Improved Fast Syndrome Based Cryptographic Hash Functions

Matthieu Finiasz, Philippe Gaborit, and Nicolas Sendrier
The Original FSB Hash Function
[Augot, Finiasz, Sendrier - Mycrypt 05]

- Based on the Merkle-Damgård construction
  - requires a collision resistant compression function.

- Provably secure:
  - collision search on the compression function requires to solve an instance of an NP-complete problem,
  - inversion too.

- These problems have been well studied
  - similar to those of the McEliece cryptosystem.
The core of the function is a binary $r \times n$ matrix $\mathcal{H}$.

- the input (data + chaining) is converted into a binary vector of weight $w$ and length $n$.
- this vector is multiplied by $\mathcal{H}$ to obtain $r$ bits of output.
The core of the function is a binary $r \times n$ matrix $\mathcal{H}$.

- the input (data + chaining) is converted into a binary vector of weight $w$ and length $n$.
- this vector is multiplied by $\mathcal{H}$ to obtain $r$ bits of output.

Constant weight encoding uses regular words

\[
\begin{array}{cccccccccccc}
10 & - & 0 & 0-010-0 & 0 & - & 01 & 0-010-0 & 010 & - & 0 \\
\end{array}
\]

- much faster than optimal encoding.
The Original FSB Hash Function

Theoretical security

► Inversion:
  ▶ find a vector of weight $w$ with given image
    → exactly Syndrome Decoding.

► Collision search:
  ▶ find a vector of weight $\leq 2w$ with null image
    → again Syndrome Decoding.

► With regular words, both of these problems are still NP-complete. [Augot, Finiasz, Sendrier - Mycrypt 05]
The best attack uses Wagner’s generalized birthday technique [Crypto 2002].

We look for $2w$ columns of $H$, XORing to 0.

- **Birthday technique:**
  - build 2 lists of XORs of $w$ columns.
  - complexity: $O \left( 2^{r/2} \right)$.

- **Wagner’s generalized birthday technique:**
  - build $2^a$ lists of XORs of $\frac{w}{2^{a-1}}$ columns.
  - complexity: $O \left( 2^{r/(a+1)} \right)$.
Wagner’s Generalized Birthday Technique

$L_i$ are lists of $2^{r/4}$ elements

$\triangleright$ each element is the XOR of $\frac{w}{4}$ columns.
Wagner’s Generalized Birthday Technique

\[ a = 3 \]

\[ L_1 \quad L_2 \quad L_3 \quad L_4 \quad L_8 \]

\[ \sim 1 \text{ collision} \]

\[ 0 \quad 0 \quad 0 \]

\[ L'_i \] are lists of \( 2^{\frac{r}{4}} \) elements

\( \triangleright \) each element is the XOR of \( \frac{w}{2} \) columns.

\( \triangleright \) each element starts with \( \frac{r}{4} \) zeroes.
Wagner’s Generalized Birthday Technique

\[ L_1 \quad L_2 \quad L_3 \quad L_4 \quad L_8 \]

\[ L_1 \quad L_2 \quad L_4 \]

\[ \sim 1 \text{ collision} \]

- \( L''_i \) are lists of \( 2^{\frac{r}{4}} \) elements
  - each element is the XOR of \( w \) columns.
  - each element starts with \( \frac{r}{2} \) zeroes.
Efficient parameters always allow to choose $a = 4$ in Wagner’s technique,

- for a security of $2^{80}$ we need $r = 400$.

The choice of $w$ and $n$ is flexible:

- tradeoff between the matrix size and the hash speed.

Example parameters:

$$r = 400, \ w = 85, \ n = 256 \times w = 21760.$$ 

→ speed: 70Mbits/s, matrix size: 1MB.
The Original FSB Hash Function
Conclusions and drawbacks

The original FSB construction is:
▶ practical,
▶ quite fast,
▶ provably collision resistant.

However it suffers from a few drawbacks:
▶ the output size is too large,
▶ the block size is quite large,
▶ the matrix is large,
   → does not fit in a CPU cache.
Improvements to the Original FSB
For a security against collision of $2^\lambda$ operations, one expects a hash of $2\lambda$ bits:
- requires to add a final compression round.

Used in many other constructions.
- If the final compression is collision resistant, then the combination is also collision resistant.
- What about provable security?
  - Must the last round be provably collision resistant?
- Use the same construction with other parameters?
Suppose we used a linear transform $L$ from $r$ to $r'$ bits:

- compute $H' = L \times H$ and use Wagner’s attack on $H'$.

$\rightarrow$ The complexity of decreases to $2^{r'/\alpha+1}$.

If the final transform is non-linear this won’t be possible.

