Generalized Secure Hash Algorithm: SHA-X

Chu-Hsing Lin
Department of Computer Science
Tunghai University
Taichung, Taiwan, R.O.C.
chlin@go.thu.edu.tw

Chen-Yu Lee
Department of Computer Science
National Chiao-Tung University
HsinChu, Taiwan, R.O.C.
chenyu@cs.nctu.edu.tw

Yi-Shiung Yeh
Department of Computer Science
National Chiao-Tung University
HsinChu, Taiwan, R.O.C.
ysyeh@cs.nctu.edu.tw

Hung-Sheng Chien
Department of Computer Science
National Chiao-Tung University
HsinChu, Taiwan, R.O.C.

Shih-Pei Chien
Department of Computer Science
Tunghai University
Taichung, Taiwan, R.O.C.
bunny.chien@gmail.com

Abstract—**This paper defines a generalized SHA algorithm based on the SHA family rules. The proposed algorithm accepts arbitrary length message as input to generate message digest with the required length. It contains the initial values, constant values, padding, parsing, as well as the generalized main body. Further, the proposed algorithm solves the Length-of-the-Hash-Value (LHV) problem that occurs when SHA-*r* cannot be expressed as *r* = *mn* uniquely.**

Keywords: Secure hash algorithm, one-way hash function, cryptography, message digest, security, standard

I. INTRODUCTION

Cryptographic hash functions play an important role in modern cryptography. They are widely used in a variety of applications such as password protection, secure protocols, digital signatures, and more. The hash function uses a string of arbitrary length as its input and creates a fixed-length sting as its output. A hash value is often called a data fingerprint or message digest. The following sections provide some definitions of collision-free hash functions.

The Secure Hash Algorithm (SHA) is a series of cryptographic hash functions published by the National Institute of Standards and Technology (NIST). NIST proposed the SHA-0 as Federal Information Processing Standard Publication (FIPS PUB) 180 in 1993 [1] and announced a revised version, the SHA-1 (also called SHA-160) in FIPS PUB 180-1 as a standard instead of the SHA-0 in 1995 [2]. In 2001, the NIST published SHA as FIPS PUB 180-2 [3] consisting of four algorithms: SHA-160, SHA-256, SHA-384 and SHA-512. NIST updated FIPS PUB 180-2 [4] in 2004, specifying SHA-224 that matches the key length of 3DES [5].

Several recent studies propose extensions based on SHA. For example, RARSHA-256 [6] is composed of the SHA-256 compression function, and is faster than SHA-256 in parallel implementation. SHACAL and SHACAL-2 [7] are block ciphers based on SHA-1 and SHA-256, respectively. They were submitted to the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project in 2003. The 42-round SHACAL-2 is based on a related-key rectangle attack, which requires $2^{243.38}$ related-key chosen plaintexts with a running time of $2^{488.37}$ [8]. Yoshida and Biryukov replaced all arithmetic additions with XOR operations in SHA-256, calling it SHA-256-XOR, and found that SHA-256-XOR has a pseudo-collision resistance weakness up to 34 rounds [9].

Based on the method of finding collisions in SHA-0 [10], Rijmen and Oswald applied the same method to find collisions in SHA-1 in early 2005 [11]. They examined message scheduling in SHA-0 and SHA-1 and proved that the complexity of finding collisions in a reduced version of SHA-1 (with 53 rounds instead of 80 rounds) was less than 2^{80}. Wang, Yin, and Yu found collisions with a complexity of 2^{69} in the full 80-step SHA-1 [12]. NIST has announced that SHA-1 will be used until 2010, at which time it will be replaced by other SHA algorithms.

Since 2004, several authors have researched collisions for SHA-256. Gilbert and Handschuh reported a 9-round local collision with the complexity 2^{66} of the differential path [13]. Mendel et al later reduced this complexity to 2^{39} by [14]. Nikoli´c and Biryukov found 21-step collisions for SHA-256 using a nonlinear differential path with a complexity of 2^{19} [15].

The paper defines a generalized SHA algorithm based on SHA family rules. The algorithm contains the initial values,

constant values, padding, parsing, as well as the main body, and accepts arbitrary length message as input to generate message digest with required length. Further, the study solves the Length-of-the-Hash-Value (LHV) problem that occurs when SHA-*r* cannot be expressed as $r = mn$ uniquely.

