody>
</table>

Flipping a Bit in the Message

<table>
<thead>
<tr>
<th>$i$</th>
<th>$A_i$</th>
<th>$W_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>00001111010010111000011111000011</td>
<td>111110000101001010110110101011</td>
</tr>
<tr>
<td>-3</td>
<td>01000000110010010101000111011000</td>
<td>01010101101001110110111110001</td>
</tr>
<tr>
<td>-2</td>
<td>011000101110101101110011111111010</td>
<td>1011110111101001111010010001</td>
</tr>
<tr>
<td>-1</td>
<td>11101111110110110101110001001</td>
<td>00110000010001001010011001010</td>
</tr>
<tr>
<td>0</td>
<td>01100111010001010010001100000001</td>
<td>111110000101001010110110101011</td>
</tr>
<tr>
<td>1</td>
<td>10011000000001101110100000001010</td>
<td>01010101101001110110111110001</td>
</tr>
<tr>
<td>2</td>
<td>10111101001101100101001010010001</td>
<td>1011110111101001111010010001</td>
</tr>
<tr>
<td>3</td>
<td>11101001110111001001110100111010</td>
<td>00110000010001001010011001010</td>
</tr>
<tr>
<td>4</td>
<td>10010001001111010001010000111111</td>
<td>001010011001101111001010000001</td>
</tr>
<tr>
<td>5</td>
<td>01011011010000010101001101001000</td>
<td>10111110111010110111100110001</td>
</tr>
<tr>
<td>6</td>
<td>101001100000000110101100110110010</td>
<td>001000110001001101011001010</td>
</tr>
<tr>
<td>7</td>
<td>101100010111000000011110011011</td>
<td>110010111100110011101010010</td>
</tr>
<tr>
<td>8</td>
<td>10010110110101101011100110011110</td>
<td>100011111110000010110011110</td>
</tr>
<tr>
<td>9</td>
<td>0101000001000001110110100001001</td>
<td>0111111101001001101001101010</td>
</tr>
<tr>
<td>10</td>
<td>101001011001001101000010111111001</td>
<td>11111010101000010110110001011</td>
</tr>
<tr>
<td>11</td>
<td>01011101011010001000100101010101</td>
<td>0010011011101011010110010001</td>
</tr>
<tr>
<td>12</td>
<td>110111110100010100011011111111101</td>
<td>1000111011111110010110110011</td>
</tr>
<tr>
<td>13</td>
<td>1100000010011011011100100001000</td>
<td>0111110111101101010001101010</td>
</tr>
<tr>
<td>14</td>
<td>01110001000000100001000101001101</td>
<td>10100001010001001111111010010</td>
</tr>
<tr>
<td>15</td>
<td>1110100001101010111111110000000</td>
<td>0001101011011111111000011001</td>
</tr>
</tbody>
</table>
Flipping a Bit in the State