We propose to use another hash function like Whirlpool:

- it is designed to be as much as possible non-linear,
- we loose provable security,
- chances are that attacks on Whirlpool won’t affect our construction.
Use of a Quasi-cyclic Matrix

Basic idea

- The matrix $\mathcal{H}$ is too large:
  - store a small amount of data and generate $\mathcal{H}$ from it,
  - must fit in the CPU cache
    - generation is done at runtime.

- Use a quasi-cyclic (QC) matrix:

$$
\mathcal{H} = \begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 
\end{bmatrix}
$$
Use of a Quasi-cyclic Matrix

Basic idea

- The matrix $\mathcal{H}$ is too large:
  - store a small amount of data and generate $\mathcal{H}$ from it,
  - must fit in the CPU cache
    - generation is done at runtime.

- Use a quasi-cyclic (QC) matrix:
  - storing the first line is enough,
  - other lines are blockwise cyclic shifts,
  - cyclic shifts can be efficient
    - no need to rebuild $\mathcal{H}$ completely before hashing.
Syndrome Decoding of a QC matrix is NP-complete.
▷ not proven for regular words.

QC codes have been extensively studied:
▷ no known efficient decoding algorithm,
▷ any attack would yield such a decoding algorithm.

For some specific sizes the outputs are proven to be uniformly distributed.

From a practical point of view:
▷ no clue how to improve Wagner’s birthday technique.
### Implementation

<table>
<thead>
<tr>
<th>secu.</th>
<th>$r$</th>
<th>$w$</th>
<th>$n$</th>
<th>$\frac{n}{w}$</th>
<th>size of $\mathcal{H}$</th>
<th>time</th>
<th>cyc./byte</th>
<th>size of $\mathcal{H}$</th>
<th>time</th>
<th>cyc./byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>512</td>
<td>512</td>
<td>131 072</td>
<td>256</td>
<td>8 388 608</td>
<td>28.8s</td>
<td>390.6</td>
<td>16 384</td>
<td>6.6s</td>
<td>89.3</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>450</td>
<td>2 304 000</td>
<td>512</td>
<td>14 745 600</td>
<td>43.1s</td>
<td>587.9</td>
<td>28 800</td>
<td>12.1s</td>
<td>165.1</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>217</td>
<td>2 25</td>
<td>256</td>
<td>2 32</td>
<td>–</td>
<td>–</td>
<td>4 194 304</td>
<td>25.0s</td>
<td>339.8</td>
</tr>
<tr>
<td>80</td>
<td>512</td>
<td>170</td>
<td>43 520</td>
<td>256</td>
<td>2 785 280</td>
<td>37.7s</td>
<td>517.0</td>
<td>5 440</td>
<td>20.5s</td>
<td>281.1</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>144</td>
<td>73 728</td>
<td>512</td>
<td>4 718 592</td>
<td>42.6s</td>
<td>581.6</td>
<td>9 216</td>
<td>17.6s</td>
<td>239.8</td>
</tr>
<tr>
<td>128</td>
<td>1 024</td>
<td>1 024</td>
<td>262 144</td>
<td>256</td>
<td>3 355 432</td>
<td>48.6s</td>
<td>669.6</td>
<td>32 768</td>
<td>8.9s</td>
<td>121.0</td>
</tr>
<tr>
<td></td>
<td>1 024</td>
<td>904</td>
<td>462 848</td>
<td>512</td>
<td>5 924 544</td>
<td>72.4s</td>
<td>989.9</td>
<td>57 856</td>
<td>27.2s</td>
<td>371.2</td>
</tr>
<tr>
<td></td>
<td>1 024</td>
<td>816</td>
<td>835 584</td>
<td>1 024</td>
<td>1 069 547 752</td>
<td>53.4s</td>
<td>727.6</td>
<td>104 448</td>
<td>11.8s</td>
<td>162.6</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td>MD5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>best known implementations from [Nakajima, Matsui - Eucrocrpyt 2002]</td>
<td></td>
<td>3.7</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td>SHA-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[Nakajima, Matsui - Eucrocrpyt 2002]</td>
<td></td>
<td>8.3</td>
</tr>
<tr>
<td>128</td>
<td></td>
<td></td>
<td>SHA-256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[Nakajima, Matsui - Eucrocrpyt 2002]</td>
<td></td>
<td>20.6</td>
</tr>
</tbody>
</table>

- Our implementation is not optimised:
  - we obtain a speed of 180Mibts/s with 128 bits security.
We propose a new variant of the FSB hash function:

- no large matrix to handle,
- standard output size,
- twice as fast as the original construction,
- not completely proven to be collision resistant:
  - use of regular words,
  - use of the final compression transform.