SHA_1 Enhancement Based on Resilent Boolean Function

Ahmed Y. Yousif [1], Tayseer S. Atia [1], Abdelsattar S.Auadh[2]
atds@yahoo.com, tsamastersc@yahoo.com

Abstract

SHA_1 is a one way hash function which is used in cryptographic systems to provide message authentication and integrity. In recent year this algorithm faced new type of attacks. These attacks make use of the simplicity of message expansion step to leak some information used to build a matching patterns or build differential path according to local collision. In this paper an enhanced version of SHA_1 was proposed depend on using resilent Boolean function which is a Boolean function that offers properties of balancedness, algebraic degree, correlation immunity and nonlinearity. This enhancement tends to countermeasures these attacks. Enhancement was made in two places for the original algorithm, first enhanced message expansion process, second change the value of 32 variable inputs to Boolean function in the algorithm.

تحسين SHA_1 بالاعتماد على الدوال المنطقية الصامتة

احمد يونس يوسف، تيسير سلمان عطية، عبد الستار سالم عوض

الخلاصة

SHA_1 هي من دوال الطريق الواحد والتي تستخدم في انظمة التجفير لتوفير الموثوقية والسلامة. في السنوات الاخيرة واجهت هذه الخوارزمية انواع جديدة من الهجمات. هذه الهجمات تستفيد من بساطة مرحلة التوسع في الرسالة لتسريب بعض المعلومات التي تستخدم في بناء انماط المطابقة او بناء الطريق التفاضلية وفقا للتصادم المحلي. في هذا البحث اصدار محسن من **SHA_1** تم اقتراحه بالاعتماد على استخدام الدوال المنطقية الصامتة والتي توفر خصائص التوازن ، الدرجة الجبرية، مناعة الارتباط واللاخطية. هذا التحسين يهدف الى مقامة تلك الهجمات. التحسين قد وضع في موضعين في الخوارزمية الاصلية، الاول تحسين مرحلة توسيع الرسالة، والثاني تغيير قيم المتغيرات ذات حجم ٣٢ بت المدخلة الى الدوال المنطقية في الخوارزمية.

1 Department of Computer Science and Information System of University of Technology in Iraq
2 Construction and Projects directorate-ministry of higher educations

1. Introduction

The hash function SHA-1 was issued by NIST in 1995 as a Federal Information Processing Standard [1]. Since its publication, SHA-1 has been adopted by many government and industry security standards, in particular standards on digital signatures for which a collision-resistant hash function is required. In addition to its usage in digital signatures, SHA-1 has also been deployed as an important component in various cryptographic schemes and protocols, such as user authentication, key agreement, and pseudorandom number generation. Consequently, SHA-1 has been widely implemented in almost all commercial security systems and products.

The group of the SHA family. Based on the idea of extended Feistel permutation , they are equipped with more complex message expansion algorithm. The first function of that family was SHA-0 [2]. It was promptly replaced by an improved version, SHA-1 [3]. Security concerns appeared to be true, as in 1998 Chabaud and Joux presented theoretical attack on SHA-0 [4], which was later implemented and improved allowing for finding an actual collision [5, 6]. Now, one of the most interesting questions in the field of hash functions analysis is how secure is the present standard SHA-1, which is different from SHA-0 by only one rotation in the message expansion process. The same principle used to attack SHA-0 could be applied to construct an attack on SHA-1 provided that there exists good enough differential pattern. Biham and Chen were able to find patterns that allowed for finding collisions for variants reduced to first 34 and 36 steps . The attack can be extended provided that one can find good differential patterns for longer variants of SHA-1.

Keywords: Hash functions, collision search attacks, SHA-0, SHA-1,SRB,LFSR

2. Description of SHA-1 compression function [7]

SHA-1 compression function hashes 512 bit input messages to 160 bit digests.

Firstly, 512 bits of the message are divided into 16 words $W_0, W_1, \dots, W_{15}$ of 32 bits each. The rest of 80 words is generated out of the first 16 words according to the following recurrent formula:

$$Wi = ROL^1(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \text{ for } 16 \le i \le 79, \quad (1)$$

where ROL^k means rotation of word by k positions left. If this is the first application of the compression function , five 32-bit registers A, B, C, D, E are initialized to values A_0 = 0x67452301, B_0 = 0xEFCDAB89, C_0 = 0x98BADCFE, D_0 = 0x10325476, E_0 = 0xC3D2E1F0 accordingly. Next, 80 steps (i = 0. . . 79) of the following form are applied:

$$A_{i+1} = ROL^5(A_i) \boxplus f_i(B_i,C_i,D_i) \boxplus E_i \boxplus W_i \boxplus K_i, \qquad (2)$$

