Characterization of Padding Rules and Different Variants of MD Hash Functions

Mridul Nandi

National Institute of Standards and Technology
Outline

• Introduction to hash function and known Padding Rules.

• **Thm1**: Suffix-free Padding rule is necessary and sufficient for MD hash functions.

• **Thm2**: A new suffix-free padding rule handling arbitrary message using log |M| bits and study comparison.

• **Thm3**: The simplest $10^k$ padding rule (no length overhead) is secure on a modified MD hash or mMD.

• **Thm4**: It also works for newly introduced design mode BCM (Backward Chaining Mode) and its modification mBCM.
Introduction to Hash Function:
Security notions, applications and MD iteration and known padding rules
Hash Function

Arbitrary Length Strings → fixed length strings

SHA-1: 160 bit hash output
Security Notions

- **Collision Resistance**: Hard to find \( M \neq M' \),
  \[ H(M) = H(M'). \]

- **Preimage Resistance**: Given random \( y \), hard to find \( M \),
  \[ H(M) = y. \]

- **2nd Preimage Resistance**: Given random \( M \) hard to find \( M' \neq M \),
  \[ H(M) = H(M'). \]
Applications

- Digital Signature
- Key Exchange
- *Message Authentication : HMAC*
- Pseudorandom number generation
- Key generation
- Public Key
- Others
Design Principle of Hash Function: Merkle-Damgård ('89)

Merkle-Damgård (MD) hash with strengthen-MD (sMD) padding rule (message size is at most $2^{64}$, practically sufficient, but theoretically not).

- divide padded message into blocks:
  \[
  \text{Pad}(M) = M \| 10...00 \| \text{binary}(|M|)_64
  \]

\[
\begin{array}{c|c|c|c|c}
M_1 & M_2 & \ldots & M_{t-1} & M_t \\
\end{array}
\]

\[
\text{IV} \quad \rightarrow \quad f \quad \rightarrow \quad f \quad \rightarrow \quad \ldots \quad \rightarrow \quad f \quad \rightarrow \quad f \quad \rightarrow \quad \text{hash value}
\]

- d-bit
- n-bit
Design Principle of Hash Function: Merkle-Damgård ('89)

- pad : \{0,1\}^{*} or \{0,1\}^{<L} \rightarrow \{0,1\}^{d}
- Characterize padding rule preserving different security properties

\[ \text{hash value} \]

\[ \text{MD}^f(M_1, \ldots, M_t) = \text{MD}^f(\text{pad}(M)) \]
Types of Padding Rules

- **Suffix-free**: A padding rule is called suffix-free if for any distinct $M$ and $M'$, $\text{pad}(M)$ is not suffix of $\text{pad}(M')$, i.e. there does not exist any $Z$ such that
  \[ \text{pad}(M') = Z || \text{pad}(M). \]

- **Prefix-free**: It is usually important for PRF-security.

- **Injective**: If not injective then clearly it is not collision resistant.
Known Padding Rules

- **sMD padding rule (Merkle’89):** $M \rightarrow M \| 10^k \| \text{binary}(M)_{64}$
  - Originally proposed by Merkle in Crypto 89.
  - Message length should be less than $2^{64}$ (length padding).
  - Suffix-free.

- **Damgård Padding Rule (Crypto 89):** Given $M$, divide $M\|0^k$ it into several blocks $M_1 \| ... \| M_t$ where $|M_i| = d-1$.
  - $\text{Pad}(M) = M_1 1 \| M_2 0 \| ... \| M_t 0 \| \text{binary}(|M|)_d$
  - Suffix-free.
  - Linear number of padding bits, applied to arbitrary message.
Sarkar’s Padding Rule:
- $\log(|M|) \log^*(|M|)$ padding bits. More than Merkle’s padding bits but defined for arbitrary messages.
- Asymptotically padding size is less than Damgård’s padding rule.
- Is there any good padding rule with less padded bits?

Good padding rules are those which preserve security notions. e.g. if $f$ is collision resistant then $MD_f^{pad()}$ is also collision resistant (similarly for other security notions) or design modes other than MD.
Known Padding Rules are Good

- Merkle-Damgård (Crypto-89) Construction with sMD or Damgård padding rule or Sarkar’s padding rule is good i.e. preserve security requirements.

\[
\begin{array}{c|c|c|c|c}
M_1 & M_2 & \ldots & M_{t-1} & M_t \\
\hline
\end{array}
\]

\begin{align*}
IV & \xrightarrow{\text{n-bit}} f & \xrightarrow{\text{n-bit}} f & \ldots & \xrightarrow{\text{n-bit}} f & \xrightarrow{\text{n-bit}} \text{hash value} \\
& \quad \text{d-bit} & \quad \text{d-bit} & & \quad \text{d-bit} & \quad \text{d-bit} \\
& \quad \text{f is CR} & \quad \text{H is CR.} & & \quad \text{f is PR} & \quad \text{H is PR}
\end{align*}
Known Padding Rules are Good

