(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

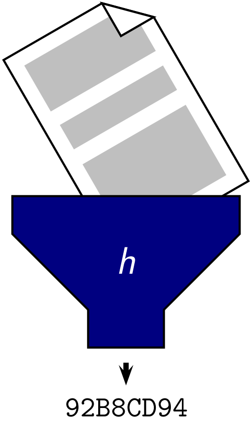
2 (Second) Preimage Attack on Reduced SHA-0
- General Ideas
- Basic Technique
- Complexity

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

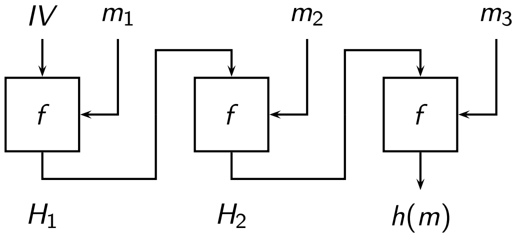
4 Example and Final Remarks

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

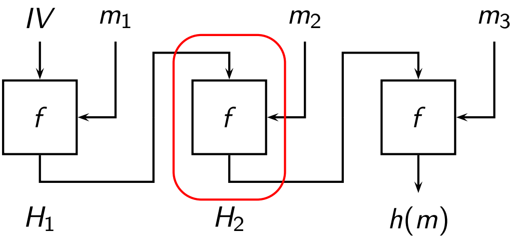


SHA-0/1 Hash Function



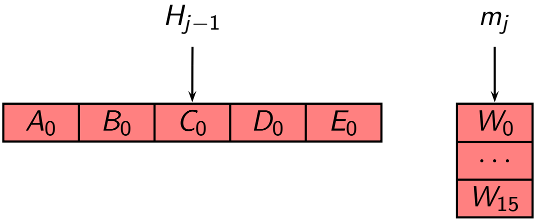
- Iterative hash function.
- 512-bit message blocks m_j.
- 160-bit chaining variable H_j.
- 160-bit hash value $h(m)$.
- Padding, MD-strengthening.

SHA-0/1 Hash Function



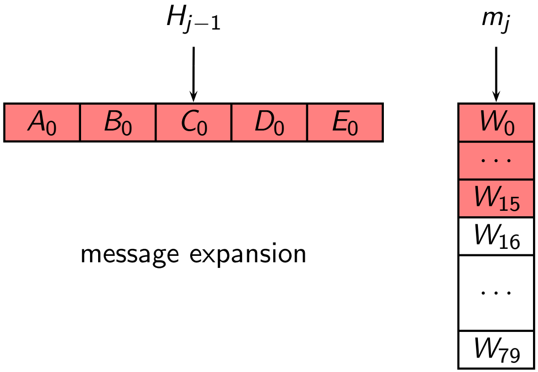
- Iterative hash function.
- 512-bit message blocks m_j.
- 160-bit chaining variable H_j.
- 160-bit hash value $h(m)$.
- Padding, MD-strengthening.

SHA-0/1 Compression Function



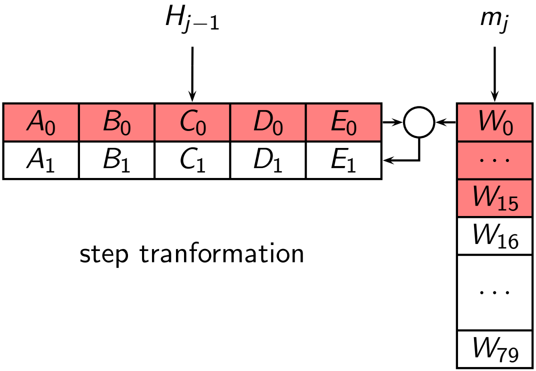
five 32-bit state variables

SHA-0/1 Compression Function



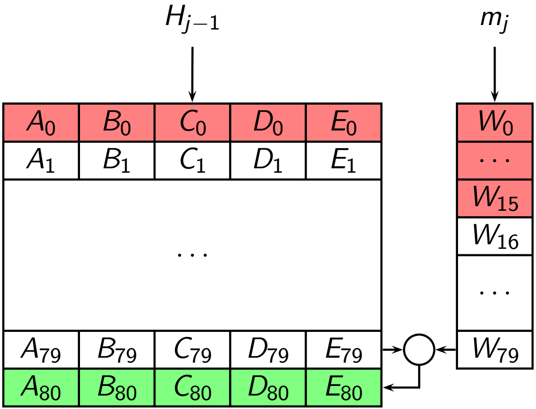
$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 0/1$$

SHA-0/1 Compression Function

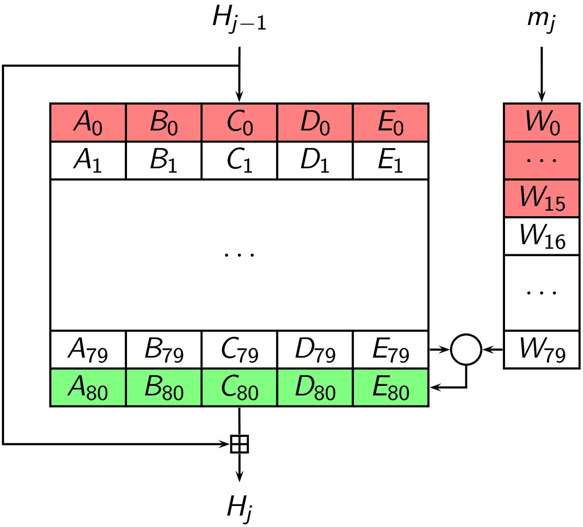


step tranformation

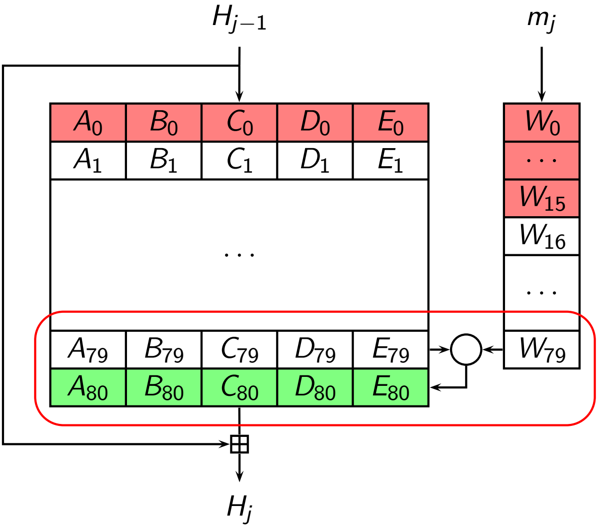
SHA-0/1 Compression Function


H_{j-1}
m_j
A_0 B_0 C_0 D_0 E_0
A_1 B_1 C_1 D_1 E_1
$\dots$
A_{79} B_{79} C_{79} D_{79} E_{79}
A_{80} B_{80} C_{80} D_{80} E_{80}
W_0
$\dots$
W_{15}
W_{16}
$\dots$
W_{79}

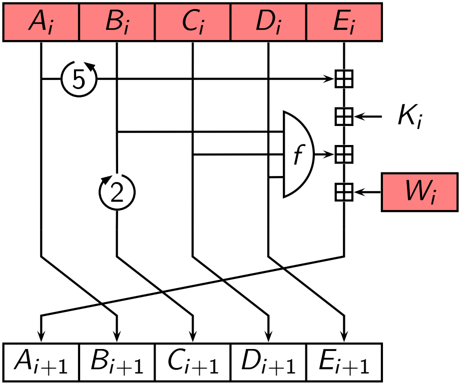
SHA-0/1 Compression Function


H_{j-1}
m_j
A_0 B_0 C_0 D_0 E_0
A_1 B_1 C_1 D_1 E_1
$\cdots$
A_{79} B_{79} C_{79} D_{79} E_{79}
A_{80} B_{80} C_{80} D_{80} E_{80}
W_0
$\cdots$
W_{15}
W_{16}
$\cdots$
W_{79}
$\boxplus$
H_j

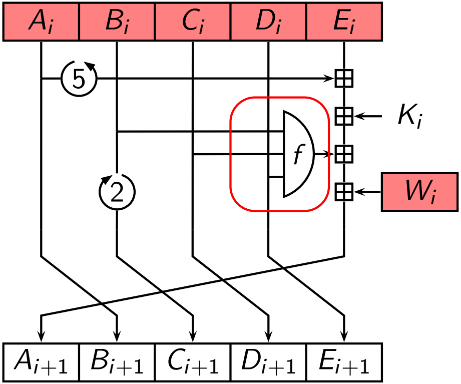
SHA-0/1 Compression Function



SHA Step Transformation



SHA Step Transformation



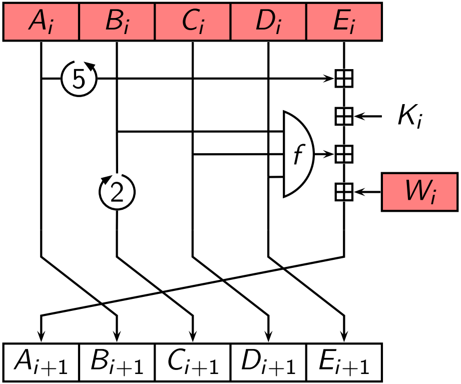
f-Function

- Bitwise boolean function f changes every 20 steps:

$$\begin{aligned}
i = \ \ 0, \ldots, 19: &\quad f_{\mathrm{IF}} = (B \wedge C) \oplus (\neg B \wedge D) \\
i = 20, \ldots, 39: &\quad f_{\mathrm{XOR}} = B \oplus C \oplus D \\
i = 40, \ldots, 59: &\quad f_{\mathrm{MAJ}} = (B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D) \\
i = 60, \ldots, 79: &\quad f_{\mathrm{XOR}} = B \oplus C \oplus D
\end{aligned}$$

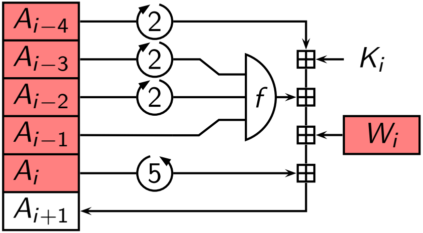
SHA Step Function (Recursive in A_i)

- All state variables can be expressed as a function of A_i

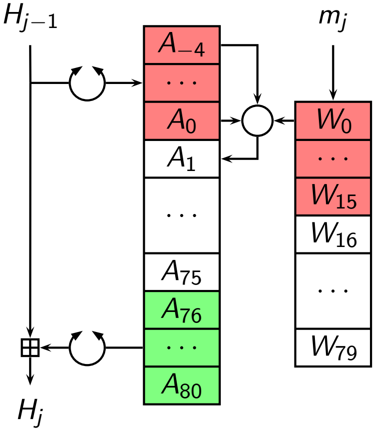


SHA Step Function (Recursive in A_i)

- All state variables can be expressed as a function of A_i



SHA Compression Function (Recursive in A_i)



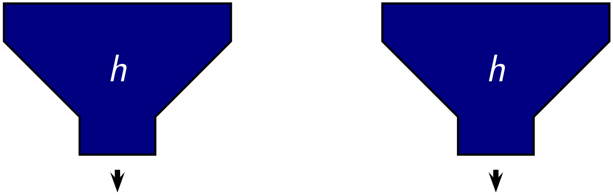
From now on, we only consider state variables A_i.

