Trivial Collisions for Simplified and Reduced SHA-2

Sebastiaan Indesteege**

sebastiaan.indesteege@esat.kuleuven.be

Abstract. In this report we describe a trivial method to find collisions for strongly simplified variants of the SHA-2 family of hash functions. The simplifications include the linearization of the message expansion by replacing modular addition with XOR, and the reduction of the number of steps to 24 out of 64 (or 80). These simplifications allow to find collision pairs for any digest length with an expected time complexity of just \(2^8\) compression function evaluations. The same method can be applied to 30 steps of SHA-1, where the expected workload is about \(2^5\) compression function evaluations.

Key words: hash functions, collisions, SHA-2

1 Introduction

This report describes a collision finding attack on simplified variants of the SHA-2 family of hash functions.

The SHA-2 Family. The SHA-2 family consists of several iterated cryptographic hash functions with different digest sizes built on similar compression functions, e.g., SHA-256, SHA-384 and SHA-512. For a complete specification of the SHA-2 hash functions, we refer to [4]. Because of the similarity between the members of the SHA-2 family, we will focus on SHA-256 here. All the results in this paper can be equally applied to the other members of the SHA-2 family.

Brief Summary of Related Work. Gilbert and Handschuh [2] showed a 9 step local collision for SHA-256 with probability \(2^{-66}\). This was improved by Hawkes et al. [1] to \(2^{-39}\). In [6], a variant of SHA-256 where all modular additions are replaced by XOR was studied, resulting in pseudo-collisions for 34 steps of this variant. Mendel et al. [3] reported collision producing characteristics for step-reduced, but otherwise unmodified SHA-256, for up to 18 steps.

* This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 ECRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

** F.W.O. Research Assistant, Fund for Scientific Research — Flanders (Belgium).
2 Description of SHA-256

The compression function of SHA-256 consists of a message expansion, which transforms a 512-bit message block into 64 expanded message words $W_i$ of 32 bits each, and a state update transformation. The latter updates eight 32-bit state variables $A, \ldots, H$ in 64 identical steps, each using one expanded message word. The message expansion can be defined recursively as follows.

$$W_i = \begin{cases} M_i & \text{for } 0 \leq i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & \text{for } 16 \leq i < 64 \end{cases} \quad (1)$$

The functions $\sigma_0(x)$ and $\sigma_1(x)$ are given by

$$\sigma_0(x) = (x \gg 7) \oplus (x \gg 18) \oplus (x \gg 3) ,$$
$$\sigma_1(x) = (x \gg 17) \oplus (x \gg 19) \oplus (x \gg 10) . \quad (2)$$

The state update transformation updates two of the state variables in every step. It uses the bitwise Boolean functions $f_{\text{IF}}$ and $f_{\text{MAJ}}$ as well as the GF(2)-linear functions

$$\Sigma_0(x) = (x \gg 2) \oplus (x \gg 13) \oplus (x \gg 22) ,$$
$$\Sigma_1(x) = (x \gg 6) \oplus (x \gg 11) \oplus (x \gg 25) . \quad (3)$$

The following equations describe the state update transformation, where $K_i$ is a step constant.

$$T_1 = H_i + \Sigma_1(E_i) + f_{\text{IF}}(E_i, F_i, G_i) + K_i + W_i ,$$
$$T_2 = \Sigma_0(A_i) + f_{\text{MAJ}}(A_i, B_i, C_i) ,$$
$$A_{i+1} = T_1 + T_2 , \quad B_{i+1} = A_i , \quad C_{i+1} = B_i , \quad D_{i+1} = C_i ,$$
$$E_{i+1} = D_i + T_1 , \quad F_{i+1} = E_i , \quad G_{i+1} = F_i , \quad H_{i+1} = G_i . \quad (4)$$

After 64 rounds, the initial state variables are fed forward using word-wise addition modulo $2^{32}$.

2.1 A Simplified Variant of SHA-256

The simplified variant studied in this article differs from the real SHA-256 in two ways. First, all additions modulo $2^{32}$ in the message expansion are replaced by XORs, making it GF(2)-linear. Second, the number of steps is reduced to 24. Hence, the simplified message expansion becomes

$$W_i = \begin{cases} M_i & \text{for } 0 \leq i < 16 \\ \sigma_1(W_{i-2}) \oplus W_{i-7} \oplus \sigma_0(W_{i-15}) \oplus W_{i-16} & \text{for } 16 \leq i < 24 \end{cases} \quad (5)$$

We will refer to this variant as SHA-256-XOR-24. The same simplifications can be applied to the other members of the SHA-2 family.
3 Finding Collisions

In this section we describe how collisions for SHA-256-XOR-24 (and other simplified SHA-2 members like SHA-384-XOR-24 and SHA-512-XOR-24) can be found using a technique that is very similar to single-message modification. Single-message modification was first introduced by Wang [5] in collision attacks on MD5, SHA-0 and others.

3.1 Alternate Description of SHA-256

In SHA-0 and SHA-1, only a single state variable is updated in every step. This naturally leads to a description where only the first state variable is considered. Something similar can be done with the SHA-2 hash functions, even though two state variables are updated in every step.

