Most Recent Results on SHA-1

Christian Rechberger

Hash&Stream, Salzburg, 2007/02/01

Institute for Applied Information Processing and Communications (IAIK) - Krypto Group

Faculty of Computer Science
Graz University of Technology
Agenda

- Authentication using hash functions – Attacks on NMAC/HMAC-SHA-1

- New view on the problem of collision search in SHA-1
- New automated method – Results and Examples
- Extensions to (partly) meaningful collisions

- Conclusions
Message Authentication? Problem Description:

Alice → Bob

“Let’s go to the pool at 17:00”

5242832

Eve
Message Authentication? Problem Description:

“Let’s go to the pool at 12:00”

5242832
Message Authentication? Problem Description:

Alice \( k \)  \rightarrow \text{“Let’s go to the pool at 12:00”} \rightarrow 5242832 \rightarrow Bob \( k \)
Message Authentication? Problem Description:

Alice

\[ k \]

(m, MAC)

Bob

\[ k \]

\[ \text{MAC} = f(m,k) \]
Deducing the key should be infeasible

Alice

Bob

k

(m₁, MAC₁)

(m₂, MAC₂)

(m₃, MAC₃)

...

Eve
MAC ↔ Hash

- Message Authentication Codes (MACs) based on hash functions started to be popular in the mid 90s

- HMAC is the most common example
  - employs hash functions like MD5 or SHA-1
  - standardized by ANSI, ETSI, FIPS, IETF, ISO,…
  - used in many products (SSH, SSL)
Fedora Core release 3 (Heidelberg)
Linux Version 2.6.18.8ZID, Compiled #1 SMP Mon Nov 27 19:35:53 CET 2006
Four 3.6GHz Intel Pentium 4 Xeon Processors, 4GB RAM, 28826.32 Bogomips Total
Load Average 0.18, 0.09, 0.03
Uptime 25 days 21 hours 47 minutes
pluto.tugraz.at

Most Recent Results on SHA-1
NMAC

\[ m \xrightarrow{h} k_1 \xrightarrow{h} k_2 \xrightarrow{MAC} \]
HMAC

Proven secure assuming:
- h is collision resistant
- h has some pseudorandom properties
Deducing the key should be infeasible

Alice

k

(m₁, MAC₁)

(m₂, MAC₂)

(m₃, MAC₃)

...

Bob

k

Eve
IAIK Krypto Group

Most Recent Results on SHA-1

(m₁, MAC₁)
(m₂, MAC₂)
(m₃, MAC₃)
...

Attack

(h, k+p₁)

(h, m)

(h, k+p₂)

MAC
Known 
Unknown 

\[ h \]
\[ k+p_1 \]
\[ \delta \]
\[ \delta m \]
\[ k+p_2 \]
\[ ? \]

Attack
Known

Unknown

\[ \delta m \rightarrow h \]

\[ ? \]

\[ h \]

\[ ? \]

\[ ? \]

\[ ? \]
Attack

**Known**

**Unknown**

After a number of trials...

⇒ 1 bit of key information recovered

\[
\delta m \quad h \quad ? \Rightarrow 1 \text{ bit}
\]

\[
\delta 0 \quad h \quad ? \quad \delta 0
\]
**Known**

**Unknown**

**Inner key recovered...**

**Outer key?**

\[ h \]

\[ \delta_m \]

\[ h \]

\[ \delta_0 \]
Results on NMAC/HMAC

- Attacks exploit non-random properties
- Compared to unkeyed hash: less severe

- Applies to NMAC/HMAC with hash functions like MD4, MD5 and reduced SHA-1
- Theoretical attacks for up to 61 steps of NMAC-SHA-1

- Some security margin left

- To be presented at Financial Cryptography 2007 (joint work with Vincent Rijmen)
Agenda

• Authentication using hash functions – Attacks on NMAC/HMAC-SHA-1
  ▪ New view on the problem of collision search in SHA-1
  ▪ New automated method – Results and Examples
  ▪ Extensions to (partly) meaningful collisions