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

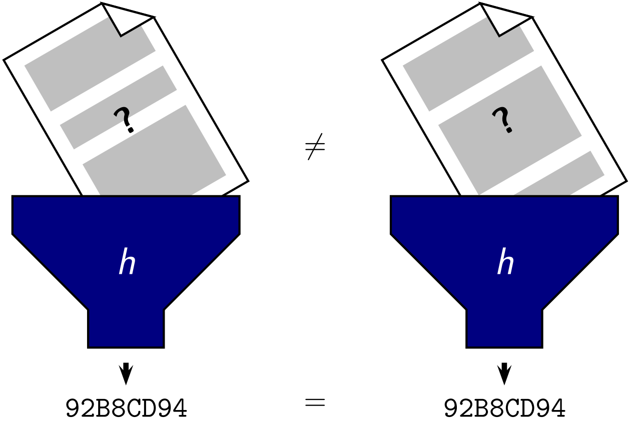
Collision Search Attack

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



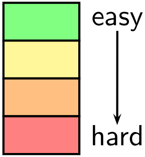
Collision Search Attack

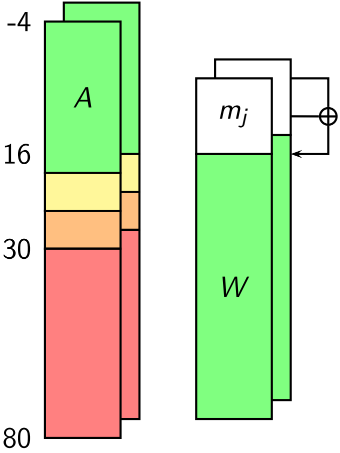
- **Goal:**
  Find two different messages with the same hash value



Differential Cryptanalysis: Not All Bits Are Equal

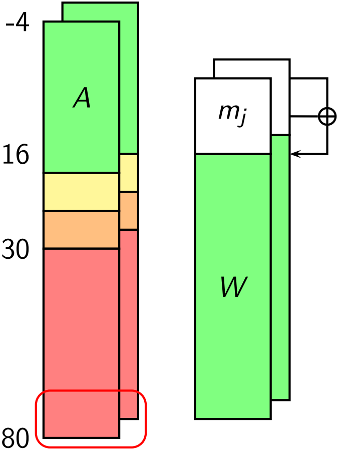
- Limit search space to pairs of messages whose bits are related throughout the hash computation.
- Depending on their position, bits of A_i and W_i depend on m_j in a more or less complex way.





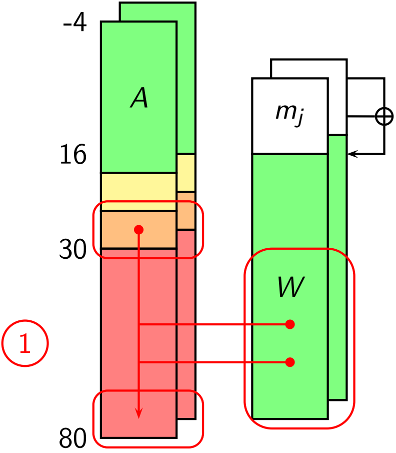
Bottom Part of Characteristic

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



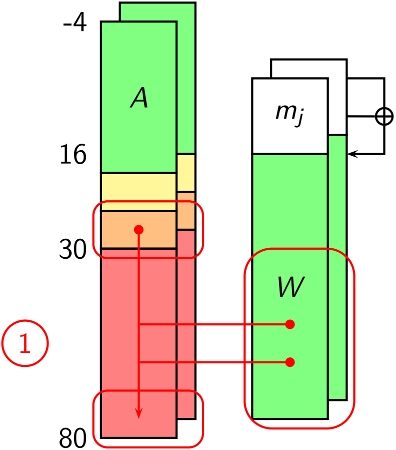
Bottom Part of Characteristic

- 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.



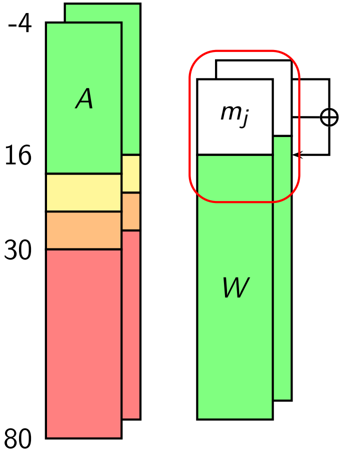
Bottom Part of Characteristic

- 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



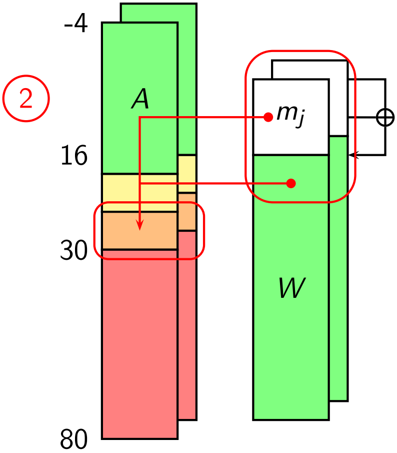
Top Part of Characteristic

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



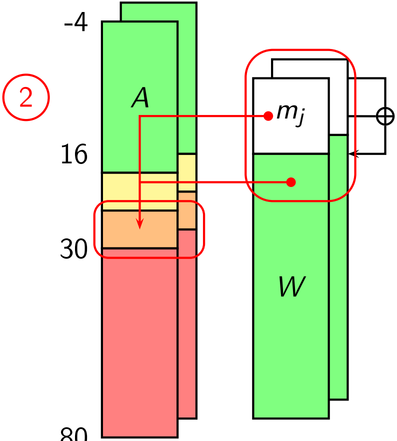
Top Part of Characteristic

- Difference in second part of W determines difference in first part of W.
  - $\longrightarrow$ **Stage 2:** find generalized characteristic which connects the difference in W to the desired difference in A.



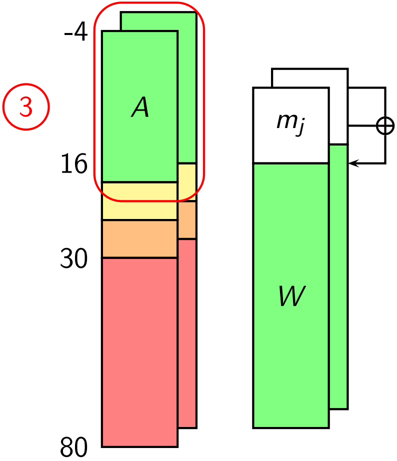
Top Part of Characteristic

- 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]



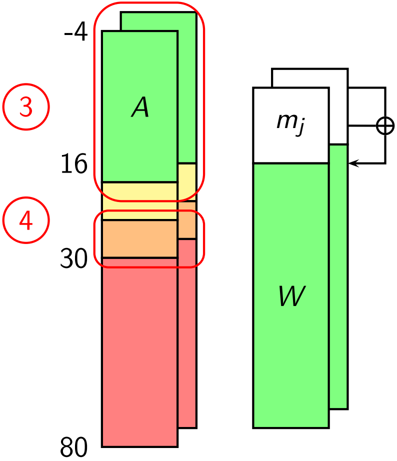
Finding a Message Pair

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



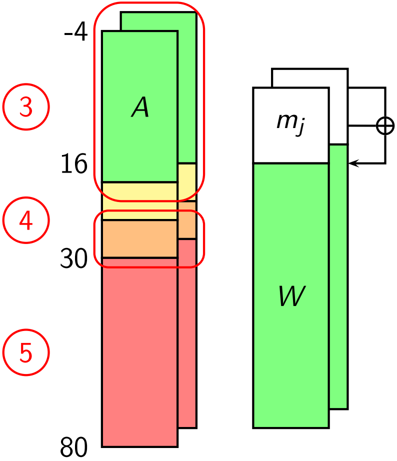
Finding a Message Pair

- **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

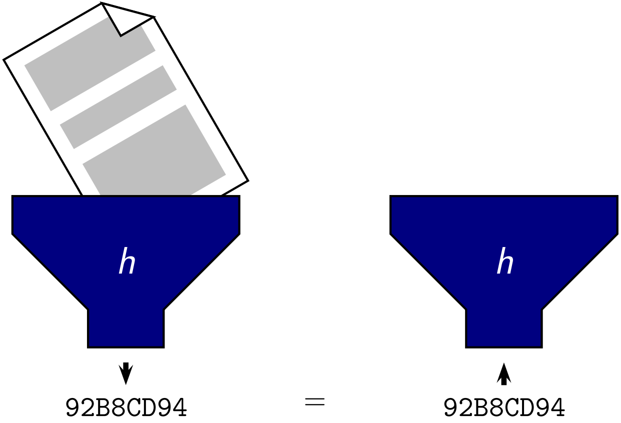
2 (Second) Preimage Attack on Reduced SHA-0
- General Ideas
- Basic Technique
- Complexity

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

4 Example and Final Remarks

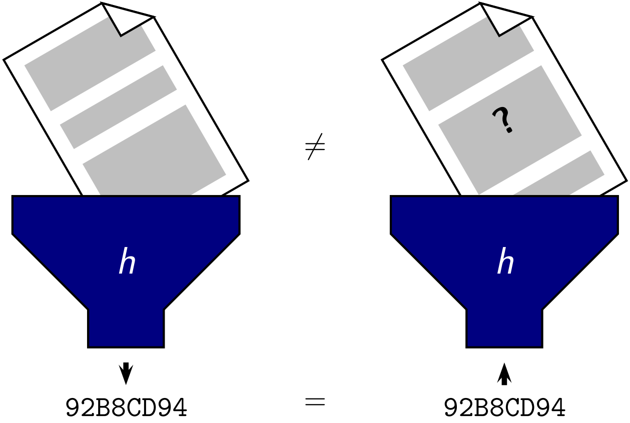
Second Preimage Attack

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



Second Preimage Attack

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



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.