To accomplish this, we derive from the state update equations (4) a series of equations which express the inputs of the \( i \)-th state update transformation, \( A_i, \ldots, H_i \), as a function of only \( A_i \) through \( A_{i-7} \).

\[
\begin{align*}
B_i &= A_{i-1} , \\
C_i &= A_{i-2} , \\
D_i &= A_{i-3} , \\
E_i &= A_{i-4} + A_i - \Sigma_0(A_{i-1}) - f_{\text{MAJ}}(A_{i-1}, A_{i-2}, A_{i-3}) , \\
F_i &= A_{i-5} + A_{i-1} - \Sigma_0(A_{i-2}) - f_{\text{MAJ}}(A_{i-2}, A_{i-3}, A_{i-4}) , \\
G_i &= A_{i-6} + A_{i-2} - \Sigma_0(A_{i-3}) - f_{\text{MAJ}}(A_{i-3}, A_{i-4}, A_{i-5}) , \\
H_i &= A_{i-7} + A_{i-3} - \Sigma_0(A_{i-4}) - f_{\text{MAJ}}(A_{i-4}, A_{i-5}, A_{i-6}) .
\end{align*}
\] (6)

Substituting these into the state update transformation of SHA-256 (4) yields an alternative description requiring only a single state variable. This description can be written as

\[
A_{i+1} = F(A_i, A_{i-1}, A_{i-2}, A_{i-3}, A_{i-4}, A_{i-5}, A_{i-6}, A_{i-7}) + W_i .
\] (7)

The function \( F(\cdot) \) encapsulates (4) and (6) except for the addition of the expanded message word \( W_i \).

Ignoring the message expansion, it is easy to see that control over eight consecutive expanded message words allows for any difference in the state variables to be eliminated. This is very similar to the idea of single message modification [5]. Note that this description of SHA-256 is only interesting for analysis purposes, not for implementation.

3.2 Inserting Odd Additive Differences.

Consider a pair of 32-bit words \( \langle X, X' \rangle \) having an XOR difference of \( \text{0xffffffff} \), i.e., there is a difference in every bit. What are the possible additive differences that can be achieved by such a pair?
From two’s complement arithmetic, we know that the additive inverse of a word $X$ can be found as $-X = \bar{X} + 1$, where $\bar{X}$ is the one’s complement of $X$. The additive difference of the pair $(X, X')$ is
\[
\delta X = X' - X = \bar{X} - X = \bar{X} + (\bar{X} + 1) = 2\bar{X} + 1.
\] (8)
Hence, any odd additive difference can be generated by an appropriate choice of the word $X$. There are exactly two possible choices for $X$ for each odd additive difference as the most significant bit of $X$ does not influence the difference $\delta X$.

3.3 The Message Difference
For SHA-256-XOR-24, there exists a message difference that will produce a difference of $0xffffffff$ in the eight expanded message words $W_8$ through $W_{15}$ and a zero difference in $W_{16}$ through $W_{23}$. Indeed, since the message expansion of this simplified SHA-256 variant is GF(2)-linear one can simply use (5) to determine the differences in the first eight expanded message words. Table 1 shows the expanded message difference.

| $\Delta W_0$ | $\Delta W_1$ | $\Delta W_2$ | $\Delta W_3$ | $\Delta W_4$ | $\Delta W_5$ | $\Delta W_6$ | $\Delta W_7$ | $\Delta W_8$ | $\Delta W_9$ | $\Delta W_{10}$ | $\Delta W_{11}$ | $\Delta W_{12}$ | $\Delta W_{13}$ | $\Delta W_{14}$ | $\Delta W_{15}$ | $\Delta W_{16}$ |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0x56c4b38b  | 0x1e01bbe2 | 0x71721c34 | 0x037ab391 | 0x0c28f460 | 0xefbc47ff | 0xfdc03800 | 0x1fffffff | 0xffffffff | 0xffffffff | 0xffffffff | 0xffffffff | 0xffffffff | 0xffffffff | 0xffffffff | 0xffffffff |

3.4 The Collision Search
Putting the pieces together yields a simple collision finding algorithm for SHA256-XOR-24. The algorithm proceeds as follows:

1. Choose the message words $W_0$ through $W_7$ arbitrarily. Imposing the message difference from Tbl. 1 allows to compute $W'_0$ through $W'_7$.
2. For each $i$, $8 \leq i < 16$.
   (a) Determine the additive difference that needs to be introduced in step $i$ to ensure a zero difference in $A_{i+1}$, i.e.,
\[
\delta W_i = W'_i - W_i = F(A_i, A_{i-1}, A_{i-2}, A_{i-3}, A_{i-4}, A_{i-5}, A_{i-6}, A_{i-7}) - F(A'_i, A'_{i-1}, A'_{i-2}, A'_{i-3}, A'_{i-4}, A'_{i-5}, A'_{i-6}, A'_{i-7}) .
\] (9)
(b) With probability $1/2$ the difference $\delta W_i$ is odd. In this case, there are two suitable choices for $W_i$. In case $\delta W_i$ is even, we start over with a different choice of $W_0$ through $W_7$.

3. After 16 steps, a zero difference is reached in the internal state. Because the message difference is zero in the next eight rounds, this zero difference is maintained with certainty for an additional eight rounds.

Under reasonable independence assumptions, the overall success probability is $2^{-8}$, hence we expect to find a collision pair after about $2^8$ attempts. An early abort strategy and not fully backtracking slightly improves this. An example collision pair is given in Tbl. 2.

Note that a very similar approach can be applied to 30 steps of SHA-1.

4 Conclusion

We described a trivial method to find collisions for strongly simplified variants of the SHA-2 family of hash functions. Linearizing the message expansion by replacing modular addition with XOR and reducing the number of steps to 24 allows to find collision pairs with an expected effort of just $2^8$ compression function evaluations, for any digest length.

References

Table 2. An example collision pair for SHA-256-XOR-24.

<table>
<thead>
<tr>
<th>$A_i$</th>
<th>$B_i$</th>
<th>$C_i$</th>
<th>$D_i$</th>
<th>$E_i$</th>
<th>$F_i$</th>
<th>$G_i$</th>
<th>$H_i$</th>
<th>$W_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**H**

H7 8a70f1 1396ded9 64fde11e e61a8c87 693fd26d ff84c749 213bc4c3 e3b35b1e