The rest of this paper is organized as follows: In section II, we present each step of generalization process including padding, parsing, setting the initial hash values, constants, Boolean expressions and functions, message schedule, initializing the eight working variables, for-loop operation and computing the i^{th} intermediate hash values. Section III proposes the generalized SHA algorithm and shows the LHV problem. Section IV proposes the generalized SHA algorithm without the LHV problem. Section V concludes.

II. GENERALIZED SECURE HASH ALGORITHM

This section describes the processing of generalizing the Secure Hash Algorithm according to the SHA family algorithms. The process of generalization includes padding, parsing, setting the initial hash values, constants, Boolean expressions and functions, and message schedule; initializing the eight working variables and for-loop operation; and computing the i^{th} intermediate hash values. In the following section, we describe the processes of generalizing in detail.

A. The Length of One Word and the Number of Output Words

First, we define the length of one word as n such that n=32 in SHA-224 and SHA-256, and n=64 in SHA-384 and SHA-512.

Second, we should define the number of output words m. For example, the output length of SHA-256 is 256 bits, 8 words equally (m=8, 256 bits=8 word × 32 bits/word). Similarly, m=6 in SHA-384 (384 bits=6 words × 64 bits/word). On the basis of the SHA, we define the value of m ($6 \le m \le 8$), and the length of one word/block n is multiple of 32. With the m and n, we can generalize the SHA family to SHA-*mn*.

In SHA-*mn*, where m={6, 7, 8}, and n={32, 64}, we find two additional formats, called SHA-192 (m = 6 and n =32) and SHA-448 (m = 7 and n =64). The Complete SHA family is defined below.

TABLE I. VALUES OF *M* AND *N* FOR SHA FAMILY

Property	**SHA-*mn***					
Algorithm	***SHA-192***	***SHA-224***	***SHA-256***	***SHA-384***	***SHA-448***	***SHA-512***
Word Size (n)	32			64		
# of Output Words (m)	6	7	8	6	7	8
Message Digest Size	192	224	256	384	448	512
Block Size	512			1024		
Security[a]	2^{96}	2^{112}	2^{128}	2^{192}	2^{224}	2^{256}

a. The security complexity is under birthday attack.

B. Padding the Message M

The section generalizes the padding step in SHA-*mn*. Assuming that M is l bits ($0 \le l < 2^{2n}$), the padding process should satisfy the following two rules:

- If we have $l \le 14n-1 \pmod{16n}$, we should pad "$1||0^*||(l)_2$" up to the length of $\left\lceil \frac{l}{16n} \right\rceil \times 16n$. Notice that " $1||0^*$" denotes that "1" is followed by zero "0" bit or more than one bits and the $(l)_2$ denotes the length of message in binary.
- If we have $l > 14n-1 \pmod{16n}$, we should pad "$1||0^*||(l)_2$ " up to the length of $(\lceil l/16n \rceil+1)\times 16n$. Notice that" $1||0^*$" denotes that "1" is followed by zero "0" bit or more than one bits and the $(l)_2$ denotes the length of message in binary.

C. Parsing the Padded Message into Message Blocks

Based on the properties of SHA family, SHA-*mn* parses the padded message into N $16\times n$ bits blocks denoted by $M(1) \ldots M(N)$. For each $16\times n$-bit $M(i)$, the M will be divided into sixteen n-bit subblocks denoted by $M_0^{(i)} \ldots M_{15}^{(i)}$.

D. Setting the Initial Hash Values

The initial hash values consist of eight n-bit words denoted by $H_0^{(0)} \ldots H_7^{(0)}$. We generalize the properties of setting initial hash value for SHA-*mn*:

- For some x, if m=8 and n= 64x-32 or 64x, we map to 1st to 8th prime numbers. And the 64x-32 bits are obtained by truncating the last 32 bits of the 64x bits.
- For some x, if m=7 and n=64x-32 or m=6 and n=64x, we map to 9th to 16th prime numbers. The 64x-32 bits are obtained by truncating the first 32 bits of the 64x bits.
- For some x, if m=6 and n=64x-32 or m=7 and n=64x, we map to 17th ~ 24th prime numbers. The 64x-32 bits are obtained by truncating the last 32 bits of the 64x bits.