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.
3. If not, try with different special message.

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.
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?

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.

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?

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.

   → 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;

$\longrightarrow$ 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

i	A_i	W_i
-4:	00001111010010111000011111000011	
-3:	01000000110010010101000111011000	
-2:	01100010111010110111001111111010	
-1:	11101111110011011010101110001001	
0:	01100111010001010010001100000001	11111000010100100101101101010111
1:	10011000000001101111010000001010	01010101101001101101101111110001
2:	10111101001101100010101001010001	10111110111110100011101110101001
3:	11010100111011100010011101010111	00111000010000100101001100101010
4:	10010001001111101000101100011111	00010100110011011111100101000001
5:	01010111010000000101110011010000	10111110111011011110010111100010
6:	10100111000001001101101000011100	00100011000010011011011000101010
7:	10110001111100001001111011000000	11010011110011011010001111010011
8:	10100110111010101010101000100010	10001111110111000001011001101100
9:	11010001110000111010110111001000	01111111101010010011011110010100
10:	01010101110110101100111101010100	11111010101000010110110001011011
11:	11100111111111101101100011011101	00100110111010101101000110001111
12:	01101010000100001001011001101100	10001110100111110010010110100110
13:	11101010011001000000100110010101	01111111111101110110101000110110
14:	11011000111000110101101010111011	10100000101000100111111011001011
15:	01001000101001100111000100000011	00011010111001111111100000011001

Flipping a Bit in the Message

i	A_i	W_i
-4:	00001111010010111000011111000011	
-3:	01000000110010010101000111011000	
-2:	01100010111010110111001111111010	
-1:	11101111111001101101010111000101001	
0:	01100111010001010010001100000001	11111000010100100101101101010111
1:	10011000000001101111010000001010	01010101101001101101101111110001
2:	10111101001101100010101001010001	10111110111110100011101110100001
3:	11010100111011100010011101001111	00111000010000100101001100101010
4:	10010001001111101000101000011111	00010100110011011111100101000001
5:	01010111010000000100110011001000	10111110111011011110010111100010
6:	10100111000000001101011100011010	00100011000010011011011000101010
7:	10110001011100000001111001101110	11010011110011011010001111010011
8:	10010110110111101001101110011010	10001111110111000000101100110100
9:	01010000010000011101101000100100	01111111101010010011011110010100
10:	10100101100100110100001111111001	11111010101000010110110001011011
11:	01011101011010100010001010101101	00100110111010101101000110001111
12:	11011111010000101000111011111101	10001110100111110010010110100110
13:	11000000100110110111101010001100	01111111111101110110101000110110
14:	01111001011000001000010010011011	10100000101000100111111011001011
15:	11101000011010101111111110000000	00011010111001111111100000011001

Flipping a Bit in the State

i	A_i	W_i
-4:	00001111010010111000011111000011	
-3:	01000000110010010101000111011000	
-2:	01100010111010110111001111111010	
-1:	11101111110011011010101110001001	
0:	01100111010001010010001100000001	11111000010100100101101101010111
1:	10011000000001101111010000001010	01010101101001101101101111110001
2:	10111101001101100010101001010001	10111110111110100011101110101001
3:	11010100111011100010011101010111	00111000010000100101001100101010
4:	10010001001111101000101100011111	00010100110011011111100101000001
5:	01010111010000001011100110100000	10111110111011011110010111100010
6:	10100111000001001101101000011100	00100011000010011011011000101010
7:	10110001111100001001111011000000	11010011110011011010001111010011
8:	10100110111010101010101000100010	100011111101110000001011001101100
9:	11010001110000111010110111001000	01111111101010010011011110010100
10:	01010101110110101100111101010100	11111010101000010110110001011011
11:	11100111111111011011000110111101	00100110111010101101000110001111
12:	01101010000100001001011001101100	10001110100111110010010110100110
13:	111010100110010000001001100101011	01111111111101110110101000110110
14:	11011000111000110101101010111011	101000000101000100111111011001011
15:	01001000101001100111000100000011	00011010111001111111100000011001

Flipping a Bit in the State

i	A_i	W_i
-4:	00001111010010111000011111000011	
-3:	01000000110010010101000111011000	
-2:	01100010111010110111001111111010	
-1:	11101111110011011010101110001001	
0:	01100111010001010010001100000001	11111000010100100101101101010111
1:	10011000000001101111010000001010	01010101101001101101101111110001
2:	10111101001101100010101001010001	10111110111110100011101110101001
3:	11010100111011100010011101010111	00111000010000100101001100100010
4:	10010001001111101000101100010111	00010100110011011111101001000001
5:	01010111010000000101110011010000	10111110111011011110010111100010
6:	10100111000001001101101000011100	00100011000010011011011000101010
7:	10110001111100001001111011000000	11010011110011011010001111010101
8:	10100110111010101010101000100010	10001111110111000000101100110110
9:	11010001110000111010110111001000	01111111101010010011011110010100
10:	01010101110110101100111101010100	11111010101000010110110001011011
11:	11100111111111011011000110111101	00100110111010101101000110001111
12:	01101010000100001001011001101100	10001110100111110010010110100110
13:	11101010011001000001001100101011	01111111111101110110101000110110
14:	11011000111000110101101010111011	10100000101000100111111011001011
15:	01001000101001100111000100000011	00011010111001111111100000011001

Flipping a Bit in the State

i	A_i	$E_i = W_i \oplus W_{i+2} \oplus W_{i+8} \oplus W_{i+13} \oplus W_{i+16}$
-4:	00001111010010111000011111000011	
-3:	01000000110010010101000111011000	
-2:	01100010111010110111001111111010	
-1:	11101111110011011010101110001001	
0:	01100111010001010010001100000001	00000000000000000000000000000010
1:	10011000000001101111010000001010	00000000000000000000000000001000
2:	10111101001101100010101001010001	00000000000000000000001100000000
3:	11010100111011100010011101010111	00000000000000000000000000001000
4:	10010001001111101000101100010111	00000000000000000000001100000000
5:	01010111010000000101110011010000	00000000000000000000000000000110
6:	10100111000001001101101000011100	00000000000000000000000000000010
7:	10110001111100001001111011000000	00000000000000000000000000000110
8:	10100110111010101010101000100010	00000000000000000000000000000010
9:	11010001110000111010110111001000	00000000000000000000000000000000
10:	01010101110110101100111101010100	00000000000000000000000000000000
11:	11100111111111011011000110111101	
12:	01101010000100001001011001101100	
13:	11101010011001000001001100101011	
14:	11011000111000110101101010111011	
15:	01001000101001100111000100000011	

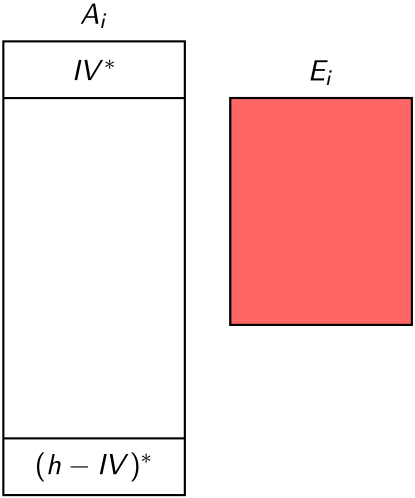
Outline

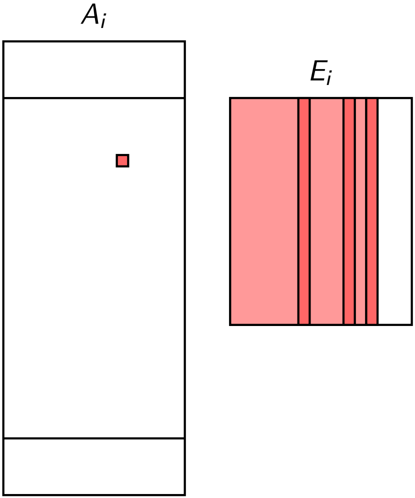
1 Background
- SHA-0/1
- Collision Attacks

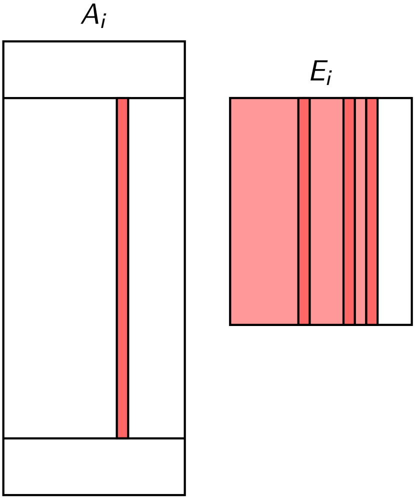
2 (Second) Preimage Attack on Reduced SHA-0
- General Ideas
- Basic Technique
- Complexity