$B_{i+1} = A_i$, $C_{i+1} = ROL^{30}(B_i)$,

$D_{i+1} = C_i$, $E_{i+1} = D_i$,

where $\boxplus$ denotes addition modulo 2^32, and Ri means the value of the register, R after i-th iteration. Functions fi and constants Ki used in each iteration are defined as

$$fi(B,C,D) = \begin{cases} BC \vee (\neg B)D & \text{for } 0 \le i \le 19 \\ B \oplus C \oplus D & \text{for } 20 \le i \le 39 \\ BC \vee BD \vee CD & \text{for } 40 \le i \le 59 \\ B \oplus C \oplus D & \text{for } 60 \le i \le 79 \end{cases}$$

$$Ki = \begin{cases} \text{0x5A827999} & \text{for } 0 \le i \le 19 \\ \text{0x6ED9EBA1} & \text{for } 20 \le i \le 39 \\ \text{0x8F1BBCDC} & \text{for } 40 \le i \le 59 \\ \text{0xCA62C1D6} & \text{for } 60 \le i \le 79 \end{cases}$$

3. Attacks on SHA

This section give an overview about the most recent attack against SHA .

3.1 Differential Attack of Chabaud and Joux

Chabaud and Joux presented in [4] differential attack on SHA-0. The fundamental observation they made is that a change in bit j of word W_i can be corrected by complementary changes in the following bits:

– bit (j + 6) mod 32 of word W_{i+1},

– bit j of word W_{i+2},

– bit (j + 30) mod 32 of word W_{i+3},

– bit (j + 30) mod 32 of word W_{i+4},

– bit (j + 30) mod 32 of word W_{i+5},

provided that functions f_{i+1},. . . ,f_{i+4} and additions $\boxplus$ behave like linear functions, that is a single change in the input to f results in a change of output of *f*, change in two inputs of *f* leaves the result unchanged and differences propagate through additions without generating carries The attack is possible due to the property of the message expansion function which does not mix bits in different positions. Thanks to that it was possible to consider message expansion algorithm as a bit-wise. Enumeration of all 2^{16} possible bit patterns in position 1 allowed for choosing disturbance pattern in bit one that led to a global differential pattern ± producing a collision with probability 2^{-61}.

3.2 Differential patterns attacks on **SHA-1**

This attack depend on the Analysis of the message expansion algorithm of SHA-1,The important property of the message expansion process given by the formula (1) is that when considered as a function producing 80 new words out of 16 old ones it is a bijection. This implies that it is possible to reconstruct the whole expanded message given any 16 consecutive words of it, in particular the first 16. Moreover, if we consider it on a bit level as a function A : $F^{512}{}_{2} \rightarrow F^{512}{}_{2}$, it is easy to see that A is F_2-linear as the only operations used are word rotations (which are permutations of bits) and bitwise XOR operations. Then the expansion of the initial message1 m € $F^{512}{}_{2}$ can be expressed as a long vector :

$$E_1(m) = \begin{array}{|c|} \hline m \\ \hline A(m) \\ \hline A2(m) \\ \hline A3(m) \\ \hline A4(m) \\ \hline \end{array} \qquad € \;\; F^{2560}{}_{2}$$

The set of correction masks is built from a disturbance pattern by rotations and delaying the pattern by 1, 2. . . 5 words in the same way as described in [4]. In order to find disturbance patterns which can give rise to correction patterns one has to look for bit patterns b € $F^{2560}{}_2$ that satisfy the following conditions:

1. Consider m to be a column vector
2. The pattern b has to be of the form (5), i.e. b is the result of the expansion Operation,
3. The pattern b ends with 5 *32 = 160 zero bits (the last five words are zero), because each disturbance is corrected in the next 5 steps, so no disturbance May occur after the word 74,
4. After delaying a pattern by up to 5 words (that is, shifting bits of b down (right) by 5 * 32 = 160 positions) the shifted pattern must also be the result of the expansion of its first 512 bits, that is $[\ 0 \ldots 0 \mid\ b0\ b1 \ldots b2399]^T = E_1([0 \ldots 0\ b0 \ldots b351]^T)$.
4. b has both the minimal Hamming weight and the maximal number of non-zero bits in position 1.

3.3 Local Collisions of SHA-1

Informally, a local collision is a collision within a few steps of the hash function. A simple yet very important observation made in [8] is that SHA-0 has a 6-step local collision that can start at any step *i*. The collision differential path on SHA-0 chooses *j* = 2 so that *j* + 30 = 32 becomes the MSB 4 (Most Significant Bit) to eliminate the carry effect in the last three steps. In addition, the following condition,

$m_i,2 = \neg m_{i+1},7$

helps to offset completely the chaining variable difference in the second step of the local collision, where *mi*,*j* denotes the *j*-th bit of message word *mi*. The message condition in round 3,

$M_i,2 = \neg m_{i+2},2$

helps to offset the difference caused by the non-linear function in the third step of the local collision. Since the local collision of SHA-0 does not depend on the message expansion, it also applies to SHA-1. Hence, this type of local collision can be used as the basic component in constructing collisions and near collisions of the full 80-step SHA-0 and SHA-1.