$$\left.\begin{array}{l} H(1) = 6a09e667\ \ f3bcc908 \\ H(2) = bb67ae85\ \ 84caa73b \\ H(3) = 3c6ef372\ \ fe94f82b \\ H(4) = a54ff53a\ \ 5f1d36f1 \\ H(5) = 510e527f\ \ ade682d1 \\ H(6) = 9b05688c\ \ 2b3e6c1f \\ H(7) = 1f83d9ab\ \ fb41bd6b \\ H(8) = 5be0cd19\ \ 137e2179 \end{array}\right\} \underbrace{\underbrace{\text{SHA-256 ; } m=8, n=32}_{\text{SHA-512; } m=8, n=64}}$$

$$\left.\begin{array}{l} H(09) = cbbb9d5d\ \ c1059ed8 \\ H(10) = 629a292a\ \ 367cd507 \\ H(11) = 9159015a\ \ 3070dd17 \\ H(12) = 152fecd8\ \ f70e5939 \\ H(13) = 67332667\ \ ffc00b31 \\ H(14) = 8eb44a87\ \ 68581511 \\ H(15) = db0c2e0d\ \ 64f98fa7 \\ H(16) = 47b5481d\ \ befa4fa4 \end{array}\right\} \underbrace{\underbrace{\text{SHA-224; } m=7, n=32}_{\text{SHA-384; } m=6, n=64}}$$

$$\begin{cases} H(17) = a\,e\,5\,f\,9\,1\,5\,6 & e\,7\,b\,6\,d\,9\,9\,b \\ H(18) = c\,f\,6\,c\,8\,5\,d\,3 & 9\,d\,1\,a\,1\,e\,1\,5 \\ H(19) = 2\,f\,7\,3\,4\,7\,7\,d & 6\,a\,4\,5\,6\,3\,c\,a \\ H(20) = 6\,d\,1\,8\,2\,6\,c\,a & f\,d\,8\,2\,e\,1\,e\,d \\ H(21) = 8\,b\,4\,3\,d\,4\,5\,7 & 0\,a\,5\,1\,b\,9\,3\,6 \\ H(22) = e\,3\,6\,0\,b\,5\,9\,6 & d\,c\,3\,8\,0\,c\,3\,f \\ H(23) = 1\,c\,4\,5\,6\,0\,0\,2 & c\,e\,1\,3\,e\,9\,f\,8 \\ H(24) = 6\,f\,1\,9\,6\,3\,3\,1 & 4\,3\,a\,0\,a\,f\,0\,e \end{cases}$$

SHA-192; m=6,n=32

SHA-448; m=7,n=64

Figure 1. Initial values of SHA-*mn*

E. Setting the Constants.

In SHA, SHA-224 and SHA-256 obtain 64 constants by computing the first 32 bits of the fractional parts of the cube roots of the first 64 prime numbers denoted by $K_0^{\{256\}} \ldots K_{63}^{\{256\}}$. Similarly, SHA-384 and SHA-512 obtain 80 constants by computing the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers denoted by $K_0^{\{512\}} \ldots K_{79}^{\{512\}}$.

We can compute the constants by computing the first n bits of the fractional parts of the cube roots of the first $f_{13}(n)$ prime numbers.

$$\mathrm{f}_{13}(\mathrm{n}) = \frac{1}{2}\mathrm{n} + 48 \tag{1}$$

F. Boolean Expressions and Functions

In SHA-*mn*, the paper renames *Ch*() and *Maj*() functions to $g_1()$ and $g_2()$ and merges some Σ and σ functions described in SHA family. Note that $ROTR^k(x)$ means to rotate right k bits, and $SHR^k(x)$ means to shoft right k bits.

- $\sum_0^{\{224\}}(\mathrm{x}), \sum_0^{\{256\}}(\mathrm{x}), \sum_0^{\{384\}}(\mathrm{x}), \sum_0^{\{512\}}(\mathrm{x})$are merged to

  $$g_3()=\mathrm{ROTR}^{\mathrm{f}_1(\mathrm{x})}(\mathrm{x}) \oplus \mathrm{ROTR}^{\mathrm{f}_2(\mathrm{x})}(\mathrm{x}) \oplus \mathrm{ROTR}^{\mathrm{f}_3(\mathrm{x})}(\mathrm{x}) \tag{2}$$

  ,where $\mathrm{f}_1(\mathrm{n}) = \frac{13}{16}\mathrm{n} - 24$, $\mathrm{f}_2(\mathrm{n}) = \frac{21}{32}\mathrm{n} - 8$, and $\mathrm{f}_3(\mathrm{n}) = \frac{17}{32}\mathrm{n} + 5$.