<table>
<thead>
<tr>
<th>$i$</th>
<th>$A_i$</th>
<th>$W_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>00001111010010111100011111100011</td>
<td>11111000010100101100111010101111</td>
</tr>
<tr>
<td>-3</td>
<td>01000000110010010100100011101100</td>
<td>01010100111111110011111111011110</td>
</tr>
<tr>
<td>-2</td>
<td>01100101110101101111001111111101</td>
<td>10111101111110011001111110110111</td>
</tr>
<tr>
<td>-1</td>
<td>111011111100110110101110001001</td>
<td>01111000010010110011110010111110</td>
</tr>
<tr>
<td>0</td>
<td>01100110100010110001100001110111</td>
<td>10010100110011011010111110101000</td>
</tr>
<tr>
<td>1</td>
<td>10011000000001101111101000001010</td>
<td>01010100110011011011111111111111</td>
</tr>
<tr>
<td>2</td>
<td>10111010011010110011010011001001</td>
<td>10111101111110011001111110110111</td>
</tr>
<tr>
<td>3</td>
<td>11010100111111000100111011001011</td>
<td>00101000010010110011110010111110</td>
</tr>
<tr>
<td>4</td>
<td>10010001001111101000101100011111</td>
<td>00010100110011011010111111010100</td>
</tr>
<tr>
<td>5</td>
<td>01010111010000001011100110100000</td>
<td>10111111101111011001111110110111</td>
</tr>
<tr>
<td>6</td>
<td>10100110000010110111010001110110</td>
<td>00100011000010011011111110101000</td>
</tr>
<tr>
<td>7</td>
<td>11100011111100001001111101100000</td>
<td>11010011110011001111011111011111</td>
</tr>
<tr>
<td>8</td>
<td>10100110111010101010101001001001</td>
<td>10011111111111000100110011110110</td>
</tr>
<tr>
<td>9</td>
<td>11010001110000111011110110111000</td>
<td>01111111101010011010011111111111</td>
</tr>
<tr>
<td>10</td>
<td>01010101110110101100111101101010</td>
<td>11111010100001011011000111111111</td>
</tr>
<tr>
<td>11</td>
<td>11001111111110110110100011110111</td>
<td>00100101110111001111111111111111</td>
</tr>
<tr>
<td>12</td>
<td>01101010000100001001111100111011</td>
<td>10011111000111111100110011111111</td>
</tr>
<tr>
<td>13</td>
<td>11101010011001000000010011001011</td>
<td>01111111111111011110110011111111</td>
</tr>
<tr>
<td>14</td>
<td>11011000111100011110110101101110</td>
<td>10100001010001011111111111111111</td>
</tr>
<tr>
<td>15</td>
<td>01001000101100111000110000011111</td>
<td>00011010110111111111111111111111</td>
</tr>
</tbody>
</table>
Flipping a Bit in the State

<table>
<thead>
<tr>
<th>$i$</th>
<th>$A_i$</th>
<th>$W_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4:</td>
<td>0000111101001011110000111111000011</td>
<td>11111000010100110110110101 1111</td>
</tr>
<tr>
<td>-3:</td>
<td>0100000011001001010100001111011000</td>
<td>001010111010001111101111111110001</td>
</tr>
<tr>
<td>-2:</td>
<td>0110001011101011101101111111111010</td>
<td>1011110111110100111011010101001</td>
</tr>
<tr>
<td>-1:</td>
<td>1110111110011010101110001001</td>
<td>0011100010001001000101001100110100100010</td>
</tr>
<tr>
<td>0:</td>
<td>01100110100010100010001100000011</td>
<td>11111000010100110110110101 1111</td>
</tr>
<tr>
<td>1:</td>
<td>1001100001100000110100001100001010</td>
<td>0101010111010001001000010101001</td>
</tr>
<tr>
<td>2:</td>
<td>101111010011000010100010100001001</td>
<td>1011110111110100111011010101001</td>
</tr>
<tr>
<td>3:</td>
<td>11010100111011100010011010101111</td>
<td>00111000010001001000100110001100100010</td>
</tr>
<tr>
<td>4:</td>
<td>10010001001011000101001100010111</td>
<td>00101001100110111111110100100001</td>
</tr>
<tr>
<td>5:</td>
<td>010101110111000010110011100100001</td>
<td>1011111011101011111011010101001</td>
</tr>
<tr>
<td>6:</td>
<td>1010011010110011000100011100001100</td>
<td>0010011000010011010110001010</td>
</tr>
<tr>
<td>7:</td>
<td>101100000110100011011100001011100000000</td>
<td>1101001111100011000110111010</td>
</tr>
<tr>
<td>8:</td>
<td>1010011010110011000100011100001100</td>
<td>1001111111111000010110011111010</td>
</tr>
<tr>
<td>9:</td>
<td>1101000111000011101101011011101000</td>
<td>0111111101010001101111010100</td>
</tr>
<tr>
<td>10:</td>
<td>01010101110111000111111101010100</td>
<td>11111010101000101101100010101010</td>
</tr>
<tr>
<td>11:</td>
<td>110011111111101001100011011111010</td>
<td>00100110111010111011100011000111</td>
</tr>
<tr>
<td>12:</td>
<td>0110100000100001001011001101101100</td>
<td>100011101111111111000101101011</td>
</tr>
<tr>
<td>13:</td>
<td>1110101001100000100001001100110111</td>
<td>011111111111110011010010011011</td>
</tr>
<tr>
<td>14:</td>
<td>110110001110000101111100111010111011</td>
<td>100000010100010011111111101011</td>
</tr>
<tr>
<td>15:</td>
<td>010010001010011001111000100000011</td>
<td>0001100111011111111000011001</td>
</tr>
</tbody>
</table>
### Flipping a Bit in the State