- Merkle-Damgård (Crypto-89) Construction with
  sMD or Damgård padding rule or Sarkar’s padding
  rule is good i.e. preserve security requirements.

\[
\begin{array}{cccc}
  M_1 & M_2 & \ldots & M_{t-1} & M_t \\
\end{array}
\]

\[\text{IV} \quad \text{n-bit} \quad f \quad \text{d-bit} \quad \text{n-bit} \quad f \quad \text{n-bit} \quad \ldots \quad \text{n-bit} \quad f \quad \text{n-bit} \quad f \quad \text{n-bit} \quad \text{hash value}\]

\[f \text{ is } 2^{\text{nd}} \text{ PR} \quad \leftrightarrow \quad H \text{ is } 2^{\text{nd}} \text{ PR (?) (open problem)}\]
<table>
<thead>
<tr>
<th></th>
<th>sMD</th>
<th>Damgård</th>
<th>Sarkar</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limit of message size</strong></td>
<td>&lt; $2^{64}$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td><strong>Padding bits</strong>¹</td>
<td>64</td>
<td>$O(</td>
<td>M</td>
</tr>
<tr>
<td><strong>Length overhead</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Collision security</strong></td>
<td>n-bit²</td>
<td>n-bit</td>
<td>n-bit</td>
</tr>
<tr>
<td><strong>(2nd) PI security</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

¹ We ignore the $10^k$ padding while counting padding bits, as it is must for every hash function to make a incomplete message block into complete.

² If n-bit compression function $f$ is collision resistant then so is hash function
Suffix-free Padding Rule is Necessary and Sufficient
Characterizing Padding Rules for MD iteration

• Is length padding necessary?
• Which property is necessary and sufficient to preserve collision resistance?
• Is there any more efficient padding rule than sMD (strengthen-MD) that applies to arbitrary message?
• We answer the above questions now.
A Sufficient condition for preserving collision security

Theorem 1a: (suffix-free is sufficient condition)

If the padding rule function $\text{pad}(.)$ is suffix-free and the compression function $f$ is collision resistant then the hash function $\text{MD}^f(\text{pad}(.))$ is collision resistant.

• The result is intuitive but no written proof is found.

• Intuitive and not difficult to prove. Similar to backward induction applied in sMD padding rule.
A necessary condition for preserving collision security

Theorem 1b: (suffix-free is necessary condition)

If the padding rule pad is not suffix-free then there exists a collision resistant compression function $f(.)$ such that the hash function $\text{MD}^f(p\text{ad}(.))$ is not collision resistant.

• In the paper, a counter example of $f(.)$ is given.

• So to have collision preserving property we need suffix-free padding rule when we use MD design mode.
New suffix-free Padding Rule:
more efficient than sMD
and applied to arbitrary message
A New Padding Rule

• Let $L = \text{binary}(|M|)$ and $L10^{k'} = L_1 || L_2 || ... || L_{t'}$ where $|L_i| = w-1$ for smallest $k'$.

• Let $P = 0L_1 || 1L_2 || ... || 1L_{t'}$ (Damgård padding rule applied to $L$, if $t' = 1$ the first bit is 0).

• $\text{pad1}(M) = M10^k || P$, for smallest $k$. 
A New Padding Rule

• \( w \) is a parameter, e.g.
  - \( w = 8, 32, 64, 128 \) etc.

**Theorem 2:** (The new padding rule \( \text{pad1} \) is suffix-free)

The padding rule \( \text{pad1}(.) \) defined above is suffix-free, requiring at most \((1 + 1/(w-1)) \log |M|\) many bits.
The New Padding Rule pad1

- Applied to arbitrary message i.e. with message space \{0,1\}^*.

- Only $\log|M|$ padding bits.

- For $w = 64$, It is identical with sMD padding rule when applied to message of size less than $2^{63}$ (practically, no change is necessary).