3.4 New Collision Search Attacks on SHA-1[9]

This technique works by; finding a disturbance vector with low Hamming weight is a necessary step in constructing valid differential paths that can lead to collision. To construct such a path for SHA-1, it needs 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*. It is easy to show that the disturbance vector satisfies the same recursion defined by the message expansion. For the 80 variables *xi*, any 16 consecutive ones determine the rest. So there are 16 free variables to be set for a total of 2^{16} possibilities. Then a "good" vector satisfying certain conditions can be easily searched with complexity 2^{16}. In order for the disturbance vector to lead to a possible collision,

On the other hand, the three conditions imposed on disturbance vectors seem to a major obstacle conditions on the disturbance vectors need to be imposed , summarize in Table1.

Table (1) Conditions on disturbance vectors for SHA-1 with *t* steps

Condition	Purpose
xi = 0 for *i* = *t* ¡ 5,.., *t* ¡ 1 to produce a collision	in the last step *t*
xi = 0 for *i* = -5,...,-1 to avoid truncated local	collisions in first few steps
no consecutive ones to avoid an impossible in the first 16 variables	in same bit position collision path due to a property of IF

There have been attempts to remove some of the conditions. For example, 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 (although it is no longer SHA-1 itself). Even with both relaxation, the Hamming weight of the disturbance vectors is still too high to be useful for the full 80-step SHA-1.this is the key idea of new attack by relax all three condition in Table(1). In other words, impose no condition on the vectors other than they satisfy the message expansion recursion. 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. This allows finding disturbance vectors whose Hamming weights are much lower than those used in existing attacks.

Then present several new techniques for constructing a valid differential path given such disturbance vectors. The resulting path is very complex in the first round due to consecutive disturbances as well as truncated local collisions that initiate from steps -5 through -1. This is the most difficult yet crucial part of new analysis, without which it would be impossible to produce a *real* collision.

Once a valid differential path is constructed, the message modification techniques were applied, first introduced by Wang et. al in breaking MD5 and other hash functions [15, 11–13], to further reduce the search complexity. Such extension requires carefully deriving the exact conditions on the message words and chaining variables, which is much more involved in the case of SHA-1 compared with SHA-0 and other hash functions.

4. Cryptographic Boolean Functions[10]

Boolean functions play a central role in the design of most cryptosystems and in their security. There are several construction methods for constructing correlation immune and resilient Boolean functions. The most common of all these is the Maiorana-McFarland construction technique.

The purpose of the nonlinear combining function f is to make the output stream difficult for the cryptanalyst to predict. Such a function should posses certain desirable properties to withstand known cryptanalytic attacks. Four such important properties are balancedness, correlation immunity, algebraic degree and nonlinearity. Construction of resilient Boolean functions achieving the upper bound on nonlinearity is an important research area.

•**Balanced Boolean function:** A function f on $\{0,1\}^n$ is said to be balanced if its output column in the truth table contains equal number of 0's and 1's (i.e., $wt(f)=2^{n-1}$), where $wt(f)$ is the Hamming weight of the Boolean function f .

•**Algebraic Normal Form and Algebraic Degree:** Every *n*-variable Boolean function can be represented with its truth-table. But the representation of Boolean functions which is most usually used in cryptography is the *n*-variable polynomial representation over $\{0,1\}$, of the form:

$$f(x_1,...,x_n) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus ... \oplus a_nx_n \oplus a_{12}x_1x_2 \oplus a_{13}x_1x_3 \oplus ... \oplus a_{12...n}x_1x_2...x_n \quad \dots (3)$$

where coefficients $a_0, a_1, a_{12}, ..., a_{12...n} \in \{0,1\}$. This representation of f is called the **A**lgebraic **N**ormal **F**orm (**ANF**) of f . The number of variables in the highest order product term with nonzero coefficient is called the *algebraic degree* d , or simply degree of f **.**

Thus $x_1 \oplus x_2$ has degree 1, $x_1 \oplus x_1x_2x_3$ has degree 3 etc.

•**Correlation Immune Boolean Functions (CI):** A Boolean function f on *n*-variables is said to be *mth*-order correlation immune (***mth*-CI**), if for any *m*-tuple of binary random variables $x_{i1}, x_{i2}, ...\ x_{im}$ we have

$$I\,(x_{i1}, x_{i2}, ...\ x_{im}; \mathrm{Z}) = 0 \quad , \quad 1 \le \mathrm{i}_1 < \mathrm{i}_2 < ... < \mathrm{i}_m \le \mathrm{n} \qquad .... \ (4)$$

where $Z = f(x_1,...,x_n)$, and I (x; Z) denotes the mutual information **.**

•Nonlinearity

The output to any Boolean function f always has correlation to certain linear functions of its inputs. But this correlation is showed *small.* In other words, the minimum Hamming distance between f and all affine functions must be *high*. This is called the *nonlinearity* of *n*-variable function f and denoted by

$$nl(f) = \min_{g \in A(n)} (d(f, g)) \qquad ..... (5)$$

where A(*n*) is the set of all *n*-variable affine functions.