<table>
<thead>
<tr>
<th>$i$</th>
<th>$A_i$</th>
<th>$E_i = W_i \oplus W_{i+2} \oplus W_{i+8} \oplus W_{i+13} \oplus W_{i+16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>00001111010010111100011111000011</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>-3</td>
<td>010000001100101010001110110000</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>-2</td>
<td>01100010111010110111001111111101</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>-1</td>
<td>111011111101101101101110001001</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0</td>
<td>01100110100010100001100000000000</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>1</td>
<td>10011000000001101111010000000010</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>2</td>
<td>10111010011011000101010011000011</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>3</td>
<td>11010100110111000110110111111001</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>4</td>
<td>10010001001111111010011000100111</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>5</td>
<td>01010110100000010111001110100000</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>6</td>
<td>10100111000000100110110100001110</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>7</td>
<td>10110001111111000100111111011000</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>8</td>
<td>10100110110110101010101010100010</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>9</td>
<td>11010001101100011101110111001000</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>10</td>
<td>01010111111111011101100111101010</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>11</td>
<td>11100111111111011011011011011010</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>12</td>
<td>01101010000100010011011001101100</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>13</td>
<td>11101010011000000100110011010101</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>14</td>
<td>11010001111110011101101101111101</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>15</td>
<td>01001000101001110111000100000011</td>
<td>00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
Outline

1. Background
   - SHA-0/1
   - Collision Attacks

2. (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity

3. Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks

4. Example and Final Remarks
(Second) Preimage Attacks on (Reduced) SHA-0/1

(Second) Preimage Attack on Reduced SHA-0

Basic Technique

\[ A_i \]

\[ IV^* \]

\[ (h - IV)^* \]

\[ E_i \]
Basic Technique

$A_i$

$E_i$
Expect $2^{11}$ solutions.
Basic Technique

$A_i$

$E_i$

$R - 5$

$R - 16$

expect $2^{11}$ solutions
Basic Technique

\[ A_i \]

\[ E_i \]

\[ R - 5 \]

\[ R - 16 \]

expect \( 2^{11} \) solutions
Basic Technique

Expect $2^{11}$ solutions
A basic technique for the (Second) Preimage Attack on Reduced SHA-0.

For $R - 5$, we expect $2^{11}$ solutions.
(Second) Preimage Attacks on (Reduced) SHA-0/1

(Second) Preimage Attack on Reduced SHA-0

Basic Technique

$$A_i$$

$$E_i$$

$$R - 5$$

$$R - 16$$

expect $$2^{11}$$ solutions
Basic Technique

\[ A_i \]

\[ E_i \]

\[ R - 5 \]

\[ R - 16 \]

expect \( 2^{11} \) solutions
Basic Technique

A_i

E_i

R - 5

R - 16

expect $2^{11}$ solutions
Basic Technique

\[ R - 5 \]

expect \( 2^{11} \) solutions
Basic Technique

\[ A_i \]

\[ E_i \]

\[ R - 5 \]

\[ R - 16 \]