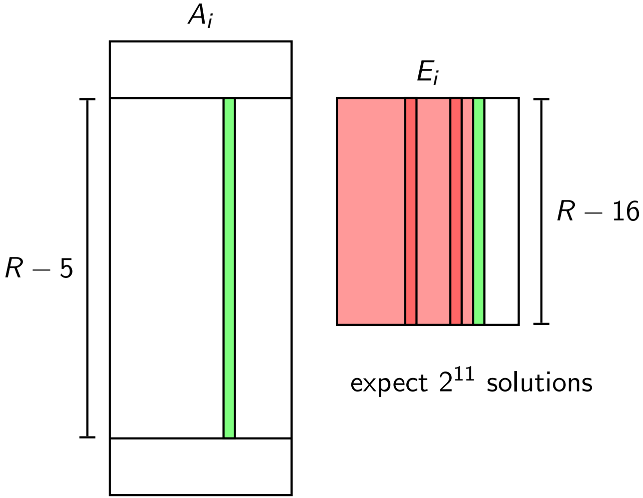
3 Improvements
- Getting Rid of Those Carries
- Using More Blocks
- Using Even More Blocks

4 Example and Final Remarks

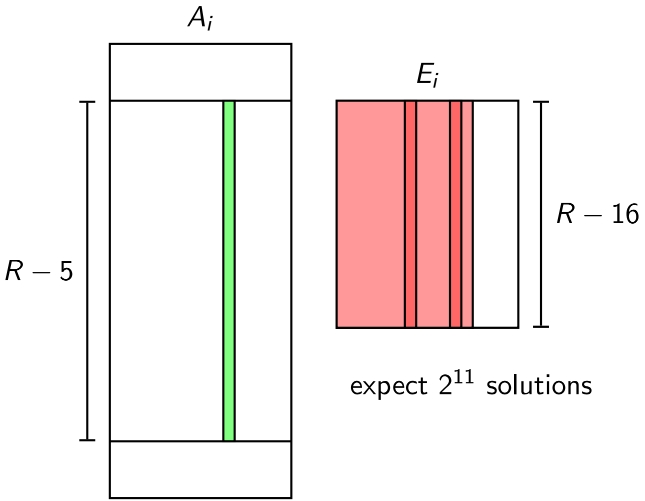

A_i
IV^*
E_i
$(h - IV)^*$


A_i
E_i

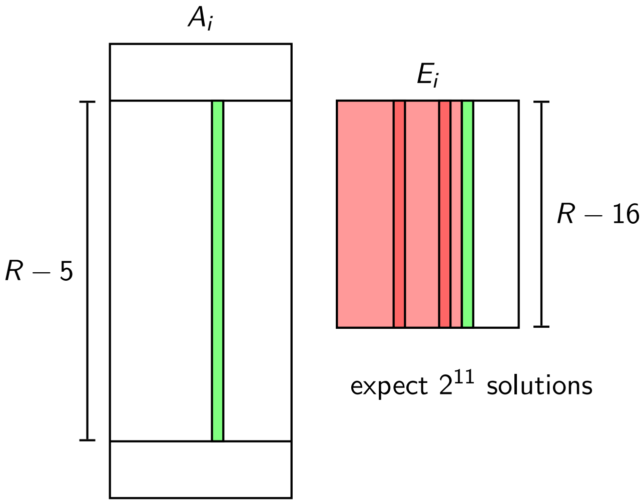

A_i
E_i



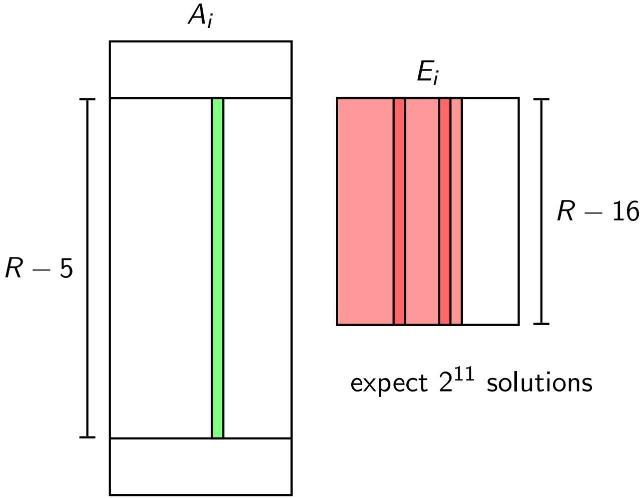
expect 2^{11} solutions



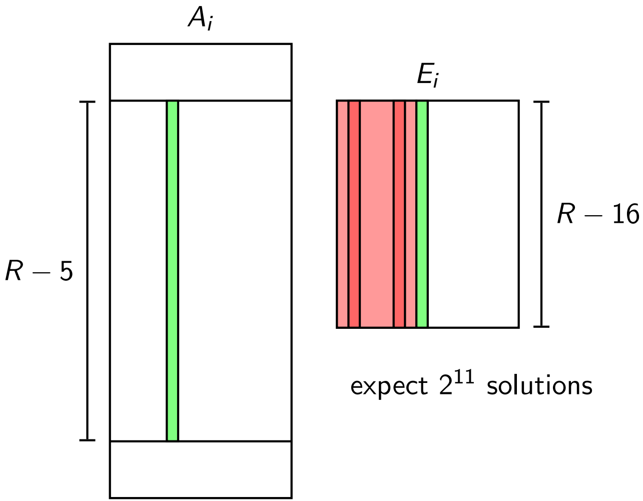
expect 2^{11} solutions



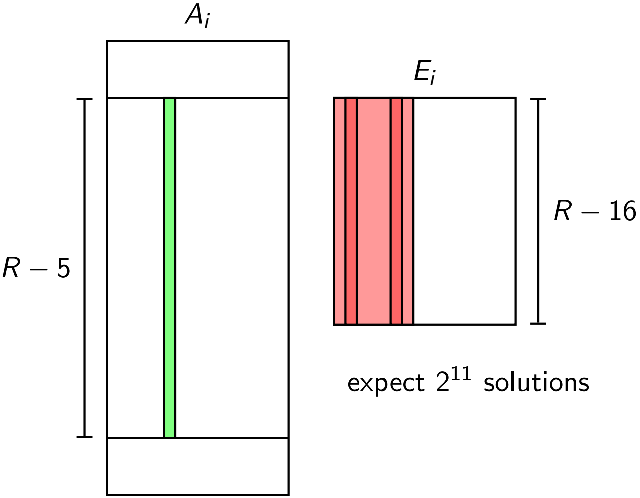
expect 2^{11} solutions



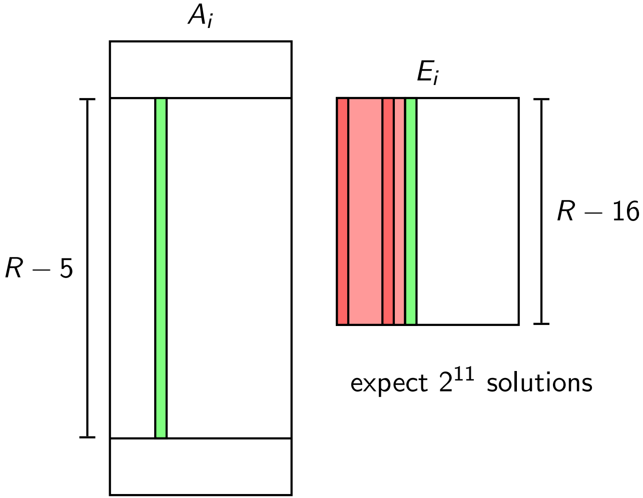
expect 2^{11} solutions



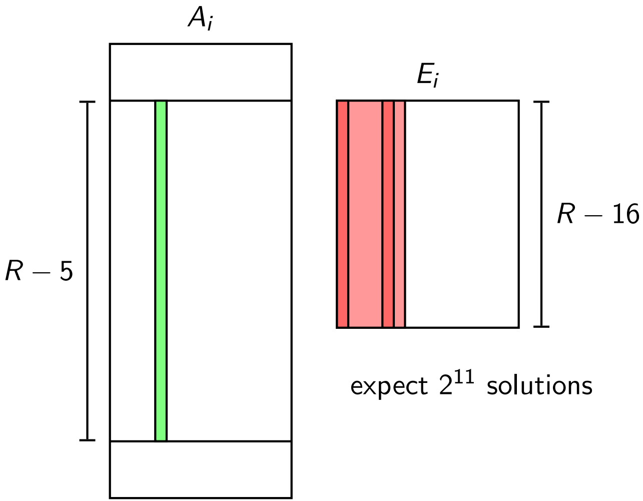
expect 2^{11} solutions

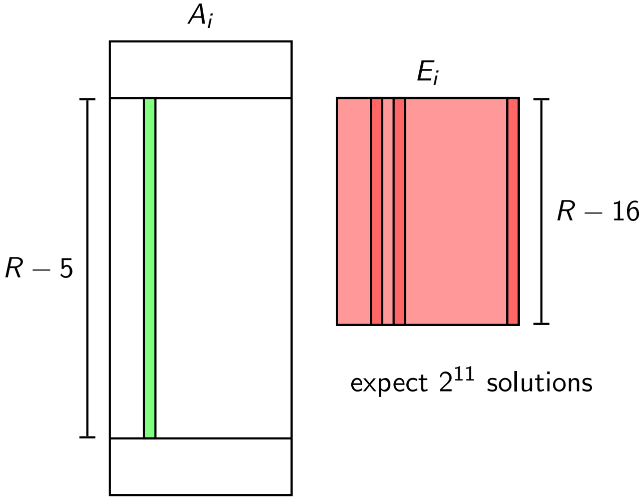


expect 2^{11} solutions

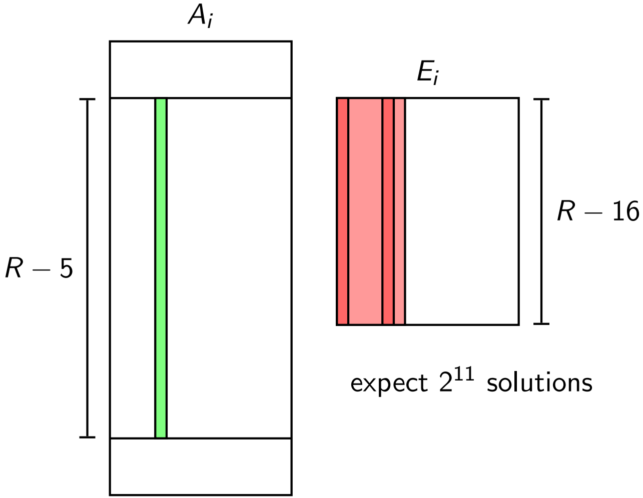


