IMPROVED SHA-1 WITH EFFICIENT COLLISION RESISTANCE

1 Medury Sri Naga Sai Abhijith, 2K. Satish Sharma, 3 G. Raja Vikram
1,2Student, Department of CSE, GITAM Institute of Technology, Visakhapatnam, Andhra Pradesh, India,
3Assistant. Professor, Department of CSE, GITAM Institute of Technology, Visakhapatnam, , Andhra Pradesh ,India.

Abstract: This Paper has a twofold relevance. In this endeavor, we aim to bring out the essence of SHA-1, a cryptographic hash function designed by the United States National Security Agency. It produces a 160-bit (20-byte) hash value and is extensively used for maintaining data integrity and also digital signatures. There have been several attempts to attack the algorithm, one among which is [3]. We provide a brief overview of their method and its significance. Attempts to find collisions for SHA-1 were successful mainly because of the linearity present in its message expansion algorithm. We suggest a design alternative to the algorithm, a method using which the message expansion becomes sufficiently random and minimize the probability of making a successful collision, theoretically.

Keywords: SHA-1, Collision, Differential Path, Distribution vector and hamming weight, Compression Function, Message Expansion.

I. INTRODUCTION

SHA-1 or "Secure Hash Algorithm" is a hash function issued by NIST in 1995 as a Federal Information processing standard. Since its publication, SHA-1 has been adopted by many government and industry security standards that require a collision resistant hash function. In addition, it has been deployed as an important component in various cryptographic schemes and protocols, such as user authentication, key agreement etc. The hash Function SHA-1 takes a message of length less than $2^{64}$ bits and produces a 160-bit hash value. The input message is padded and then processed in 512-bit blocks in the Damgard/Merkel iterative structure. Each iteration involves so called compression function which takes a 160 bit chaining value and a 512-bit message block and outputs another 160 bit chaining value. The initial chaining value is a set of fixed constants. And the final chaining value is the hash of the message.

The Compression function is an integral part of the algorithm. For each 512 bit block of the padded message, divide it into sixteen 32-bit words ($m_0, m_1, \ldots, m_{15}$). The message words are first expanded as follows: for $i = 16, \ldots, 79$.

\[ m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) \ll 1 \]

The expanded message words are then processed in four rounds, each consisting of 20 steps. The step function is defined as follows.

\[ a_i = (a_{i-1} \ll 5) + f(b_{i-1}, c_{i-1}, d_{i-1}) + e_{i-1} + m_i + k_i \]
\[ b_i = a_{i-1} \]
\[ c_i = b_{i-1} \ll 30 \]
\[ d_i = c_{i-1} \]
\[ e_i = d_{i-1} \]

For $i = 1, 2, \ldots, 80$

The initial chaining value IV = $(a_0, b_0, c_0, d_0, e_0)$ is defined as: (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476, 0xc3d2e1f0). Each round employs a different Boolean function $f_i$ and constant $k_i$, which is summarized in Table1 below.

<table>
<thead>
<tr>
<th>Round</th>
<th>Step</th>
<th>Boolean Function $F_i$</th>
<th>Constant $K_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-20</td>
<td>IF: $(X \land Y) \lor \neg(X \land Z)$</td>
<td>0x5a827999</td>
</tr>
<tr>
<td>2</td>
<td>21-40</td>
<td>XOR: $X \lor Y \lor Z$</td>
<td>0x6ed6eba1</td>
</tr>
<tr>
<td>3</td>
<td>41-60</td>
<td>MAJ: $(X \land Y) \lor (X \land Z) \lor (Y \land Z)$</td>
<td>0x8fabbbe6</td>
</tr>
<tr>
<td>4</td>
<td>61-80</td>
<td>XOR: $X \lor Y \lor Z$</td>
<td>0xea62c1d6</td>
</tr>
</tbody>
</table>

DIFFERENTIAL CRYPTANALYSIS

Differential cryptanalysis is a general form of cryptanalysis applicable primarily to block ciphers, but also to stream ciphers and cryptographic hash functions. In the broadest sense, it is the study of how differences in an input can affect the resultant difference at the output. The basic method uses pairs of plaintext related by a constant difference; difference can be defined in several ways, but the exclusive operation is usual. The attacker then computes the differences of the corresponding cipher texts, hoping to detect statistical patterns in their distribution. The resulting pair of differences is called a differential.