4.1 Resilient Boolean Functions[10]

A balanced *mth*-order correlation immune Boolean function is called *m*-resilient Boolean function.

Let $x=(x_1,...,x_n)$ *and* $\omega=(\omega_1,...,\omega_n)$ both belong to $\{0,1\}^n$ and $x.\omega = x_1.\omega_1 \oplus ... \oplus x_n.\omega_n$. Let $f(x)$ be a Boolean function on *n*-variables. Then the ***Walsh transform*** of $f(x)$ is a real valued function over $\{0,1\}^n$, that can be defined as

$$W_f(\omega) = \sum_{x\in\{0,1\}^n} (-1)^{f(x)\oplus x.\omega} \qquad \text{...... (6)}$$

The Walsh transform is sometimes called the spectral distribution or simply the *spectra* of a Boolean function. The spectra (value of Walsh distance) is

$$\begin{aligned} wd(f,g) &= \#(f=g) - \#(f\neq g) \\ &= \lambda - 2\, d(f,g) \end{aligned}$$

where λ is the length of both f and g.

The *linear complexity* of an infinite binary sequence *s*, denoted L(*s*), is defined as follows :

- if *s* is the zero sequence *s* = 0, 0, 0, … , then L(*s*) = 0.
- if no LFSR generates *s*, then L(*s*) = ∞ .
-otherwise, L(*s*) is the length of the shortest LFSR that generates *s*.

The *linear complexity* of a finite binary sequence s^n, denoted L(s^n), is the length of the shortest LFSR that generates a sequence having s^n as its first *n* terms .

4.2 Construction the Sequences of Saturated Best Resilient Function SBRSs (*i*)

This section presents the construction of the sequences of saturated best resilient functions. In defining SBRS we state that any function in an SBRS must be an SBR function[10].

Let $f_{i,j}$ be a *j*-th function of SBRS(*i*). Then the function $f_{i,j+1} = X \oplus f_{i,j}$ (where the variable X does not occur in $f_{i,j}$) is $f_{i,j+1}$ function of SBRS(*i*). Consequently, if one can construct $f_{i,j}$, then one can construct $f_{i,k}$ for all $k > j$.

For SBRS(1), it is easy to construct $f_{1,1}$, since $f_{1,0}$ is (5,1,3,12)(i.e. 5-varaibles, 1-resilient, 3-degee, 12-nonlinearity) SBR function. Note that all functions in SBRS(*i*) have the same degree **2+*i***.

This shows that if one can construct any one of the functions in SBRS(*i*), then it is possible to construct any function in the succeeding part of the sequence. Thus it is enough if the initial function of each sequence is constructed.

Example (2)

To construct the function (6,2,3,24), we take an initial (5,1,3,12) SBR function:

$$f_1(x_1,...,x_5) = x_2x_4x_5 \oplus x_1x_4 \oplus x_1x_5 \oplus x_2x_5 \oplus x_3x_5 \oplus \mathrm{x}_1 \oplus x_2 \oplus x_3$$

This function can be testing by using Walsh transform to ensure that it is resilent Boolean function.

$$f(x_1,...,x_6) = f(x_1,...,x_5) \oplus \mathrm{x}_6$$

Then

$$f_2(x_1, ..., x_6) = x_2x_4x_5 \oplus x_1x_4 \oplus x_1x_5 \oplus x_2x_5 \oplus x_3x_5 \oplus \mathrm{x}_1 \oplus x_2 \oplus x_3 \oplus x_6$$

Then $nl(f_{0,1}) = 2nl(f_{0,0}) = 24$ according to theorem 2 in [1]. Then the function (6,2,3,24) is SBR function.

5. Proposed Enhanced SHA_1

The proposed modification to the SHA_1 algorithm takes place in two steps according to the analysis of attacks against the algorithm, the analysis to SHA_1 attacks show that these attack depends on their attacks on the message expansion process, so the Enhancement is implemented in two places to made the algorithm more complex and complicated to be analyzed. This is done by using the saturated Best Resilient Function since this Boolean function offer the properties of balancedness, algebraic degree, correlation immunity and nonlinearity. The first place in message expansion equation , second it used in the 32 value processed by the SHA_1 Boolean function also to make them difficult to guessed. SBR used in this paper to enhance the SHA_1 has the following parameters:

6-varaibles, 2-resilient, 3-degee, 24-nonlinearity write as (6,2,3,24).