expect \(2^{11}\) solutions
Outline

1 Background
   - SHA-0/1
   - Collision Attacks

2 (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity

3 Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks

4 Example and Final Remarks
(Second) Preimage Attacks on (Reduced) SHA-0/1

(Second) Preimage Attack on Reduced SHA-0

Complexity

$A_i$ and $E_i$ with

$R - 5$ and $R - 16$

and

$7$
Complexity

$A_i$

$E_i$

$2^7 \cdot (R - 16)$ trials

$R - 5$

$R - 16$

$7$
Outline

1 Background
   - SHA-0/1
   - Collision Attacks

2 (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity

3 Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks

4 Example and Final Remarks
(Second) Preimage Attacks on (Reduced) SHA-0/1

Improvements

Getting Rid of Those Carries

\[ A_i \]

\[ E_i \]
Improvements

Getting Rid of Those Carries

\[ A_i \]

\[ E_i \]

\[ R - 5 \]

\[ R - 16 \]

expect \( 2^{27-R} \) solutions
Improvements

Getting Rid of Those Carries

expect $2^{27-R}$ solutions
Improvements

Getting Rid of Those Carries

\[ A_i \]

\[ E_i \]

\[ R - 5 \]

\[ R - 16 \]
(Second) Preimage Attacks on (Reduced) SHA-0/1

- Improvements
- Getting Rid of Those Carries

\[ R - 5 \]

\[ R - 16 \]
(Second) Preimage Attacks on (Reduced) SHA-0/1

- Improvements
- Getting Rid of Those Carries

\[ R - 5 \]

\[ R - 16 \]

\[ A_i \]

\[ E_i \]
(Second) Preimage Attacks on (Reduced) SHA-0/1

Improvements

Getting Rid of Those Carries

\[ R - 5 \]

\[ R - 16 \]
(Second) Preimage Attacks on (Reduced) SHA-0/1

Improvements

Getting Rid of Those Carries

\[ A_i \]

\[ E_i \]

\[ R - 5 \]

\[ R - 16 \]
Improvements

Getting Rid of Those Carries

\[ A_i \]

\[ E_i \]

\[ R - 5 \]

\[ R - 16 \]
Improvements

Getting Rid of Those Carries
Improvements

Getting Rid of Those Carries

\[ A_i \]

\[ E_i \]

\[ R - 5 \]

\[ \frac{1}{2} \]

\[ 2^{R-16} + 5 \cdot (R-27) \] trials
Outline

1. Background
   - SHA-0/1
   - Collision Attacks

2. (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity

3. Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks

4. Example and Final Remarks
(Second) Preimage Attacks on (Reduced) SHA-0/1

Improvements

Using More Blocks

\[ \text{near preimage} \]

\[ 2^{2 \cdot (R-16) + 5 \cdot (R-32)} \text{ trials} \]
(Second) Preimage Attacks on (Reduced) SHA-0/1

Improvements

Using More Blocks

\[ A_i \]

\[ C \]

\[ (h - C)^* \]

\[ E_i \]

\[ R \]

\[ R - 16 \]

pseudo preimage

\[ 2^{2(R-16)+5(R-32)} \] trials
Outline

1. Background
   - SHA-0/1
   - Collision Attacks

2. (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity

3. Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks

4. Example and Final Remarks
(Second) Preimage Attacks on (Reduced) SHA-0/1

Improvements

Using Even More Blocks

\[ A_i \]

\[ E_i \]

\[ R + 5 \]

\[ 2^{(R-16)+5(R-37)} \text{ trials} \]
P3Graph ($N$ nodes)
P3Graph \((N/4 \text{ edges})\)
P3Graph ($N/2$ edges)
P3Graph ($N$ edges)
P3Graph \((2 \times N \text{ edges})\)
Second Preimage Complexities for SHA-0

![Graph showing complexities for SHA-0](image-url)
31-step Example

Given Message
Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, ‘and what is the use of a book,’ thou
ght Alice ‘without pictures or conversation?’ So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and pic
Example and Final Remarks

king the daisies, when suddenly a White Rabbit with pink eyes ran close by her. There was nothing so VERY remarkable in that; no r did Alice think k it so VERY muc h out of the way to hear the Rab bit say to itself f, ‘Oh dear! Oh dear! I shall be late!’ (when sh
thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS POCKET, and looked at it, and then hurried on, Alice star...
(Second) Preimage Attacks on (Reduced) SHA-0/1