• Conclusions
Open Problem

SHA-1 old approach

SHA-1 new approach

Most Recent Results on SHA-1
Finding Collisions as a Continuing Optimization Process

work factor

message freedom
Two key techniques of Wang et al.:
- Manually find suitable complex characteristic NL₁ and NL₂
- Advanced message modification to improve work factor

Methods are rather ad hoc (manual)
Optimization?
New View – Roughly Illustrated

- Fix Message Difference
- Fix L-characteristic
- Fix NL-characteristic
- Optimize Search

(message freedom vs. work factor graph)
Principles

- Generalized conditions

\[
\begin{array}{c|c|c}
0 & 0 & 0* \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

<table>
<thead>
<tr>
<th>Type</th>
<th>Possibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>2</td>
</tr>
<tr>
<td>Signed-bit</td>
<td>4-6</td>
</tr>
<tr>
<td>Generalized:</td>
<td>16</td>
</tr>
</tbody>
</table>
Generalized Conditions - Notation

<table>
<thead>
<tr>
<th>$(x_i, x_i^*)$</th>
<th>$(0, 0)$</th>
<th>$(1, 0)$</th>
<th>$(0, 1)$</th>
<th>$(1, 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>−</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>x</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>o</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>u</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>n</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>#</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Principles

- Generalized conditions
- Use “bit-sliced design” to efficiently
  - Propagate conditions within one step transformation
  - Propagate conditions among all step transformations
Animated Illustration
New View – Roughly Illustrated

Fix Message Difference

Fix L-characteristic

Fix NL-characteristic

Optimize Search

message freedom

1
512
1024
work factor

$2^{n/2}$
$2^n$
Principles

- Generalized conditions

- Use “bit-sliced design” to efficiently
  - Propagate conditions within one step transformation
  - Propagate conditions among all step transformations

- Continuously add more conditions to improve work factor
Most Recent Results on SHA-1

New View – Roughly Illustrated

- Fix Message Difference
- Fix L-characteristic
- Fix NL-characteristic
- Optimize Search

Work factor vs. message freedom graph:
- Work factor increases from $2^{n/2}$ to $2^n$.
- Message freedom decreases from 1024 to 512.

The graph illustrates the different levels of work factor and message freedom, with green check marks indicating the focus areas for optimization.
New View – Roughly Illustrated

- Fix Message Difference
- Fix L-characteristic
- New Unified Method

Graph with axes:
- Y-axis: message freedom (1, 512, 1024)
- X-axis: work factor ($2^{n/2}$, $2^n$)

Legend:
- Red: New Unified Method
- Green: Fix L-characteristic
- Blue: Fix Message Difference
Example: 64-step SHA-1 collision

<table>
<thead>
<tr>
<th>$i$</th>
<th>Message 1, first block</th>
<th>Message 1, second block</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>63DAEFDD 30A0D167 52EDCDA4 90012F5F</td>
<td>3B2AB4E1 AAD112EF 669C9BAE 5DEA4D14</td>
</tr>
<tr>
<td>5-8</td>
<td>0DB4DFB5 E5A3F9AB AB66EE56 12A5663F</td>
<td>1DBE220E AB46A5E0 96E2D937 F3E58B63</td>
</tr>
<tr>
<td>9-12</td>
<td>D0320F85 8505C67C 756336DA DFFF4DB9</td>
<td>BE594F1C BD63F044 50C42AA5 8B793546</td>
</tr>
<tr>
<td>13-16</td>
<td>596D6A95 0855F129 429A41B3 ED5AE1CD</td>
<td>A9B24128 816FD53A D1B663DC B615DD01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$i$</th>
<th>Message 2, first block</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>63DAEFDE 70A0D135 12EDCDE4 70012F0D</td>
</tr>
<tr>
<td>5-8</td>
<td>ADB4DFB5 65A3F9EB 8B66EE57 32A5665F</td>
</tr>
<tr>
<td>9-12</td>
<td>50320F84 C505C63E B5633699 9FFF4D9B</td>
</tr>
<tr>
<td>13-16</td>
<td>596D6A96 4855F16B 829A41F0 2D5AE1EF</td>
</tr>
</tbody>
</table>