The formula of the SBR is

$$f(x_1, ..., x_6) = x_2x_4x_5 \oplus x_1x_4 \oplus x_1x_5 \oplus x_2x_5 \oplus x_3x_5 \oplus \mathrm{x}_1 \oplus x_2 \oplus x_3 \oplus x_6$$

This function required 6 variables, each variables is a binary bit can be 0 or 1 and the output from it also one bit. To apply this function in the proposed algorithm, six LFSR were used with the following length(4,5,7,4,5,7) respectively, the size of LFSR chosen such that gcd(m,n)=1 . Figure(1) shows the structure of the generator.

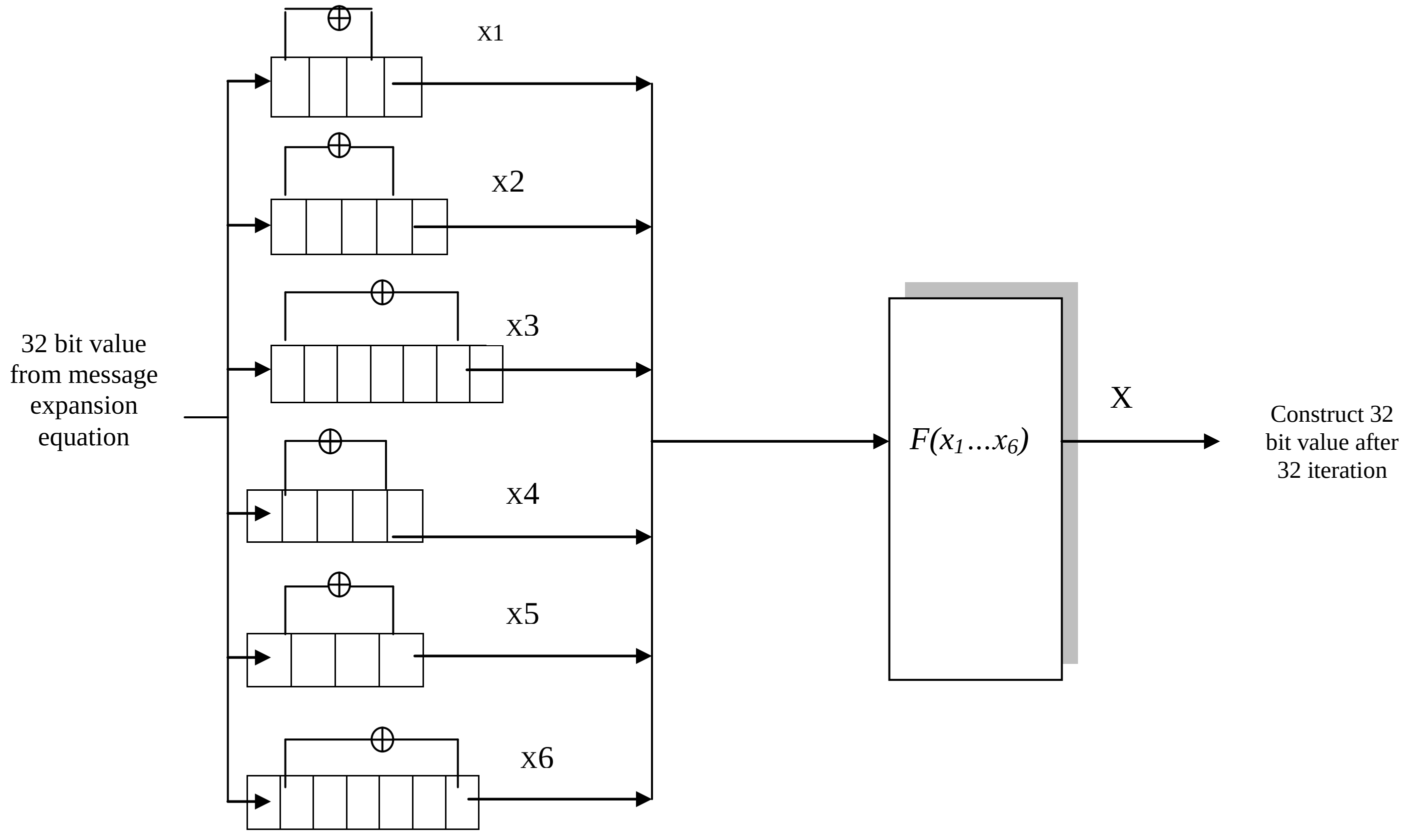

x1
x2
x3
x4
x5
x6
32 bit value from message expansion equation
F(x1…x6)
X
Construct 32 bit value after 32 iteration

Figure (1) structure of binary sequence for SBR of six variables

The selected SBR used in this paper is not constant any other functions can be used with different parameters. SBR used in this paper is the best functions among several function where the test result shows that it's gave reasonable execution time with the desired complexity. Finally tap function in LSBR is not constant it can implemented between any selected cell.