An analysis of the algorithm's internals is undertaken; the standard method is to trace a path of highly probable differences through the various stages of encryption, termed a differential characteristic.
II. FINDING COLLISIONS IN SHA-1

Heed [3], presents new collision search attacks on SHA-1 and introduces a set of strategies and corresponding techniques that can be used to remove some major obstacles in collision search for SHA-1. It presents new collision search attacks on SHA-1 and introduces a set of strategies and corresponding techniques that can be used to remove some major obstacles in collision search for SHA-1. Essentially, look for a near-collision differential path which has low Hamming weight in the “disturbance vector” where each 1-bit represents a 6-step local collision. Subsequently, adjust the differential path in the first round to another possible differential path so as to avoid impossible consecutive local collisions and truncated local collisions. Finally, transform two one-block near-collision differential paths into a two-block collision differential path with twice the search complexity. By combining these techniques, collisions of SHA-1 can be found with complexity less than $2^{69}$ hash operations. This was the first attack on the full 80-step SHA-1 with complexity less than the $2^{80}$ theoretical bound then.

To construct such a path, we need to find appropriate starting steps for the local collisions. They can be specified by an 80-bit 0-1 vector $x = (x_0, \ldots, x_{79})$ called a disturbance vector. For the 80 variables $x_i$, any 16 consecutive ones determine the rest. So there are 16 free variables to be set for a total of $2^{16}$ possibilities.

In order for the disturbance vector to lead to a possible collision, several conditions on the disturbance vectors need to be imposed, and they are discussed in details in [1][2]. These conditions also extend to SHA-1 in a straightforward way, and we summarize them in Table 2.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_i = 0$ for $i = t - 5, \ldots, t - 1$</td>
<td>To produce a collision in the last step $t$</td>
</tr>
<tr>
<td>$x_i = 0$ for $i = -5, \ldots, -1$</td>
<td>To avoid truncated local collisions in first few steps</td>
</tr>
<tr>
<td>No consecutive ones in same bit position in the first 16 variables</td>
<td>To avoid an impossible collision path due to a property of IF</td>
</tr>
</tbody>
</table>

A key idea of our new attack is to relax all the conditions on the disturbance vectors. In other words, we impose no condition on the vectors other than they satisfy the message expansion recursion. This allows us to find disturbance vectors whose Hamming weights are much lower than those used in existing attacks.

Finding multi-block collisions using near collisions effectively relax the first condition, and finding collisions for SHA-1 without the first round effectively relax the second condition.

We note that the 80 disturbance vectors $x_0, \ldots, x_{79}$ can be viewed as an 80-by-32 matrix where each entry is a single 0/1 bit. A simple observation is that for a matrix with low hamming weight, the non-zero entries are likely to concentrate in several consecutive columns of the matrix. Hence, we can first pick two entries $x_{i,j-1}$ and $x_{i,j}$ in the matrix and let two 16-bit columns starting at $x_{i,j-1}$ and $x_{i,j}$ to vary through all 232 possibilities. There are 64 choices for $i$ ($i = 0, 1, \ldots, 63$) and 32 choices for $j$ ($j = 1, 2, \ldots, 32$). In fact, with the same i, different choices of $j$ produce disturbance vectors that are rotations of each other, which would have the same Hamming weight. By setting $j = 2$, we can minimize the carry effect as discussed in Section 3.1. Overall, the size of the search space is at most $64 \times 2^{32} = 2^{58}$

Using the above strategy, we first search for the best vectors predicting one-block collisions. For the full SHA-1, the best one is obtained by setting $x_{64,2} = 1$ and $x_{i,2} = 0$ for $i = 65, 79$.

Now, we continue to search for good disturbance vectors that predict near collisions and two-block collisions. To do so, we compute more vectors after step 80 using the same SHA-1 message expansion formula. Then we search all possible 80-vector intervals $[x_i, \ldots, x_{i+79}]$. Any set of 80 vectors with small enough Hamming weight can be used for constructing a near collision. Finally, we compare the minimal Hamming weight of disturbance vectors found by experiments when different conditions are imposed.