expect 2^{11} solutions


A_i
E_i
$R-5$
$R-16$
expect 2^{11} solutions



expect 2^{11} solutions



expect 2^{11} solutions

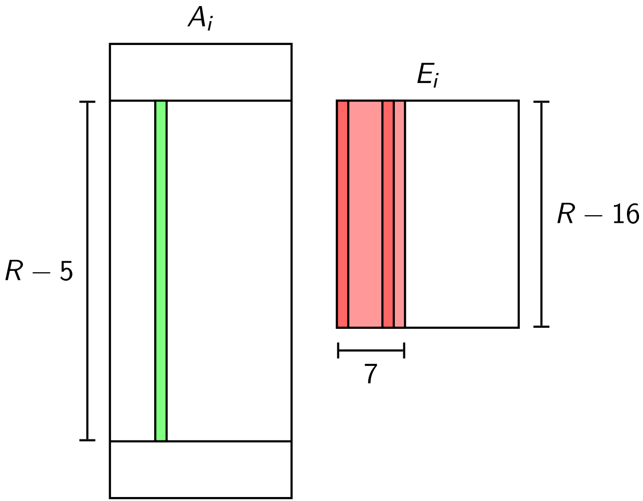
Outline

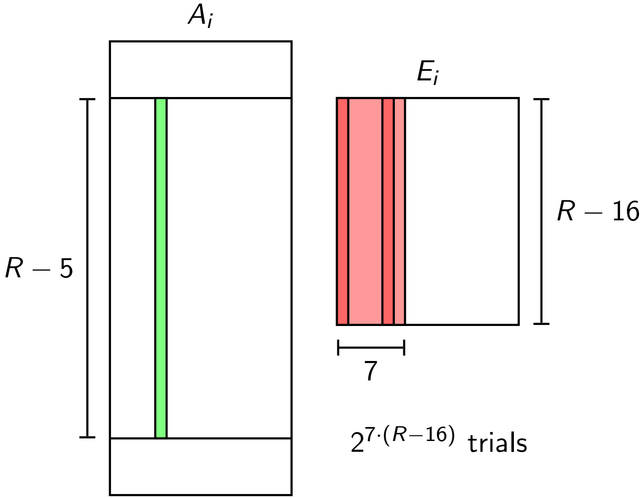
1 Background
- SHA-0/1
- Collision Attacks

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


A_i
$R - 5$
E_i
$R - 16$
7



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

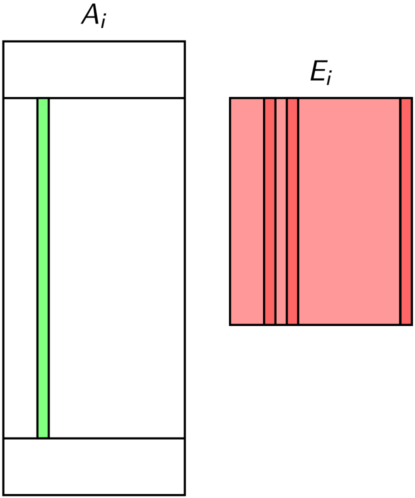
Outline

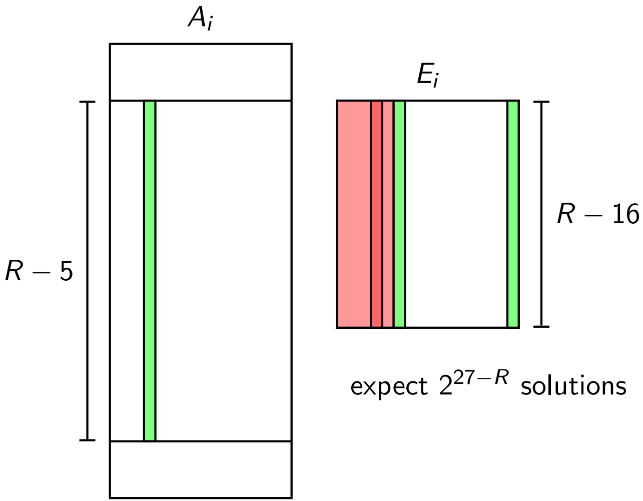
1 Background
- SHA-0/1
- Collision Attacks

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

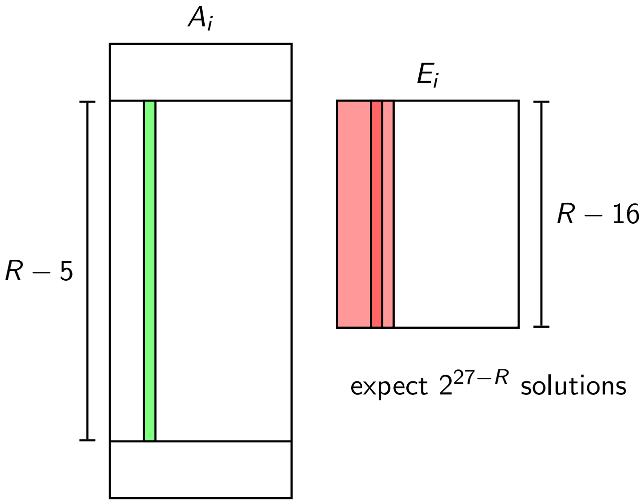
3 Improvements
- Getting Rid of Those Carries
- Using More Blocks
- Using Even More Blocks