- $\sum_1^{\{224\}}(x), \sum_1^{\{256\}}(x), \sum_1^{\{384\}}(x), \sum_1^{\{512\}}(x)$ are merged to

  $$g_4()=\mathrm{ROTR}^{\mathrm{f}_4(\mathrm{x})}(\mathrm{x}) \oplus \mathrm{ROTR}^{\mathrm{f}_5(\mathrm{x})}(\mathrm{x}) \oplus \mathrm{ROTR}^{\mathrm{f}_6(\mathrm{x})}(\mathrm{x}) \tag{3}$$

  , where $f_4(n) = \frac{1}{4}n - 2$, $f_5(n) = \frac{7}{32}n + 4$, and $f_6(n) = \frac{1}{2}n + 9$.

- $\sigma_0^{\{224\}}(x), \sigma_0^{\{256\}}(x), \sigma_0^{\{384\}}(x), \sigma_0^{\{512\}}(x)$ are merged to

  $$g_5()=\mathrm{ROTR}^{\mathrm{f}_7(\mathrm{x})}(\mathrm{x}) \oplus \mathrm{ROTR}^{\mathrm{f}_8(\mathrm{x})}(\mathrm{x}) \oplus \mathrm{SHR}^{\mathrm{f}_9(\mathrm{x})}(\mathrm{x}) \tag{4}$$

  , where $f_7(n) = -\frac{3}{16}n + 13(\mathrm{mod}\, n)$, $f_8(n) = -\frac{5}{16}n + 28(\mathrm{mod}\, n)$, and $f_9(n) = \frac{1}{8}n - 1$.

- $\sigma_1^{\{224\}}(x), \sigma_1^{\{256\}}(x), \sigma_1^{\{384\}}(x) = \sigma_1^{\{512\}}(x)$ are merged to

  $$g_6()=\mathrm{ROTR}^{\mathrm{f}_{10}(\mathrm{x})}(\mathrm{x}) \oplus \mathrm{ROTR}^{\mathrm{f}_{11}(\mathrm{x})}(\mathrm{x}) \oplus \mathrm{SHR}^{\mathrm{f}_{12}(\mathrm{x})}(\mathrm{x}) \tag{5}$$

  , where $f_{10}(n) = \frac{1}{16}n + 15(\mathrm{mod}\, n)$, $f_{11}(n) = \frac{21}{16}n - 23(\mathrm{mod}\, n)$, and $f_{12}(n) = -\frac{1}{8}n + 14(\mathrm{mod}\, n)$.

G. Message Schedule

In SHA-224 and SHA-256, the padded message is parsed into N 512-bit blocks, $M^{(1)} \ldots M^{(N)}$, for each 512-bit block, $M^{(i)}$, which is divided into 16 32-bit blocks, $M_0^{(i)} \ldots M_{15}^{(i)}$. In SHA-384 and SHA-512, for each 1024-bits block, $M^{(i)}$, which is divided into 16 64-bit blocks, $M_0^{(i)} \ldots M_{15}^{(i)}$. The message schedule $\{W_t\}$ is implemented as following.

$$\mathrm{W_t} = \begin{cases} \mathrm{M_t^{(i)}} & 0 \le \mathrm{t} \le 15 \\ \mathrm{g_5(W_{t-15}) + g_6(W_{t-2}) + W_{t-16} + W_{t-7}} & 16 \le \mathrm{t} \le \mathrm{f_{13}(n)} - 1 \end{cases} \tag{6}$$

,where $f_{13}(n) = \frac{1}{2}n + 48$, and the addition(+) is performed modulo 2^n.

H. For-Loop Operation.

The paper generalizes the for-loop operation of SHA-*mn*, which is the core part of SHA family algorithms. For each message block $M^{(i)}$, i = 1, 2, …,N, should be executed $f_{13}(n)$ rounds. Notice that addition (+) is performed modulo 2^n.

Algorithm For-Loop-Operations

For t = 0 to ($f_{13}(n)$-1) {

$T_1 = a_7 + g_4(a_4) + g_1(a_4, a_5, a_6) + K_t(mn) + W_t$;

$T_2 = g_3(a_0) + g_2(a_0, a_1, a_2)$;

$a_7 = a_6$; $a_6 = a_5$; $a_5 = a_4$; $a_4 = a_3 + T_1$;

$a_3 = a_2$; $a_2 = a_1$; $a_1 = a_0$; $a_0 = T_1 + T_2$;