Enhanced algorithm

a. *Divide M(r) into 16 words W(0), W(1), ... , W(15), where W(0) is the left-most word.*

b. *For t = 16 to 79 let*

$$W(t) = S^{1} (F_{SBR}(W(t-3) \oplus W(t-8) \oplus W(t-14) \oplus W(t-16))).$$

Where S^{n} *mean circular left shift operation by n*

F_{SBR} *is saturated Best Resilient Function with (6,2,3,24)*

c. *Let A = H0, B = H1, C = H2, D = H3, E = H4.*

d. *For t = 0 to 79 do*

$F_{RSB}(B); F_{RSB}(C); F_{RSB}(D)$

$TEMP = S^{5}(A) \boxplus f(t;B,C,D) \boxplus E \boxplus W(t) \boxplus K_{t}$

e. $E = D; D = C; C = S^{30}(B); B = A; A = TEMP.$

f. *Let H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.*

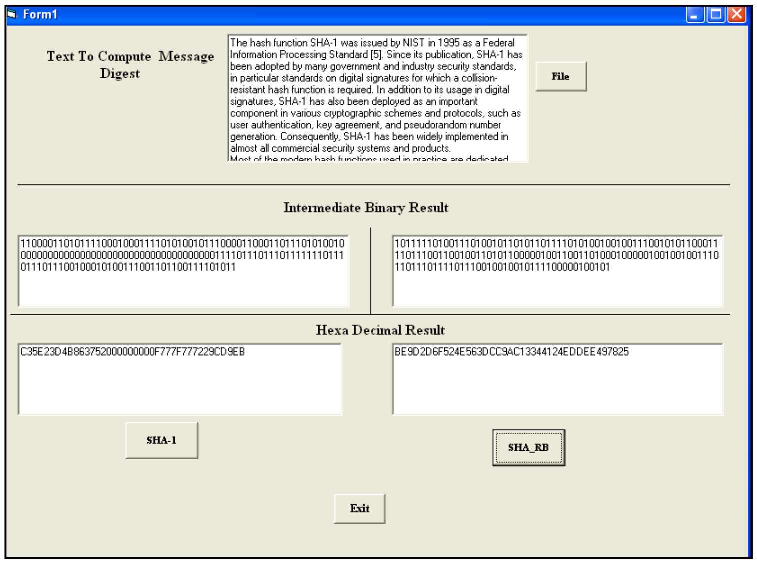
6. Experimental Results

Test result to the proposed enhanced SHA_1 algorithm is done in two cases, first implement the proposed algorithm using SBR at message expansion step only and its demonstrated in Table(2),second implement the algorithm using SBR at message expansion step and before the Boolean function. These test is made against original SHA_1 with different tests to check the correctively of the enhanced algorithm. These tests are implemented in different ways, if the message modified by add, change or delete some character from it. Also running time for both version of SHA_1 was measured. Figure (2) shows the computed message digest in binary and hexadecimal result for selected random input message using SHA_1 and the enhanced algorithm.

Table (2) output message digest using SBR in message expansion stage only

Output digest	case
34E69D2FA7D26384CC9AC13344124EDDEE497825	Input message without change
CB82920679FCDEA2CC9AC13344124EDDEE497825	Remove first character
7939B7BDA3BC8887CC9AC13344124EDDEE497825	Add new character
228591DECCE71D90CC9AC13344124EDDEE497825	Change first character with another

Figure (3) shows the output digest when the input message was altered by remove one character from the beginning by remove the character “T” from word “this”, its clear that the output digest from both algorithm was differ also its differ from the output digest in Figure(2).



Form1

Text To Compute Message Digest

The hash function SHA-1 was issued by NIST in 1995 as a Federal Information Processing Standard [5]. Since its publication, SHA-1 has been adopted by many government and industry security standards, in particular standards on digital signatures for which a collision-resistant hash function is required. In addition to its usage in digital signatures, SHA-1 has also been deployed as an important component in various cryptographic schemes and protocols, such as user authentication, key agreement, and pseudorandom number generation. Consequently, SHA-1 has been widely implemented in almost all commercial security systems and products.
Most of the modern hash functions used in practice are dedicated

Intermediate Binary Result

```
110000110101111000100011110101001011100001100011011101010010
000000000000000000000000000000000111101110111011111110111
0111011100100010100111001101100111101011
```