Example 64-step 2-block colliding pair of messages

- Work factor (both blocks) was less than $2^{35}$ SHA-1 computations (1st block much faster)

- Underlying method recently presented at NIST Hash Workshop and Asiacrypt 2006 (joint work with Christophe De Cannière)
Agenda

• Authentication using hash functions – Attacks on NMAC/HMAC-SHA-1

  ▪ New view on the problem of collision search in SHA-1
  ▪ New automated method – Results and Examples
  ▪ Extensions to (partly) meaningful collisions

• Conclusions
Motivation

- Setting: Collisions for a hash function can be constructed

- Cryptanalyst perspective: Some more interesting things to find out w.r.t. collision resistance?
  - Constructing collisions faster
  - Finding and exploiting degrees of freedom to construct (partially) meaningful collisions

Practically relevant if hash function is widely deployed
Color Code

Under control, attacker can freely choose → meaningful

Not under direct control, determined by the collision search algorithm → not meaningful
Meaningful Collisions: Challenges for MD4-style Hash Functions

1. One Commonly Chosen Prefix

2. One Commonly Chosen Prefix +
   Partial Control over Colliding Blocks

3. Two Arbitrary Different Chosen Prefixes

4. Two Arbitrary Different Chosen Prefixes +
   Partial Control over Colliding Blocks
Meaningful Collisions: Challenges for MD4-style Hash Functions

1. One Commonly Chosen Prefix
2. One Commonly Chosen Prefix + Partial Control over Colliding Blocks
3. Two Arbitrary Different Chosen Prefixes
4. Two Arbitrary Different Chosen Prefixes + Partial Control over Colliding Blocks
Meaningful Collisions: Challenges for MD4-style Hash Functions

1. One Commonly Chosen Prefix
2. **One Commonly Chosen Prefix** + Partial Control over Colliding Blocks
3. Two Arbitrary Different Chosen Prefixes
4. Two Arbitrary Different Chosen Prefixes + Partial Control over Colliding Blocks
Meaningful Collisions: Challenges for MD4-style Hash Functions

1. One Commonly Chosen Prefix
2. One Commonly Chosen Prefix + Partial Control over Colliding Blocks
3. Two Arbitrary Different Chosen Prefixes
4. Two Arbitrary Different Chosen Prefixes + Partial Control over Colliding Blocks
Meaningful Collisions: Challenges for MD4-style Hash Functions

1. One Commonly Chosen Prefix
2. One Commonly Chosen Prefix + Partial Control over Colliding Blocks
3. Two Arbitrary Different Chosen Prefixes
4. Two Arbitrary Different Chosen Prefixes + Partial Control over Colliding Blocks
Meaningful Collisions: Challenges for MD4-style Hash Functions

I. One Commonly Chosen Prefix

II. One Commonly Chosen Prefix + Partial Control over Colliding Blocks

III. Two Arbitrary Different Chosen Prefixes

IV. Two Arbitrary Different Chosen Prefixes + Partial Control over Colliding Blocks

“easier”

“harder”
I. One Commonly Chosen Prefix

- Small number of colliding blocks
- Enough for colliding meaningful postscript files, etc… (see tomorrow)
Example: Collision for 64-step SHA-1

- Same Stuff
  [random not meaningful garbage]

- Same Hash

- Same Stuff
  [different random not meaningful Garbage]
II. One Commonly Chosen Prefix + Partial Control over Colliding Blocks

- Small number of colliding blocks
- Application in areas where format restrictions apply
Example: Collision for 64-step SHA-1

I hereby solemnly promise to finish my PhD thesis by the end of **2005**

[Garbage]

Same Hash

I hereby solemnly promise to finish my PhD thesis by the end of **2006**

[different Garbage]
Details (shown at Rump Session of Crypto 2006)

2nd (colliding) message:

4920 6865 7265 6279 2073 6f6c 656d 6e6c I hereby solemnly promise to finish my PhD thesis by the end of 2005............!