- Short messages with small $w$ require less padding bits than sMD and hence (possibly) one less invocation of compression function.
<table>
<thead>
<tr>
<th></th>
<th>sMD</th>
<th>Damgård</th>
<th>Sarkar</th>
<th>pad1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message space</strong></td>
<td>&lt; $2^{64}$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td><strong>Padding bits</strong></td>
<td>64</td>
<td>$O(</td>
<td>M</td>
<td>)$</td>
</tr>
<tr>
<td><strong>Length overhead</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Collision security</strong></td>
<td>n-bit</td>
<td>n-bit</td>
<td>n-bit</td>
<td>n-bit</td>
</tr>
<tr>
<td><strong>(2nd) PI security</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Note that $\log|M|$ can be smaller than 64 for suitable choice of $w$ and for short messages.
Modified MD or mMD without length overhead:
Modified MD (or mMD) without length padding

- Modified MD with simple padding rule \( \text{pad}(M) = M \mathbin{||} 10^k \) call it \( 10^k \) padding rule (the minimum padding rule to make a complete block)

\[ \begin{array}{cccccc}
M_1 & M_2 & \ldots & M_{t-1} & M_t & 10^k \\
\end{array} \]

- It is nothing but MD with the compression function \( f' \). But the padding rule is not suffix-free. However, we can prove that the hash function is collision resistant if \( f \) is collision resistant and it is hard to find two preimages of \( 0^n \) and \( 0^{n-1} \).
**Modified MD (or mMD) without length padding**

- Modified MD with simple padding rule $\text{pad}(M) = M \| 10^k$ call it $10^k$ padding rule (the minimum padding rule to make a complete block)

```
\begin{array}{cccc}
M_1 & M_2 & \ldots & M_{t-1} & M_t 10^k \\
\end{array}
```

- If the chain value is $\text{IV} \| 0$ change it to $\text{IV} \| 1$
- o.w. do nothing

If the chain value is $\text{IV} \| 0$ change it to $\text{IV} \| 1$
- o.w. do nothing
Modified MD (or mMD) without length padding

- No overhead due to length.
- One extra “if” checking for each message block.
- Security Guarantee: The hash function is collision resistant if the underlying compression function $f$ is collision resistant and it is hard to find two different inputs $X, X'$ such that $f(X) = 0^n$ and $f(X') = 0^{n-1}1$. We call strongly collision resistant.
Backward Chaining Mode (BCM)

- $K_1$, $K_2$, $K_3$ (2n+d) bits public random strings like salt.

- Salt size is larger than usual.

$M^*_t$ denotes the last padded block.
Strongly Collision Resistant

- It is automatically guaranteed from \((n-1)\)-bit collision security of \(f\). It seems to be a weaker assumption as we have to find two different messages whose first \((n-1)\)-bit is zero (not any arbitrary value).
Modified MD (or mMD) without length padding

**Theorem 3:** The modified MD is collision resistant if the underlying compression function $f$ is strongly collision resistant.
Backward Chaining Mode and its efficient modification
Backward Chaining Mode (BCM)

- Introduced in SAC 08 by Andreeva and Preneel.

- MD hash is not known to preserve 2nd preimage (2PI).

- BCM preserves second preimage along with preimage and collision.

- We propose a modification and the efficient padding rule with the variant used in MD are applicable here.
More Efficient BCM or mBCM

• Drop $K_1$ and $K_2$. So, only n-bit salt!

• Same tweak applies here as we did for MD (change chain value $0^n$ to $0^{n-1}$1, if there is any such chain value). Call it $f'$.
mBCM preserves security notions

**Theorem 4**: The modified BCM or mBCM is collision resistant if the underlying compression function $f$ is strongly collision resistant. Moreover, if $f$ is $(2^{nd})$ preimage resistant then so is mBCM for a message space of all messages of size at least $(d+n)$. 
<table>
<thead>
<tr>
<th></th>
<th>sMD+ MD</th>
<th>sMD +BCM</th>
<th>mMD</th>
<th>mBCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding bits</td>
<td>64&lt;sup&gt;1&lt;/sup&gt;</td>
<td>64&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0&lt;sup&gt;1&lt;/sup&gt;</td>
<td>0&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>Length overhead</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Salt-size</td>
<td>0</td>
<td>2n+d</td>
<td>0</td>
<td>n</td>
</tr>
<tr>
<td>Collision security</td>
<td>n-bit</td>
<td>n-bit</td>
<td>(n-1)-bit</td>
<td>(n-1)-bit</td>
</tr>
<tr>
<td>(2nd) PI security</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<sup>1</sup> We ignore the 10<sup>k</sup> padding while counting padding bits, as it is must for every hash function to make a incomplete message block into complete.
Conclusion and Future Works

• Suffix-free padding rule for MD iteration is necessary and sufficient to prove the collision preserving property. We provide an example of suffix-free padding rule padding \(\log|M|\) bits for any arbitrary message.
  - Open problem: To prove \(\log|M|\) is the minimum or to find lower bound of padding bits for suffix-free padding rules.

• A simple modification of MD without length padding is collision resistant. We also modify the BCM. With the similar tweak applied to the modified BCM
  - preserves all three basic security requirements
  - applied to arbitrary messages
  - no length padding is required.
  - Open problem: Is there any design with the \(10^k\)-padding preserving collision security?
Thank You for your attention