Example and Final Remarks

ted to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it.
pop down a large rabbit-hole under the hedge. In another moment down went Alice after it, never once considering how in the world she was to get out again..
31-step Example

2nd Preimage
Example and Final Remarks

0000100: fa9e 8747 255d a7e9 cafbd 73dd b87d 3785 ...G%.....s..}7.
0000110: b63d 3c42 2e35 3292 771b 690c a41b 77f1 .=<B.52.w.i...w.
0000120: abfd 84fa d93d 8646 9c3d 7774 b23d 7c79 .......F.=wt.=|y
0000130: aef9 1db8 c192 413e d8ef 6d8b b39e f536 ......A>..m.....6
0000140: 0fa1 c66f 3ffd 955e 6f3b c780 3265 afa6 ...o?..^o;..2e..
0000150: 76ac 6b63 fa32 6784 510b 5c5d cd0d 5413 v.kc.2g.Q.|]..T.
0000160: babd 6b15 c5fd 7cab b17d 7c12 a97d 7d5a ..k...|..}|..}|Z
0000170: d313 a994 f376 99d2 49b4 e6df 154a 5d84 .....v..I.....J].
0000180: 38a0 0a47 d12e 07c9 9065 778b 1b7d 7f34 8..G......ew..}.4
0000190: 54bc dbfd 2cb4 96e2 0ebb 3db1 8afb 8442 T,...,......=....B
00001a0: 74bd 7b59 25fd 7951 86fd 7ff1 717d 78be t.Y%.yQ....q}x.
00001b0: 5357 37b3 6524 7861 6ab2 ec05 8f4c 966e SW7.e$xa*j....L.n
00001c0: ec5d 8b9f 2d7d 6fb7 f36b fba1 eb6d 7b34 .1..-}o..k...m{4
00001d0: bdc5 8179 08c5 5b61 89fd 3b15 2b7d 59ab ...y...[a...;++}Y.
00001e0: f07d 7fcc 36fd 7c85 3cbd 7eac 45fd 85c4 ..}..6.|.<.~.E...
00001f0: 752d aeeef df79 9808 a886 8285 a5dd ff34 u-...y...........4
<table>
<thead>
<tr>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
<th>Hex</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000500</td>
<td>d281</td>
<td>c797</td>
<td>9775</td>
<td>6000</td>
<td>77df</td>
<td>9f95</td>
<td>3737</td>
<td>7fbb</td>
<td>......</td>
<td>u‘.w...77..</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 0000510 | 485c | 79e1 | 0b9c | 7585 | 0344 | efea | 56e4 | f0e6 | H\y...u..D..V...
| 0000520 | 4b7d | 78a6 | 2efd | 7fc3 | f03d | 80c3 | 3f3d | 827a | K}\x........=..?=..z |
| 0000530 | 30c8 | 3047 | 1144 | d3a9 | 104a | 7c41 | 3947 | 4120 | 0.0G.D...J|A9GA |
| 0000540 | 49a0 | 8a9f | 5c1d | 026b | e885 | 6374 | 2775 | 8269 | I...\..k..ct’u.i |
| 0000550 | cb7d | 017c | fcb4 | c107 | 50fb | 6c2e | 37bb | 71a6 | .}\....P.|l.7.q. |
| 0000560 | eb7d | 821c | d3bd | 8633 | 6ffd | 7cbd | 81fd | 77e7 | .}\.....3o.|....w. |
| 0000570 | b2c4 | fef3 | 1c48 | 7d72 | 136a | 2995 | 0afe | 99d5 | .....H}\r.j)..... |
| 0000580 | 6420 | 7368 | 6520 | 7761 | 7320 | 746f | 2067 | 6574 | d she was to get |
| 0000590 | 206f | 7574 | 2061 | 6761 | 696e | 2e0a |   |    | out again.. |
What About SHA-1?