}

I. Compute the ith Intermediate Hash Value H(i)

For each 16×n-bit block $M^{(i)}$, i = 1,2…N, the intermediate message digests in the SHA standard execute the following operations:

$$H_j^{(i)} = a_j + H_j^{(i-1)},\ 0 \le j \le 7 \tag{7}$$

J. The Message Digest.

After repeating steps N times (i.e., After processing $M^{(N)}$), the m×n bits message digest of the message is:

$$H_0^{(N)} \parallel H_1^{(N)} \parallel \ldots \parallel H_{m-1}^{(N)}. \tag{8}$$

III. SHA(x) FAMILY

The paper reduces the parameter n = {32, 64} as n = 32×(2-i), $i \in \{0, 1\}$ and replaces the generalized SHA as SHA(x) which is the family of SHAs of x. The section discusses SHA(1) and SHA(2) first and the LHV (Length-of-the-Hash-Value) problem of SHA(x).

Definition *SHA*(x)

SHA(x) = {*SHA-m*[32×(2x-i)] | $i \in \{0, 1\}$ and $m \in \{6, 7, 8\}$}

A. SHA(1) and SHA(2).

According to the Definition, SHA(1) = {SHA-192 , SHA-224, SHA-256, SHA-384, SHA-448, SHA-512}. And SHA(2) ={SHA-576 , SHA-672, SHA-768, SHA-896, SHA-1024} for x=2. The number of SHA(2) elements is 5, because m×n = 768 when n = 96, m = 8 and n = 128, n = 6. Therefore, we only use SHA-768 to denote the two cases. The elements of SHA(2) are listed in Table II.

TABLE II. SHA(2)

SHA(2)	SHA-576	SHA-672	SHA-768	SHA-768	SHA-896	SHA-1024
m	6	7	8	6	7	8
n (bits)	96	96	96	128	128	128
m×*n (bits)*	576	672	768	768	896	1024

B. Length-of-the-Hash-Value Problem

From the cases of SHA-768 and SHA-1536, which is $m \times n = 1536$ when n=256, $m=6$ and $n=192$, $n=8$, it exists LHV problem that some SHA-*r* cannot be expressed as $r=mn$ uniquely. The LHV problems are classified into 6-7-8-LHV problem, 6-7-LHV problem, 6-8-LHV problem and 7-8-LHV problem. The section defines the LHV problem. Otherwise, SHA- *r* has no LHV problem.

Definition *LHV* (*Length-of-the-Hash-Value*) *problem*

(1) Let SHA-*r* have a LHV problem if *r* satisfies
$\{r = m \times n = m' \times n' \mid \exists \text{ distinct } n, n' \in \{32(2x\text{-}i) \mid x \in \mathrm{N}, i = \{0, 1\}\}, \forall\, m, m' \in \{6,7,8\}\}$

(2) Let SHA-*r* have 6-7-8-LHV problem if *r* satisfies
$\{r = 6\times n = 7\times n' = 8\times n'' \mid \exists \text{ distinct } n, n', n'' \in \{32(2x\text{-}i) \mid x \in N, i = \{0, 1\}\}\}$

(3) Let SHA-*r* have 6-7-LHV problem if *r* satisfies
$\{r = 6\times n = 7\times n' \neq 8\times n'' \mid \exists \text{ distinct } n, n', n'' \in \{32(2x\text{-}i) \mid x \in N, i = \{0, 1\}\}\}$

(4) Let SHA-*r* have 6-8-LHV problem if *r* satisfies
$\{r = 6\times n = 8\times n' \neq 7\times n'' \mid \exists \text{ distinct } n, n', n'' \in \{32(2x\text{-}i) \mid x \in N, i = \{0, 1\}\}\}$

(5) Let SHA-*r* have 7-8-LHV problem if *r* satisfies
$\{r = 7\times n = 8\times n' \neq 6\times n'' \mid \exists \text{ distinct } n, n', n'' \in \{32(2x\text{-}i) \mid x \in N, i = \{0, 1\}\}\}$

IV. SHA′(*X*) WITHOUT LHV PROBLEM

The previous section defines LHV problem within all the situations of the LHV problem within SHA(x). Consider the length of one word in the form of $32\times2^{k-1}$ for $k \in N$, if we take $m = 6$, $n = 32\times23 = 256$, $r = m\times n = 1536$. However, $n' = 1536/8 = 192 \notin \{32\times2^{k-1} \mid k \in N\}$. That is, $\nexists\, n \in \{32\times2^{k-1} \mid k \in N\}$ such that $1536 = 6\times256 = 8\times n$. We solve 6-8-LHV problem. The found SHA-$m\times(32\times2^{k-1})$ has no LHV problem and is defined as SHA′(x). Therefore, we have SHA′(x) = $\{\text{SHA-192} \times 2^{x-1}, \text{SHA-224} \times 2^{x-1}, \text{SHA-256} \times 2^{x-1} \mid x \in N\}$.