4 Example and Final Remarks

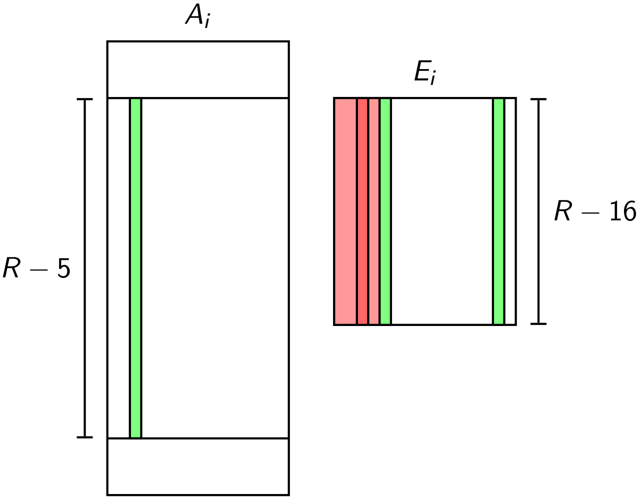

A_i
E_i

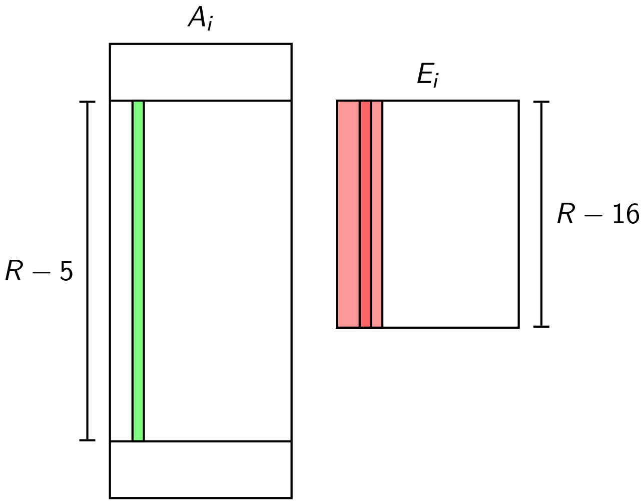


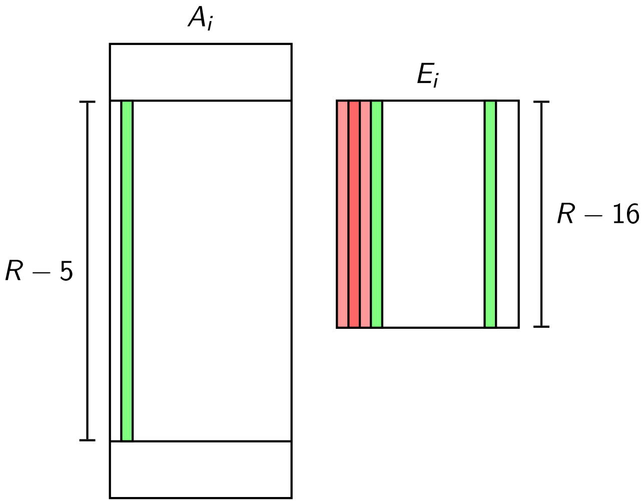
expect 2^{27-R} solutions

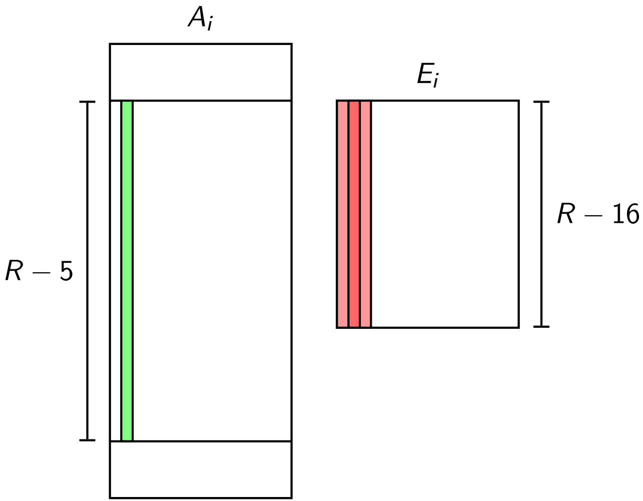


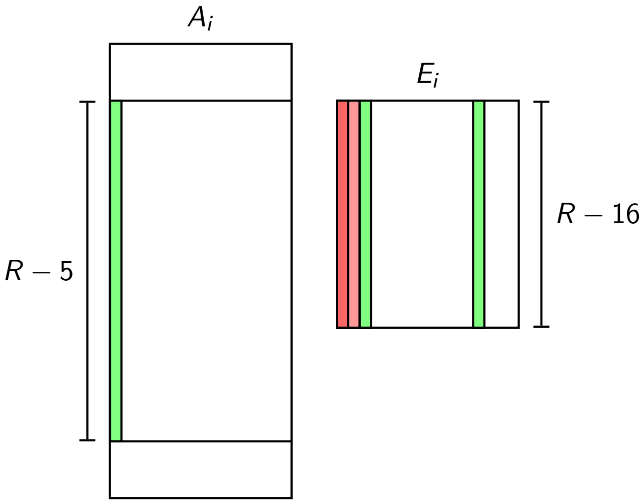
expect 2^{27-R} solutions

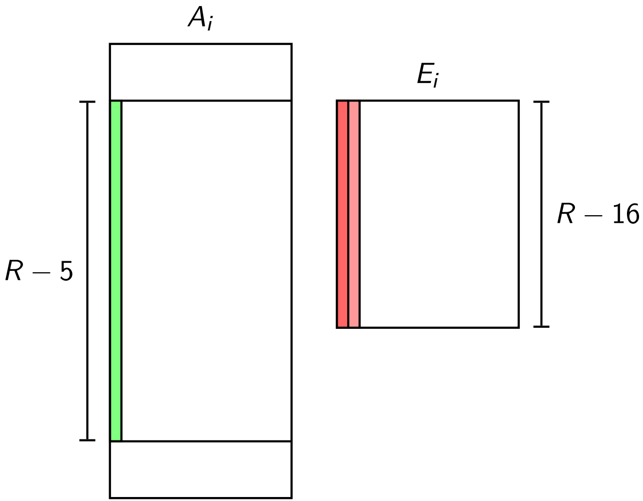

A_i
E_i
$R - 5$
$R - 16$

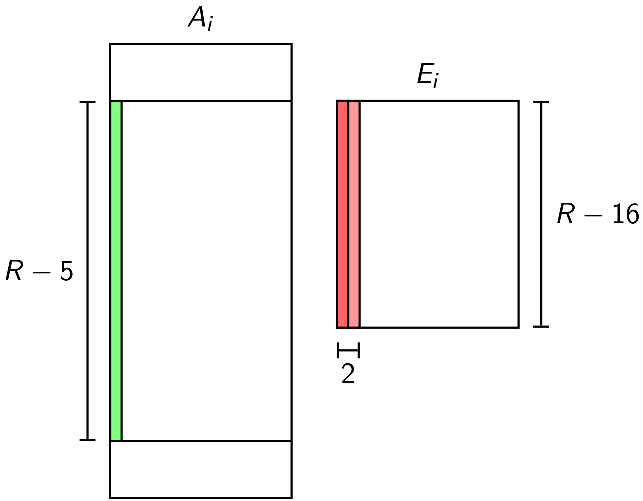

A_i
E_i
$R - 5$
$R - 16$

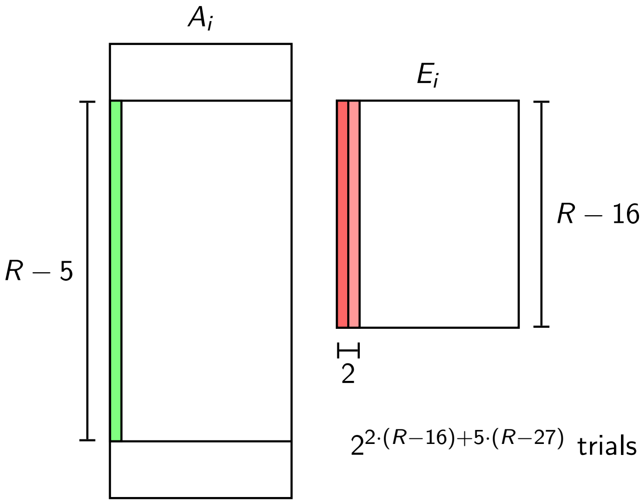

A_i
E_i
$R - 5$
$R - 16$


A_i
$R - 5$
E_i
$R - 16$


A_i
E_i
$R - 5$
$R - 16$


A_i
E_i
$R - 5$
$R - 16$

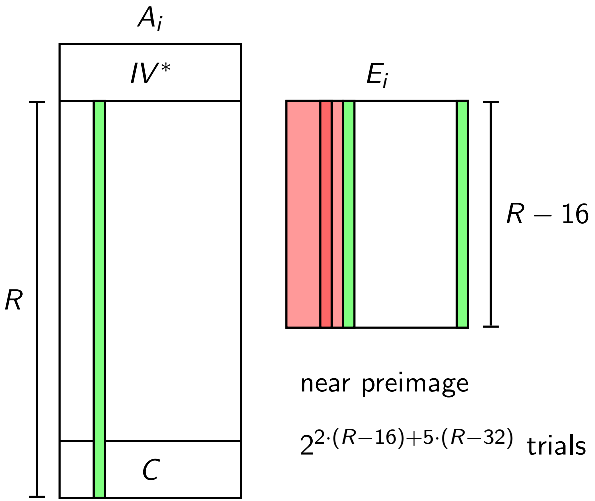

A_i
$R-5$
E_i
$R-16$
2



$2^{2\cdot(R-16)+5\cdot(R-27)}$ trials

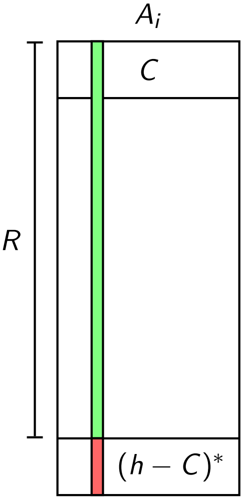
Outline

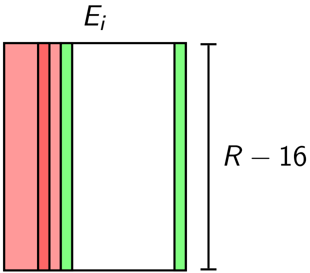
1. Background
   - SHA-0/1
   - Collision Attacks
2. (Second) Preimage Attack on Reduced SHA-0
   - General Ideas
   - Basic Technique
   - Complexity
3. Improvements
   - Getting Rid of Those Carries
   - Using More Blocks
   - Using Even More Blocks
4. Example and Final Remarks



near preimage

$2^{2\cdot(R-16)+5\cdot(R-32)}$ trials





pseudo preimage

$2^{2\cdot(R-16)+5\cdot(R-32)}$ trials

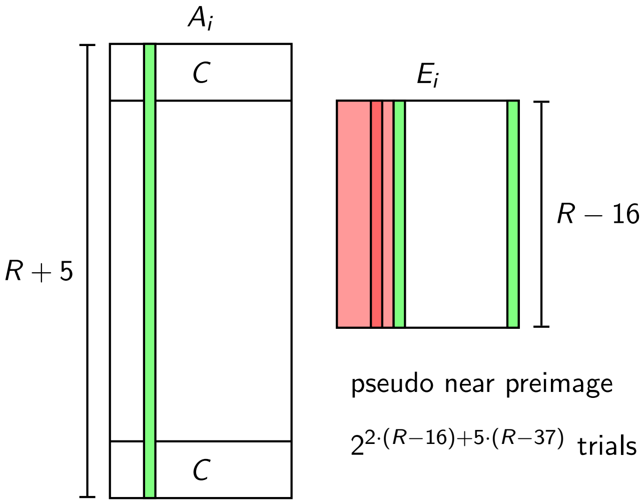
Outline

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

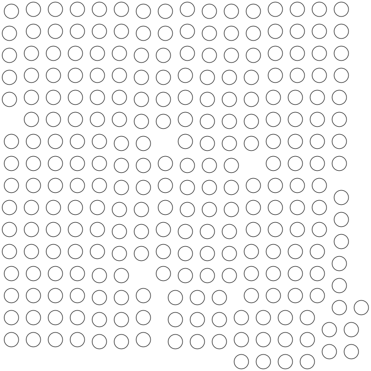
4 Example and Final Remarks



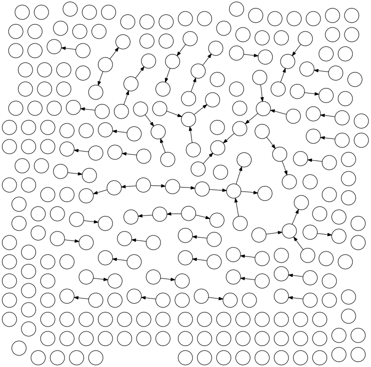
pseudo near preimage

$2^{2\cdot(R-16)+5\cdot(R-37)}$ trials

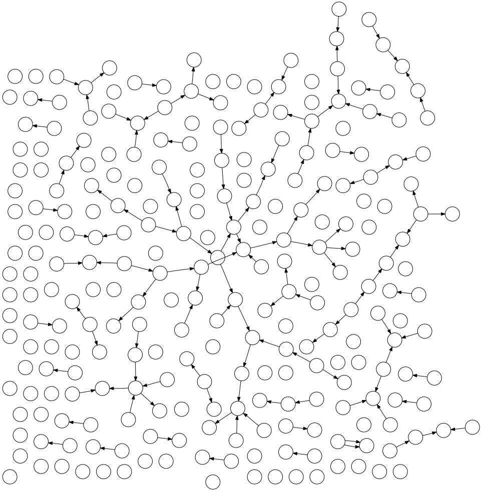
P3Graph (N nodes)