New analysis techniques:

1) Use “subtraction” instead of “exclusive-or” as the measure of difference to facilitate the precision of the analysis.
2) Take advantage of special differential properties of IF. In particular, when an input difference is 1, the output difference can be 1, −1 or 0. Hence the function can preserve, flip or absorb an input difference, giving good flexibility for constructing differential paths.
3) Take advantage of the carry effect. Since $2^j = -2^j - 2^{j+1} - \ldots - 2^{j+k-1} + 2^{j+k}$ for any $k$, a single bit difference $j$ can be expanded into several bits. This property makes it possible to introduce extra bit differences.

4) Use different message differences for the 6-step local collision. For example, $(2^j, 2^j + 5, 0, 0, 0, 2^j + 30)$ is a valid message differences for a local collision in the first round.

5) Introduce extra bit differences to produce the impossible bit-differences in the consecutive local collisions corresponding to the consecutive disturbances in the first 16 steps, or to offset the bit differences of chaining variables produced by truncated local collisions.

We describe a useful technique for utilizing two sets of message differences corresponding to two consecutive disturbances within the same step $i$ to produce one 6-step local collision. For example, if there is a disturbance in both bit 1 and bit 2 of $x_i$, we can set the signs of the message differences $\Delta m_i$ to be opposite in those two bits.

This way, the actual message difference can be regarded as one difference bit in position 1, since $2^1 - 2^0 = 2$. Hence the number of conditions can be reduced from $4 + 2 = 6$ to 4.

Using the modification techniques described in this section, we can correct the conditions of steps 17-22. Furthermore, message modification will not result in increased complexity if we use suitable implementation tricks such as “pre computation”. First, we can pre compute and fix a set of messages in the first 10 steps and leave the rest as free variables. By Table 3, we know that there are 70 conditions in steps 23-77.

For three conditions in steps 23-24, we use the “early stopping technique”. That is, we only need to carry out the computation up to step 24 and then test whether three conditions in steps 23-24 hold. This needs about 12 step operations including message modification for correcting Conditions of steps 17-22. This is equivalent to about two SHA-1 operations.

Hence, the total complexity of finding the near-collision for the full SHA-1 is about $2^{68}$ computations. Considering the complexity of finding the second near-collision differential path, the total complexity of finding a full SHA-1 collision is thus about $2^{69}$.

<table>
<thead>
<tr>
<th>Index</th>
<th>Number Of Conditions</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>$4 - 1 - 1 = 2$</td>
<td>4 Cond’s: $A_{20}, A_{21}, A_{22}, A_{23} - A_{20}$ Due To Truncation $- A_{21}$ Using Modification</td>
</tr>
<tr>
<td>23,24,27,28</td>
<td>$2 \times 7 = 14$</td>
<td></td>
</tr>
<tr>
<td>32,35,36</td>
<td>$2 \times 7 = 14$</td>
<td></td>
</tr>
<tr>
<td>25,29,33,39</td>
<td>$4 \times 4 = 16$</td>
<td></td>
</tr>
<tr>
<td>43,45,47,49</td>
<td>$4 \times 4 = 16$</td>
<td></td>
</tr>
<tr>
<td>65,68,71,73,74</td>
<td>$4 \times 5 = 20$</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>3</td>
<td>Truncation</td>
</tr>
<tr>
<td>79</td>
<td>0</td>
<td>2 Conditions Ignored</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>1 Condition Ignored</td>
</tr>
<tr>
<td>Total</td>
<td>71</td>
<td></td>
</tr>
</tbody>
</table>

III. IMPROVED SHA-1 WITH EFFICIENT COLLISION RESISTANCE

Message expansion is the most vulnerable part of the SHA-1 algorithm. The vulnerability stems from the linearity of the message expansion formula. In order to remedy this subtle design problem this paper suggests a method which addresses this shortcoming by randomizing the 80-word message expansion.

The Improved SHA-1 is algorithmically similar to SHA-1. The word size and the number of rounds are same as that of SHA-1. In order to increase the security aspects of the algorithm the number of chaining variables is increased by one (six working variables) to give a message digest of length 192 bits. Also a different message expansion is used in such a way that, the message expansion becomes stronger by generating more bit difference in each chaining variable.