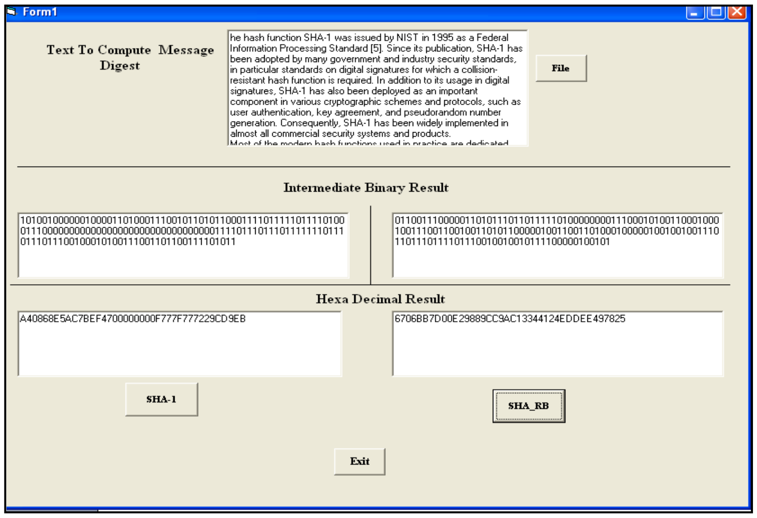
```
1011111010011101001011010110111101010010010011100101011000111
110111001100100110101100000100110011010001000001001001001110
1101110111101110010010010111100000100101
```

Hexa Decimal Result

```
C35E23D4B863752000000000F777F777229CD9EB
```

```
BE9D2D6F524E563DCC9AC13344124EDDEE497825
```

Figure (2) output digest for both version of SHA_1 using random input message



Form1

Text To Compute Message Digest

he hash function SHA-1 was issued by NIST in 1995 as a Federal Information Processing Standard [5]. Since its publication, SHA-1 has been adopted by many government and industry security standards, in particular standards on digital signatures for which a collision-resistant hash function is required. In addition to its usage in digital signatures, SHA-1 has also been deployed as an important component in various cryptographic schemes and protocols, such as user authentication, key agreement, and pseudorandom number generation. Consequently, SHA-1 has been widely implemented in almost all commercial security systems and products.
Most of the modern hash functions used in practice are dedicated

Intermediate Binary Result

```
1010010000001000011010001110010110101100011110111110111101000
01110000000000000000000000000000001111011101110111111110111
0111011100100010100111001101100111101011
```

```
0110011100000110101110110111101000000001110001010011000100 0
100111001100100110101100000100110011010001000001001001001110
1101110111101110010010010111100000100101
```

Hexa Decimal Result

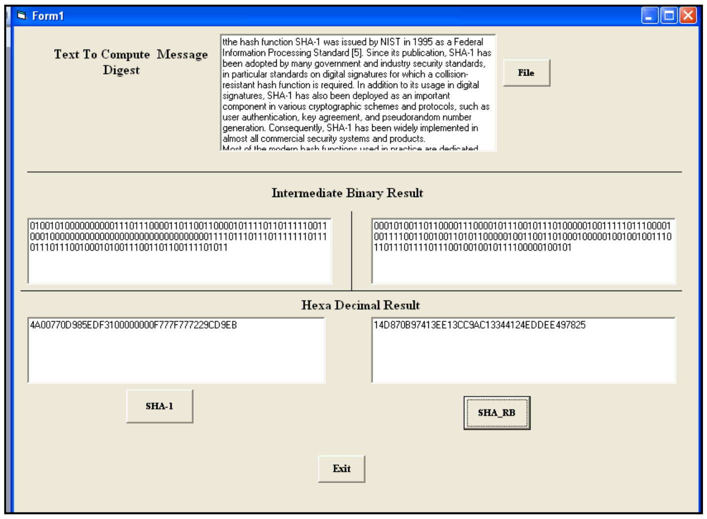
```
A40868E5AC7BEF4700000000F777F777229CD9EB
```

```
6706BB7D00E29889CC9AC13344124EDDEE497825
```

a. output after delete character "T"

Figure (4) and Figure (5) display the message digest when the input message modified by add new character "T" to word "This" and change character "T" to character "a" respectively.

Finally , the average time measured for both implementation show that the original SHA_1 need 2 sec to compute message digest while the proposed algorithm need 10 sec and this is due to further execution time needed by SBR in LSBR but this delay is compatible with the required complexity .