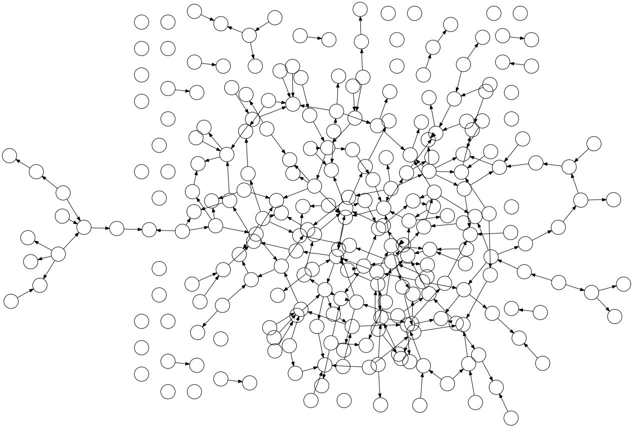
P3Graph ($N/4$ edges)



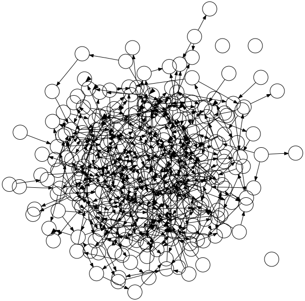
P3Graph ($N/2$ edges)



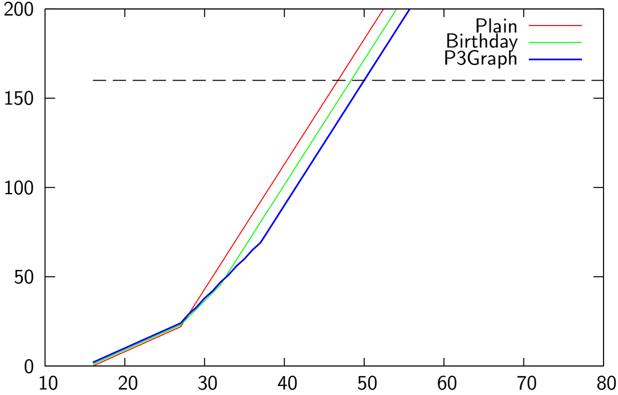
P3Graph (N edges)



P3Graph ($2 \times N$ edges)



Second Preimage Complexities for SHA-0



31-step Example

Given Message

```
0000000: 416c 6963 6520 7761 7320 6265 6769 6e6e  Alice was beginn
0000010: 696e 6720 746f 2067 6574 2076 6572 7920  ing to get very
0000020: 7469 7265 6420 6f66 2073 6974 7469 6e67  tired of sitting
0000030: 2062 7920 6865 7220 7369 7374 6572 206f   by her sister o
0000040: 6e20 7468 6520 6261 6e6b 2c20 616e 6420  n the bank, and
0000050: 6f66 2068 6176 696e 6720 6e6f 7468 696e  of having nothin
0000060: 6720 746f 2064 6f3a 206f 6e63 6520 6f72  g to do: once or
0000070: 2074 7769 6365 2073 6865 2068 6164 2070   twice she had p
0000080: 6565 7065 6420 696e 746f 2074 6865 2062  eeped into the b
0000090: 6f6f 6b20 6865 7220 7369 7374 6572 2077  ook her sister w
00000a0: 6173 2072 6561 6469 6e67 2c20 6275 7420  as reading, but
00000b0: 6974 2068 6164 206e 6f20 7069 6374 7572  it had no pictur
00000c0: 6573 206f 7220 636f 6e76 6572 7361 7469  es or conversati
00000d0: 6f6e 7320 696e 2069 742c 2060 616e 6420  ons in it, ‘and
00000e0: 7768 6174 2069 7320 7468 6520 7573 6520  what is the use
00000f0: 6f66 2061 2062 6f6f 6b2c 2720 7468 6f75  of a book,’ thou
```

```
0000100: 6768 7420 416c 6963 6520 6077 6974 686f  ght Alice ‘witho
0000110: 7574 2070 6963 7475 7265 7320 6f72 2063  ut pictures or c
0000120: 6f6e 7665 7273 6174 696f 6e3f 2720 536f  onversation?’ So
0000130: 2073 6865 2077 6173 2063 6f6e 7369 6465   she was conside
0000140: 7269 6e67 2069 6e20 6865 7220 6f77 6e20  ring in her own 
0000150: 6d69 6e64 2028 6173 2077 656c 6c20 6173  mind (as well as
0000160: 2073 6865 2063 6f75 6c64 2c20 666f 7220   she could, for 
0000170: 7468 6520 686f 7420 6461 7920 6d61 6465  the hot day made
0000180: 2068 6572 2066 6565 6c20 7665 7279 2073   her feel very s
0000190: 6c65 6570 7920 616e 6420 7374 7570 6964  leepy and stupid
00001a0: 292c 2077 6865 7468 6572 2074 6865 2070  ), whether the p
00001b0: 6c65 6173 7572 6520 6f66 206d 616b 696e  leasure of makin
00001c0: 6720 6120 6461 6973 792d 6368 6169 6e20  g a daisy-chain 
00001d0: 776f 756c 6420 6265 2077 6f72 7468 2074  would be worth t
00001e0: 6865 2074 726f 7562 6c65 206f 6620 6765  he trouble of ge
00001f0: 7474 696e 6720 7570 2061 6e64 2070 6963  tting up and pic
```

```
0000200: 6b69 6e67 2074 6865 2064 6169 7369 6573  king the daisies
0000210: 2c20 7768 656e 2073 7564 6465 6e6c 7920  , when suddenly
0000220: 6120 5768 6974 6520 5261 6262 6974 2077  a White Rabbit w
0000230: 6974 6820 7069 6e6b 2065 7965 7320 7261  ith pink eyes ra
0000240: 6e20 636c 6f73 6520 6279 2068 6572 2e20  n close by her.
0000250: 5468 6572 6520 7761 7320 6e6f 7468 696e  There was nothin
0000260: 6720 736f 2056 4552 5920 7265 6d61 726b  g so VERY remark
0000270: 6162 6c65 2069 6e20 7468 6174 3b20 6e6f  able in that; no
0000280: 7220 6469 6420 416c 6963 6520 7468 696e  r did Alice thin
0000290: 6b20 6974 2073 6f20 5645 5259 206d 7563  k it so VERY muc
00002a0: 6820 6f75 7420 6f66 2074 6865 2077 6179  h out of the way
00002b0: 2074 6f20 6865 6172 2074 6865 2052 6162   to hear the Rab
00002c0: 6269 7420 7361 7920 746f 2069 7473 656c  bit say to itsel
00002d0: 662c 2060 4f68 2064 6561 7221 204f 6820  f, ‘Oh dear! Oh
00002e0: 6465 6172 2120 4920 7368 616c 6c20 6265  dear! I shall be
00002f0: 206c 6174 6521 2720 2877 6865 6e20 7368   late!’ (when sh
0000300: 6520 7468 6f75 6768 7420 6974 206f 7665  e thought it ove
0000310: 7220 6166 7465 7277 6172 6473 2c20 6974  r afterwards, it
0000320: 206f 6363 7572 7265 6420 746f 2068 6572   occurred to her
0000330: 2074 6861 7420 7368 6520 6f75 6768 7420   that she ought 
0000340: 746f 2068 6176 6520 776f 6e64 6572 6564  to have wondered
0000350: 2061 7420 7468 6973 2c20 6275 7420 6174   at this, but at
0000360: 2074 6865 2074 696d 6520 6974 2061 6c6c   the time it all
0000370: 2073 6565 6d65 6420 7175 6974 6520 6e61   seemed quite na
0000380: 7475 7261 6c29 3b20 6275 7420 7768 656e  tural); but when
0000390: 2074 6865 2052 6162 6269 7420 6163 7475   the Rabbit actu
00003a0: 616c 6c79 2054 4f4f 4b20 4120 5741 5443  ally TOOK A WATC
00003b0: 4820 4f55 5420 4f46 2049 5453 2057 4149  H OUT OF ITS WAI
00003c0: 5354 434f 4154 2d50 4f43 4b45 542c 2061  STCOAT-POCKET, a
00003d0: 6e64 206c 6f6f 6b65 6420 6174 2069 742c  nd looked at it,
00003e0: 2061 6e64 2074 6865 6e20 6875 7272 6965   and then hurrie
00003f0: 6420 6f6e 2c20 416c 6963 6520 7374 6172  d on, Alice star
```

```
0000400: 7465 6420 746f 2068 6572 2066 6565 742c  ted to her feet,
0000410: 2066 6f72 2069 7420 666c 6173 6865 6420   for it flashed
0000420: 6163 726f 7373 2068 6572 206d 696e 6420  across her mind
0000430: 7468 6174 2073 6865 2068 6164 206e 6576  that she had nev
0000440: 6572 2062 6566 6f72 6520 7365 656e 2061  er before seen a
0000450: 2072 6162 6269 7420 7769 7468 2065 6974   rabbit with eit
0000460: 6865 7220 6120 7761 6973 7463 6f61 742d  her a waistcoat-
0000470: 706f 636b 6574 2c20 6f72 2061 2077 6174  pocket, or a wat
0000480: 6368 2074 6f20 7461 6b65 206f 7574 206f  ch to take out o
0000490: 6620 6974 2c20 616e 6420 6275 726e 696e  f it, and burnin
00004a0: 6720 7769 7468 2063 7572 696f 7369 7479  g with curiosity
00004b0: 2c20 7368 6520 7261 6e20 6163 726f 7373  , she ran across
00004c0: 2074 6865 2066 6965 6c64 2061 6674 6572   the field after
00004d0: 2069 742c 2061 6e64 2066 6f72 7475 6e61   it, and fortuna
00004e0: 7465 6c79 2077 6173 206a 7573 7420 696e  tely was just in
00004f0: 2074 696d 6520 746f 2073 6565 2069 7420   time to see it
```

```
0000500: 706f 7020 646f 776e 2061 206c 6172 6765  pop down a large
0000510: 2072 6162 6269 742d 686f 6c65 2075 6e64   rabbit-hole und
0000520: 6572 2074 6865 2068 6564 6765 2e20 496e  er the hedge. In
0000530: 2061 6e6f 7468 6572 206d 6f6d 656e 7420   another moment 
0000540: 646f 776e 2077 656e 7420 416c 6963 6520  down went Alice 
0000550: 6166 7465 7220 6974 2c20 6e65 7665 7220  after it, never 
0000560: 6f6e 6365 2063 6f6e 7369 6465 7269 6e67  once considering
0000570: 2068 6f77 2069 6e20 7468 6520 776f 726c   how in the worl
0000580: 6420 7368 6520 7761 7320 746f 2067 6574  d she was to get
0000590: 206f 7574 2061 6761 696e 2e0a             out again..
```