**New Chaining variables:**

The new chaining variables, denoted by $A,B,C,D,E,F$ are initialized with the following values:

$A = 01234567$

$B = 9ABCDEF$
C = EDCBA98

D = 76543210

E = C3D2E1F0

F = 1F83D9AB

**Description of the new message expansion:**

After preprocessing is complete, following steps are followed for the message expansion:

Begin the message expansion by populating all the 80 32-bit words with the 16 available words (obtained from the 512-bit message block). This is accomplished by repeating the 16 word sequence 5 times until all the 80 words are obtained. Then, apply the equations below to obtain the actual message expansion.

For \( i = 0 \) to \( N(79) \):
- If \( 0 \leq N \leq 7 \), \( W_n = (W_{n+3} \oplus W_{n+9} \oplus W_{n+15} \oplus W_{n+16}) < 1 \)
- If \( 8 \leq N \leq 15 \), \( W_n = (W_{n-8} \oplus W_{n-9} \oplus W_{n-15} \oplus W_{n-16}) > 16 \)
- If \( 16 \leq N \leq 31 \), \( W_n = (W_{n-3} \oplus W_{n-5} \oplus W_{n-13} \oplus W_{n-16}) < 16 \)
- If \( 32 \leq N \leq 63 \), \( W_n = (((W_{n-3} \oplus W_{n-3}) < 3) \oplus (W_{n-3} \oplus W_{n-3})) > 3 \)
- If \( 64 \leq N \leq 79 \), \( W_n = (W_{n+16}) < 16 \)

**Modified Compression function:**

For \( N = 0 \) to \( 79 \):

\[
P = \text{ROTL}^3(A) + F1(B,C,D) + E + KN + WN
\]

\[
Q = \text{ROTL}^16(A) + F1(B,C,D) + E + F + K_N + WN
\]

\[
F = P
\]

\[
E = \text{ROTL}^5(D)
\]

\[
D = C
\]

\[
C = \text{ROTL}^{30}(B)
\]

\[
B = A
\]

\[
A = Q
\]

Where \( K_t \) is a constant defined by a Table 1, F1 is a bitwise Boolean function, for different rounds defined by,

\[
F1(B,C,D) = \text{IF B THEN C ELSE D}
\]

\[
F1(B,C,D) = B \oplus C \oplus D
\]

\[
F1(B,C,D) = \text{MAJORITY}(B,C,D)
\]

\[
F1(B,C,D) = B \oplus C \oplus D
\]

where the “IF...THEN...ELSE...” function is defined by IF B THEN C ELSE D = (BAC)(¬B)AD

And “MAJORITY” function is defined by MAJ(B, C, D) = (BAC) V (CAD) V (DAB)

Also, ROTL is the bit wise rotation to the left by a number of positions specified as a superscript.

<table>
<thead>
<tr>
<th>Rounds</th>
<th>Steps</th>
<th>( F )</th>
<th>( K_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3-15</td>
<td>IF</td>
<td>5e827999</td>
</tr>
<tr>
<td>2</td>
<td>20-39</td>
<td>XOR</td>
<td>6dd6bab1</td>
</tr>
<tr>
<td>3</td>
<td>40-59</td>
<td>MAJ</td>
<td>8fbbbe3c</td>
</tr>
<tr>
<td>4</td>
<td>60-79</td>
<td>XOR</td>
<td>m62c1f4c</td>
</tr>
</tbody>
</table>

Security of the improved algorithm is higher than SHA1. Sophisticated message modification techniques were applied. This scheme is 192 bits and need \( 2^{96} \) bits. Heed [5], for birthday paradox and is strong enough to pre image and second pre image attack. In SHA-1 the input 512 block is the source for the 16 independent words of the 80-word message block. The other 64 words are obtained by manipulating the initial 16. By [3], we know that the 16 differential words mitigate the chances of collisions in the initial rounds of the algorithm but eventually when the differential propagates through the various rounds, collision becomes a possibility. The improvised message expansion which has been suggested in this paper rectifies this discrepancy by using the initial 16 words as random variables throughout the message expansion. The iterations concerning each set of rounds minimize the differential propagation by making use of already existing values which occur previously in the expansion. At the same time, other values from the next indices are also used. This not only randomizes the expansion but also eliminates propagation by choosing anything but a serial path.

Obtaining the hash out of plain text using the techniques listed above invariably increases the time taken. This is mainly due to extra set of operations which have been included to fulfill the objective.

**REFERENCES**