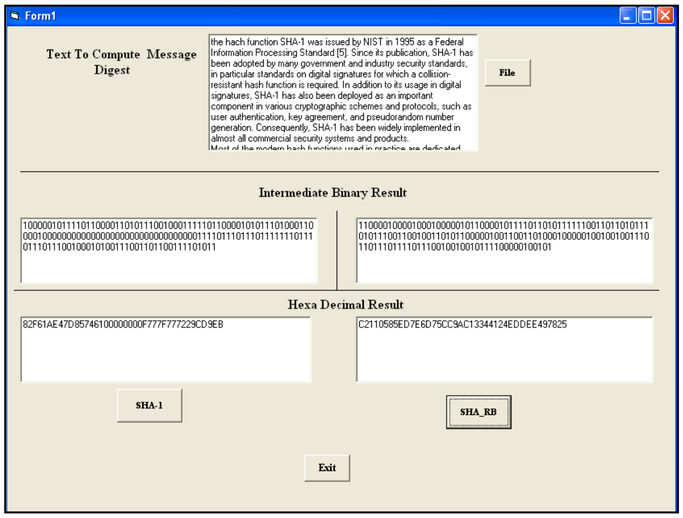
tthe hash function SHA-1 was issued by NIST in 1995 as a Federal Information Processing Standard [5]. Since its publication, SHA-1 has been adopted by many government and industry security standards, in particular standards on digital signatures for which a collision-resistant hash function is required. In addition to its usage in digital signatures, SHA-1 has also been deployed as an important component in various cryptographic schemes and protocols, such as user authentication, key agreement, and pseudorandom number generation. Consequently, SHA-1 has been widely implemented in almost all commercial security systems and products.
Most of the modern hash functions used in practice are dedicated

```
0100101000000000011101110000110110011000010111101101111110011
0001000000000000000000000000000001111011101110111111110111
0111011100100010100111001101100111101011
```

```
000101001101100001110000101110010111010000010011111011100001
001111001100100110101100000100110011010001000001001001001110
1101110111101110010010010111100000100101
```

```
4A00770D985EDF3100000000F777F777229CD9EB
```

```
14D870B97413EE13CC9AC13344124EDDEE497825
```



the hach function SHA-1 was issued by NIST in 1995 as a Federal Information Processing Standard [5]. Since its publication, SHA-1 has been adopted by many government and industry security standards, in particular standards on digital signatures for which a collision-resistant hash function is required. In addition to its usage in digital signatures, SHA-1 has also been deployed as an important component in various cryptographic schemes and protocols, such as user authentication, key agreement, and pseudorandom number generation. Consequently, SHA-1 has been widely implemented in almost all commercial security systems and products.
Most of the modern hash functions used in practice are dedicated

```
100000101111011000011010111001000111110110000101011101000110
0001000000000000000000000000000001111011101110111111110111
0111011100100010100111001101100111101011
```

```
110000100001000100000101100001011110110101111110011011010111
010111001100100110101100000100110011010001000001001001001110
1101110111101110010010010111100000100101
```

```
82F61AE47D85746100000000F777F777229CD9EB
```

```
C2110585ED7E6D75CC9AC13344124EDDEE497825
```

b. output after add new character "t" **c. output after change character t to a**

Figure (3) output digest for both version of SHA_1 after modify input message

7. Conclusion

In this paper an enhanced version of SHA_1 was proposed, this enhancement based on resilent Boolean function which offer the properties of balancedness, algebraic degree, correlation immunity and nonlinearity. this enhancement tend to countermeasures attacks on original algorithm which make use of the message expansion process to leak some information about message blocks and build a matching pattern ,or construct differential path based on local collision since this process is linear.

Experimental result show that using SBR function in message expansion only that the output digest correlated for different implementation for the same input message in different form. But using SBR in message expansion and before the Boolean function in the algorithm show uncorrelated message digest for the same message in different forms also the output is different completely from the original version of SHA_1 in each implementation.

Finally , the average time measured for both implementation show that the original SHA_1 need 2 sec to compute message digest while the proposed algorithm need 10 sec and this is due to further execution time needed by SBR in LFSR and its compatible with required degree of complexity .

References

1. NIST. *Secure hash standard*. Federal Information Processing Standard, FIPS-180-1, April 1995.

2. FIPS 180. Secure hash standard (SHS). National Institute of Standards and Technology, May 1993. Replaced by [15].

3. National Institute of Standards and Technology. Secure hash standard (SHS). FIPS 180-2, August 2002.

4. F. Chabaud and A. Joux. Differential collisions in SHA-0. In H. Krawczyk, ed- itor, Advances in Cryptology - CRYPTO'98, volume 1462 of LNCS, pages 56–71.
Springer-Verlag, 1998.

5. C. Lemuet. Collision in SHA-0. sci.crypt newsgroup message, Message-ID: cfg007$1h1b$1@io.uvsq.fr, 12 August 2004.

6. A. Joux. Collisions in SHA-0. Short talk presented at CRYPTO'04 Rump Session, 2004.

7. K. Matusiewicz and J. Pieprzyk, Finding good differential patterns for attacks on SHA-1,Centre for Advanced Computing - Algorithms and Cryptography, Department of Computing, Macquarie University,Sydney, NSW 2109.

8. 14. X. Y. Wang," The Collision attack on SHA-0".
www.infosec.edu.cn, 1997.

9. X. Wang, Y. Yin, and Hongbo Yu," Finding Collisions in the Full SHA-1", Shandong University, Jinan 250100, China.

10. A. Malik and S. Bader Sadkhan ," Mathematical Analysis of Design Parameters for Nonlinear Stream Cipher Systems",Msc,thesis, University of Technology , Department of Applied Science, June 2005.