31-step Example

2nd Preimage

```
0000000: 6093 e793 8844 423f cf3e 4140 3479 5078  `....DB?.>A@4yPx
0000010: f8ac 0a92 7e6a 1956 d8b7 b004 1bf9 027f  ....~j.V........
0000020: 13fd 7b20 5cbd 783c 9b3d 78d2 e0bd 8106  ..{ \.x<.=x.....
0000030: fee5 2a1d 8efe 23eb 6bd8 7621 354f 0c9c  ..*...#.k.v!5O..
0000040: 9b86 3bbf 6469 db87 b11d 9195 707d 3f5a  ..;.di......p}?Z
0000050: 277b 582e 44fa 9440 a57c be61 14bc 7c39  '{X.D..@.|.a..|9
0000060: aabc 785e 3c7d 85ef 35bd 855d 1b7d 84fd  ..x^<}..5..].}..
0000070: a7d6 c497 a55a d1ae 21ea 5210 19cc f5e1  .....Z..!.R.....
0000080: b6a5 86d7 e20e 085d e7ab ab81 dd74 ffad  .......].....t..
0000090: 6a33 7421 b5cf 5fa2 c709 48b3 836d 6f2a  j3t!.._...H..mo*
00000a0: 8d3d 7e50 eefd 793c 2cbd 84ea d83d 78bc  .=~P..y<,....=x.
00000b0: 7d7b 64a9 483c 18f3 f559 a0d5 bf69 d5f8  }{d.H<...Y...i..
00000c0: 5e7d 920f 9cbe 10a2 0d5d 5bb1 453d 7b31  ^}.......][.E={1
00000d0: d03d 7f7f fe6d 019b 5fa4 fed5 fbf5 79dd  .=...m.._.....y.
00000e0: 37bd 7ced ddfd 79aa 18fd 7da7 063d 8622  7.|...y...}..=."
00000f0: ece1 65d6 0372 499e 9c7c 8472 5267 8c88  ..e..rI..|.rRg..
```

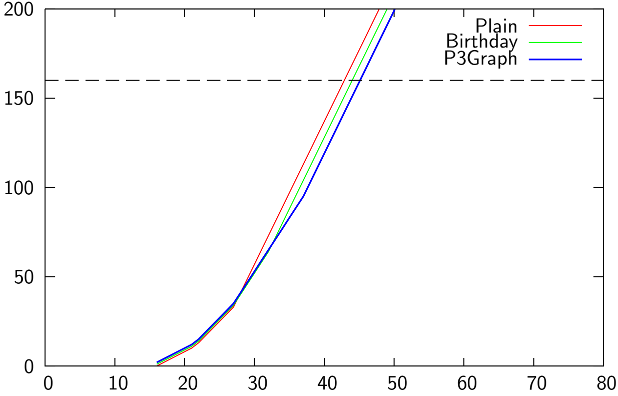
```
0000100: fa9e 8747 255d a7e9 cafd 73dd b87d 3785  ...G%]....s..}7.
0000110: b63d 3c42 2e35 3292 771b 690c a41b 77f1  .=<B.52.w.i...w.
0000120: abfd 84fa d93d 8646 9c3d 7774 b23d 7c79  .....=.F.=wt.=|y
0000130: aef9 1db8 c192 413e d8ef 6d8b b39e f536  ......A>..m....6
0000140: 0fa1 c66f 3ffd 955e 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 96c2 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$xaj....L.n
00001c0: ec5d 8b9f 2d7d 6fb7 f36b fba1 eb6d 7b34  .]..-}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 aeef df79 9808 a886 8285 a5dd ff34  u-...y.........4
```

```
0000200: 5c8d 9e8f b2ba 8079 167d 657a c33d 43bc  \......y.}ez.=C.
0000210: 1db9 76d0 e3e9 70df 986d 7c1e 657d 8363  ..v...p..m|.e}.c
0000220: 613d 7750 3e3d 7944 fa7d 77a5 373d 7765  a=wP>=yD.}w.7=we
0000230: c560 ac62 e5b2 47dd 01fe aebe e8ac e99a  .‘.b..G.........
0000240: 887d 930f 5f7c 0fc3 f789 7790 de7d 7f71  .}.._|....w..}.q
0000250: b4bd 7ba9 4d3d 6c8a 1579 75b8 c439 84d2  ..{.M=l..yu..9..
0000260: 513d 7b27 a3bd 7f43 357d 7fa9 e9bd 7704  Q={’...C5}....w.
0000270: ff1d 6a35 02bd 3859 2703 d027 4915 5452  ..j5..8Y’..’I.TR
0000280: dd05 9eb7 577a 8263 01a2 a46f d8bd 5daa  ....Wz.c...o..].
0000290: eebd 72a2 21db 732a 98b3 f657 d033 fb18  ..r.!.s*...W.3..
00002a0: 987d 82f5 f2bd 7c08 2dfd 85c8 38fd 82ca  .}....|.-...8...
00002b0: 5939 ee8e 140f 5b3d 0cc9 9c81 9c92 5965  Y9....[=......Ye
00002c0: 3b9d 96af 8b47 7d9f e2ff 8392 c6ac ff71  ;....G}........q
00002d0: b5f3 81bd d482 750b 5749 f1aa 4cfc e77a  ......u.WI..L..z
00002e0: b1fd 7ead e23d 7900 aabd 7f55 3cbd 83f5  ..~..=y....U<...
00002f0: 97bb e4dd 6941 50cd 567f 37d0 3e5c 9e26  ....iAP.V.7.>\.&
```

```
0000300: 7a23 d3cf cdbc 6851 fc6b 6fdc 0a73 e75c  z#....hQ.ko..s.\
0000310: 5c53 e94b c211 c83c 9d3b 59c7 77fd 7a5a  \S.K...<.;Y.w.zZ
0000320: 9afd 7b0b 883d 835f c8fd 7f30 98bd 7f34  ..{..=._...0...4
0000330: 570a e920 9bc7 4e38 9d9f 7faa 7e51 9dbd  W.. ..N8....~Q..
0000340: 0f0c c697 20e5 9f98 9c99 fff8 442d 7383  .... .......D-s.
0000350: 583a 2e86 7bc5 a5a9 48e1 57da 0675 61ce  X:..{...H.W..ua.
0000360: 1a3d 78d0 23bd 7ac5 24fd 804e 473d 7aa0  .=x.#.z.$..NG=z.
0000370: b7c3 6cdc 9ce1 2251 87d2 dbef 4739 a47c  ..l..."Q....G9.|
0000380: 9d15 92a7 4a9c bcc5 74a9 579c 41dd 7e99  ....J...t.W.A.~.
0000390: a8db 7a99 398f 4864 1fa4 54bd 9d6c 7c8e  ..z.9.Hd..T..l|.
00003a0: 57bd 7ac7 12fd 84b9 703d 7a02 9cbd 7c37  W.z.....p=z...|7
00003b0: f88f b361 8ec1 1971 f419 9d71 beb2 f4ca  ...a...q...q....
00003c0: 1c42 eccf 31e1 3783 3e6d bf75 3765 83a6  .B..1.7.>m.u7e..
00003d0: 41cc 5f17 c588 0436 df79 4dd9 fafd 752f  A._....6.yM...u/
00003e0: 353d 7fcc fffd 79e5 057d 7cc1 c93d 84b5  5=....y..}|..=..
00003f0: 9080 9f98 75f5 c427 c6d3 ffbb 2d55 00d0  ....u..’....-U..
0000400: 3c01 d6c7 410b 7bcd 8d7c f79e c27d 7b5c  <...A.{..|...}{\
0000410: f6dc 7047 4bd6 6e66 2ab7 84a2 2e7d 8676  ..pGK.nf*....}.v
0000420: b1fd 795b dbbd 7e58 043d 82bf 9b3d 836b  ..y[..~X.=...=.k
0000430: fbc6 0485 29f2 5213 6b02 b802 3b6a 30df  ....).R.k...;j0.
0000440: fa7d 8887 177d 4027 298e 7ba9 145b 7aed  .}...}@').{..[z.
0000450: 303d 8219 9cbd 7c5f 1cf9 36b5 b439 3dee  0=....|_..6..9=.
0000460: b63d 76d4 9bfd 7b6c bdbd 83b8 7e3d 8463  .=v...{l....~=.c
0000470: 93b0 32ab c928 2966 29aa ae16 6ec5 9ad0  ..2..()f)...n...
0000480: 067e 86bf 306d 7b87 f77d ffb8 446d 7bcf  .~..0m{..}..Dm{.
0000490: 143d 35b6 e879 39cf d7b9 5c05 79bd 571f  .=5..y9...\.y.W.
00004a0: 2cfd 8640 4f7d 80d7 bf3d 85b5 7d7d 7e35  ,..@O}...=..}}~5
00004b0: ef2e 8255 95e1 8361 6086 946e e1ce 3da9  ...U...a`..n..=.
00004c0: e88c eab7 23f1 0da3 261b 7baf ce35 6bae  ....#...&.{..5k.
00004d0: 2f39 e040 12a1 a732 463d 693f d915 7566  /9.@...2F=i?..uf
00004e0: bfbd 7d9d 853d 7bee f6bd 7d1e 1e3d 7afe  ..}..={...}..=z.
00004f0: 8ecb 8c22 62eb 7e25 7d3d fbc1 0f75 350d  ..."b.~%}=...u5.
```

```
0000500: d281 c797 9775 6000 77df 9f95 3737 7fbb  .....u‘.w...77..
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?


Plain
Birthday
P3Graph