Collision Resistant Usage of SHA-1 via Message Pre-processing

Michael Szydlo
RSA Security

Yiqun Lisa Yin
Independent Consultant
Recent Advances in Hash Collision Attacks

• Efficient collisions found for MD4, MD5
  – Improved techniques include differential, message modification approaches
  – Other hash functions affected

• Wang, Yin, Yu focus on full SHA-1 (2005)
  – Complexity of collision currently $2^{69}$
  – Compare to design goal of $2^{80}$

• Security community planning response
Standard Track Response

• Option #1: Upgrade hash function
  – Completely new hash function
  – Use SHA-256
  – Truncate to SHA-256 output to 160 bits

• Option #2: Re-design affected protocols
  – Incorporate randomness into hashing
  – Randomized Hashing (Halevi, Krawczyk)
    • \( H_r(m) = H(m \ XOR \ r||r||r\ldots r) \)
    • \( RSASign(m) = (r, RSA(r, H_r(m)) \)
Considerations

• **Upgrade Option**
  – New hash function design takes years
  – Larger output of SHA-256 inconvenient
  – Security of “Truncated SHA-256” has not been explicitly studied

• **Randomized Hashing Option**
  – Randomness is required and needs to be managed
  – Possible changes in signature size
  – Alter protocols such as PKCS#1
Message Pre-processing

• A simple message transformation
  – $M' = \_ (M)$, $\_$ is very simple function
  – New derived hash function is
    • $\text{SHApp}(m) = \text{SHA-1}(\_ (M))$

• Effects on applications
  – Prevents all known collision attacks
  – $\_$ stretches message length 33-100%
Two Candidate Transformations

• Message Whitening (word-wise)
  – \(m_1 m_2 m_3 m_4 m_5 \ldots\) becomes
  – \(m_1 m_2 \ldots m_{12} \, 0 \, 0 \, 0 \, 0 \, m_{13} m_{14} \ldots m_{24} \, 0 \, 0 \, 0 \, 0 \, m_{25} \ldots\)
  – Each block contains *whitened* words

• Message Interleaving
  – \(m_1 m_2 m_3 m_4 m_5 \ldots\) becomes
  – \(m_1 m_1 m_2 m_2 m_3 m_3 \ldots\)
  – Each block contains *duplicated* words
Implementation Options

• Pre-processing within SHA-1 Function
  – Change SHAUpdate() to SHAppUpdate()
  – New function SHAppUpdate()
    • expands m via _
    • calls usual SHAUpdate() as black box

• Pre-processing outside SHA-1 Function
  – Processing occurs first and then calls usual SHA-1 as black box

• Two options are interoperable
  – Which option is better depends on the application
Implementation and Security

Features

• Zero “API signature” change
  – Output of $\text{SHApp}(m)$ is automatically 160-bit

• Almost zero change to protocol specification
  – Only need a new algorithm identifier for $\text{SHApp}$

• Security analysis
  – Leverages on existing analysis of SHA-1
  – Effects of pre-processing techniques can be quantified
# Comparing Approaches

<table>
<thead>
<tr>
<th></th>
<th>Truncate SHA-256</th>
<th>Random Hash</th>
<th>Preprocess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash Output Truncation</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change Signature Size</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Randomness Required</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Replace SHA1 Code</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change Message before Hashing</td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td><strong>Execution Cost</strong></td>
<td>50-200%</td>
<td>(not %)</td>
<td>33-100%</td>
</tr>
<tr>
<td>(time increase)</td>
<td>Depends on SHA-256 slowdown on platform</td>
<td>Depends on random generation</td>
<td>Depends whitening parameter</td>
</tr>
</tbody>
</table>
Conclusions

• Message preprocessing is viable solution to increasing secure life of SHA-1
• Technique can also be applied to MD5
• Long term solutions involve design of new hash function from the ground up
• See paper for additional detail including security analysis
  – Submitted to NIST for inclusion in the Cryptographic Hash Workshop scheduled for 31-Oct-2005
  – Available online at: http://eprint.iacr.org/2005/248