Definition *SHA′*(*x*) *without LHV problem*
SHA′(x) = $\{\text{SHA-}m[32\times(2x\text{-}1)] \mid x \in N \text{ and } m \in \{6, 7, 8\}\}$

V. CONCLUSIONS

This work generalized the SHA family as SHA-*mn* that takes arbitrary length message as input to generate a message digest with required length. We modify each of the steps of SHA-*mn* as generalized version that contains padding and parsing; setting the initial hash values, constants, Boolean expressions and functions and message schedule; initializing the eight working variables and for-loop operation; and, computing the i^{th} intermediate hash values. Further, the LHV problem that does not exist in the original SHA standard is solved.

Owing to security considerations, SHA-*mn* is generalized based on the rules of SHA family design. Although many may not agree the method for calculating complexity according to the birthday paradox as the collision of full SHA-1 has been found in 2005, the design of SHA is improved. Efficient ways of finding collisions of SHA-256 remain the focus of many researchers to date. We believe, therefore, the approximate complexity of SHA-*mn* is $2^{mn/2}$ under birthday attack.

ACKNOWLEDGMENT

This work was supported in part by the National Science Council under the grant NSC99-2221-E029-034-MY3.

REFERENCES

[1] National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publications *FIPS PUB* 180, May. 1993.

[2] National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publications *FIPS PUB* 180-1, 1995.

[3] National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publications *FIPS PUB* 180-2", 2001.

[4] National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publications *FIPS PUB* 180-2", 2002.

[5] National Institute of Standards and Technology, " Data encryption standard (DES)," Federal Information Processing Standards Publications *FIPS PUB* 46-3", 1999.

[6] Pinakpani Pal, and Palash Sarkar, "PARSHA-256 – A New Parallelizable Hash Function and a Multithreaded Implementation," *Proc. of Fast Software Encryption* 2003 (*FSE*2003), *LNCS* 2887, pp. 347–361, Springer-Verlag, 2003.

[7] H. Handschuh and D. Naccache, "SHACAL," NESSIE, 2001. Archive available at https://www.cosic.esat.kuleuven. be/nessie/tweaks.html

[8] Jiqiang Lu1, Jongsung Kim, Nathan Keller, and Orr Dunkelman, "Related-Key Rectangle Attack on 42-Round SHACAL-2," *Proc. of 9th Information Security Conference* (*ISC*2006), *LNCS* 4176, pp. 85–100, Springer-Verlag, 2006.

[9] Hirotaka Yoshida and Alex Biryukov, "Analysis of a SHA-256 Variant," *Proc. of 12th Annual Workshop on Selected Areas in Cryptography* (*SAC*2005), *LNCS* 3897, pp. 245–26, Springer-Verlag, 2006.

[10] E. Biham, R. Chen, A. Joux, P. Carribault, W. Jalby and C. Lemuet, "Collisionsion SHA-0 and Reduced SHA-1," *Proc. of Eurocrypt'05*, pp.36-57, May 2005.

[11] Rijmen, V.; Oswald, M. E., "Update on SHA-1," *Proc. of RSA* 2005, *LNCS* 3376, pp. 58-71, 2005.

[12] Xiaoyun Wang, Hongbo Yu and Yiqun Lisa Yin, "Finding Collision in the Full SHA-1," *Proc. of Crypto'05*, *LNCS* 3621, pp.17-36, 2005.

[13] Gilbert, H., Handschuh, H. "Security Analysis of SHA-256 and Sisters," *Proc. of 6th Information Security Conference* (*SAC*2003), *LNCS* 3006, pp. 175–193, Springer-Verlag, 2004.

[14] Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V., "Analysis of Step-Reduced SHA-256," *Proc. of 13th annual Fast Software Encryption workshop* (*FSE*2006), *LNCS* 4047, pp. 126–143, Springer-Verlag, 2006.

[15] Ivica Nikoli´c and Alex Biryukov, "Collisions for Step-Reduced SHA-256," *Proc. of 15th annual Fast Software Encryption workshop* (*FSE*2008), *LNCS* 5086, pp. 1–15, Springer-Verlag, 2008.