7920 7072 6f6d 6973 6520 746f 2069 6e y promise to finish my PhD thesis by the end of

6973 6820 6d79 2050 6844 2074 6865 7369 s by the end of

7320 6279 2074 6865 2065 6e64 206f 6620 s by the end of

3230 3035 200a 0a 029a 9f21 9821 2005 ............!

f0f0 ff92 13e4 3df4 07ca 4a69 0673 6850 .........=...Ji.shP

7f39 7c77 dddf 45c1 52ac 0ab0 9d15 11cf .9\v..E.R.........

a15f dc78 9f4d 8621 5d1d 41f3 c2a7 3c6a .._.x.M.].A...<j

c2b5 d3a1 1ebb 7dee ffc2 7fb5 5c31 535c .......

8fb1 3dce c26a 4b89 0e82 d260 8ce7 31fb ..=.jK.....1.

383b 24d9 37fb eca9 f5e3 90b6 c123 15d5 8;$.7.............#

c1a4 8abe 9ad3 c1df f6d5 50c9 6bd9 572d ........P.k.W-

Most Recent Results on SHA-1
III. Two Arbitrary Different Chosen Prefixes

- Using feed-forward operation, iteratively cancel out chaining differences with selected near-collision paths
- Usually much more than two message blocks needed
- Speedup: birthday phase before
- Example: see tomorrow
IV. Two Arbitrary Different Chosen Prefixes + Partial Control over Colliding Blocks

- Using feed-forward operation, iteratively cancel out chaining differences with selected near-collision paths

- Combination of methods
### Example Characteristic for type IV

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\nabla A_i$</th>
<th>$\nabla W_i$</th>
<th>$F_W$</th>
<th>$P_u(i)$</th>
<th>$P_c(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4:</td>
<td>0000uuuuuuu1nnu0111u1u100nn111u1nn00u1</td>
<td>n0n00000000000000000000000000n</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>-3:</td>
<td>01000nn0110nnnnunnn1nu01nu1n1nu00n0</td>
<td>0n0100010001100000110110un0011</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>-2:</td>
<td>0uu0nn10uu1nunu10111n01uu1u11nun</td>
<td>uu11111010100000000000000000</td>
<td>17</td>
<td>-14.00</td>
<td>0.00</td>
</tr>
<tr>
<td>-1:</td>
<td>uu1n1111111n11n1110101n1nu1n00101n</td>
<td>00u001001100110000011011100un10000n</td>
<td>8</td>
<td>-7.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0:</td>
<td>uu1n10uuuu0u11nn0111u0n0000u0n010001nu</td>
<td>x1u1n010010-110-011-0101u-11-10</td>
<td>8</td>
<td>-3.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>xu1n0-11------0-00-0-------x-u-u</td>
<td>20</td>
<td>-14.61</td>
<td>-0.19</td>
</tr>
<tr>
<td>2:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>3:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>4:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>5:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>6:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>7:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>8:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>9:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>10:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>11:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>12:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>13:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>14:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
<tr>
<td>15:</td>
<td>uu1n11111uuuu01nu10u10u0-1n0001n0</td>
<td>uu1010------1------1--x--u0</td>
<td>22</td>
<td>-19.00</td>
<td>-0.68</td>
</tr>
</tbody>
</table>
Agenda

• Authentication using hash functions – Attacks on NMAC/HMAC-SHA-1

  ▪ New view on the problem of collision search in SHA-1
  ▪ New automated method – Results and Examples
  ▪ Extensions to (partly) meaningful collisions

▪ Conclusions
Conclusions / Future Work

- Collision for full (80-step) SHA-1 is getting closer
- Optimization is ongoing
  - 2005: $2^{69}$
  - 2006: $2^{62} - 2^{63}$
  - Advanced techniques as used for partial meaningful collisions can also be turned into faster collision search
  - 2007: ?

- Apply to other hash functions like RIPEMD-160, SHA-2?
- More powerful attacks on NMAC/HMAC?
Most Recent Results on SHA-1

Q&A

Christian.Rechberger@iaik.tugraz.at
www.iaik.tugraz.at/aboutus/people/rechberger

Hash&Stream, Salzburg, 2007/02/01