JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 29, 889-905 (2013)

Finding Near-Optimum Message Scheduling Settings for SHA-256 Variants Using Genetic Algorithms

CHU-HSING LIN[1], CHEN-YU LEE[2], KRISHNA M. KAVI[3], DENG-JYI CHEN[2] AND YI-SHIUNG YEH[2]
[1]*Department of Computer Science*
Tunghai University
Taichung, 407 Taiwan
[2]*Department of Computer Science*
National Chiao Tung University
Hsinchu, 300 Taiwan
[3]*Department of Computer Science and Engineering*
University of North Texas
Denton, TX 76203, USA

One-way hash functions play an important role in modern cryptography. Matusiewicz *et al.* proved that the message scheduling is essential for the security of SHA-256 by showing that it is possible to find collisions with complexity 2^{64} hash operations for a variant without it. In this article, we first proposed the conjecture that message scheduling of SHA algorithm has higher security complexity (or fitness value in Genetic algorithm) if each message word (W_i) involves more message blocks (M_i) in each round. We found some evidence supports the conjecture. Consider the security of SHA-0 and SHA-1. Since Chabaud and Joux shown that SHA-1 is more secure than SHA-0. Further, Wang found collisions in full SHA-0 and SHA-1 hash operations with complexities less than 2^{39} and 2^{69}, respectively. We found it is consistent from the viewpoint of message blocks (terms) involved in each message word. It clearly shown that the number of terms involved in SHA-1 is more than that in SHA-0, taking W^{27} as an example, 14 and 6, respectively. Based on the conjecture we proposed a new view of complexity for SHA-256-XOR functions, a variant of SHA-256, by counting the terms involved in each equation, instead of analyzing the probability of finding collisions within SHA-256-XOR hash function. Our experiments shown that the parameter set in each equation of message schedule is crucial to security fitness. We applied genetic algorithms to find the near-optimal message schedule parameter sets that enhance the complexity 4 times for SHA-1 and 1.5 times for SHA-256-XOR, respectively, when compared to original SHA-1 and SHA-256-XOR functions. The analysis would be interesting for designers on the security of modular-addition-free hash function which is good for hardware implementation with lower gate count. And the found message schedule parameter sets would be a good reference for further improvement of SHA functions.

Keywords: genetic algorithms, cryptography, secure hash algorithm, message scheduling, optimisation

1. 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, and digital signatures. The hash function uses a string of arbitrary length as its input, and

Received July 8, 2011; revised July 23, 2012; accepted September 11, 2012.
Communicated by Vincent Rijmen.

creates a fixed-length string as its output. A hash value is often called a data *fingerprint* or *message digest*.

Definition 1 [1] (Collision-free hash function) A *collision-free hash function H* uses a message M of arbitrary length as its input, and produces a fixed-length message digest when it satisfies the following conditions:

- The description of $H(M)$ is publicly known and it is easy to implement
- Pre-image resistant: Given message digest y, it is difficult to find a message M such that $H(M) = y$.
- Second pre-image resistant: Given M and its image $H(M)$, it is difficult to find another M' such that $H(M) = H(M')$.
- (Strong) Collision Resistance: It is difficult to find two distinct messages M and M' such that $H(M) = H(M')$.

The Secure Hash Algorithm (SHA) is a series of cryptographic hash functions published by the US National Institute of Standards and Technology (NIST). NIST proposed the SHA-0 as a Federal Information Processing Standard Publication (FIPS PUB) 180 in 1993 [2]. In 1995, NIST announced a revised version, the SHA-1, in FIPS PUB 180-1 [3] as a standard to replace the SHA-0. In 2001, the NIST published SHA-2 as FIPS PUB 180-2 [4], which consisted of four algorithms: SHA-1, SHA-256, SHA-384, and SHA-512. Table 1 lists the characteristics of the five SHA-2 algorithms.

Table 1. SHA algorithms.

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Security
SHA-1	$< 2^{64}$	512	32	160	2^{80}
SHA-224	$< 2^{64}$	512	32	224	2^{112}
SHA-256	$< 2^{64}$	512	32	256	2^{128}
SHA-384	$< 2^{128}$	1024	64	384	2^{192}
SHA-512	$< 2^{128}$	1024	64	512	2^{256}
The term security in this table means that a birthday attack [11] [Remark 1] on a message digest of size n produces a collision with a factor of approximately $2^{n/2}$.					

Recent studies have proposed extensions based on SHA. For example, RARSHA-256 [7] [Remark 1] is composed of the SHA-256 compression function, and is faster than SHA-256 when implemented in parallel. SHACAL and SHACAL-2 [8, 9] are block ciphers that are based on SHA-1 and SHA-256, respectively, and which were submitted to the New European Schemes for Signatures, Integrity, and Encryption project in 2003. Yoshida and Biryukov replaced all arithmetic additions with XOR operations in SHA-256, naming it SHA-256-XOR, and found that SHA-2-XOR has a pseudo-collision resistance weakness up to 34 rounds [10].

A birthday attack [11, 12] is a type of cryptographic attack based on the birthday problem in probability theory. Given a function f, the attack attempts to find two different inputs x_1, x_2 such that $f(x_1) = f(x_2)$. Such a pair (x_1, x_2) is called a collision input. The birthday attack on a message digest of size n produces a collision after trying $1.2 \times \sqrt{2^n} \approx 2^{n/2}$

input values. Under the birthday attack, the security of SHA-1, SHA-192, SHA-224, SHA-256, SHA-384, SHA-448, and SHA-512 are approximately 2^{80}, 2^{96}, 2^{112}, 2^{128}, 2^{192}, 2^{224}, and 2^{256}, respectively, and are listed in Table 1. Many researchers have tried to develop a cryptanalytic method with a lower complexity than the birthday attack.

In 1998, Chabaud and Joux announced a method for finding the SHA-0 collisions [19]. They reduced this complexity to 2^{61} using a differential cryptanalysis technique, but they could not successfully apply it to SHA-1. This result implied that SHA-1 is more secure than SHA-0. In early 2005, Rijmen and Oswald applied the same method to find collisions in SHA-1 [13]. They examined message scheduling in SHA-0 and SHA-1, and proved that the complexity associated with 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 [14]. In 2010, Grechnikov announced the practical collision attack on the 73-step SHA-1 based on an automated approach [26]. NIST announced that SHA-1 will be used until 2010, at which time it will be replaced by SHA-2.

Since 2004, several authors have reported on collisions for SHA-256. Gilbert and Handschuh reported a 9-round local collision with a complexity of 2^{66} using differential path analysis [15]. Mendel *et al.* later reduced this complexity to 2^{39} [16]. Nikolić and Biryukov realized 21-step collisions for SHA-256 using a nonlinear differential path analysis with a complexity of 2^{19} [17]. In 2008, Sanadhya and Sarkar found a local collision with 24-step SHA-256 and SHA-512 with $2^{28.5}$ and $2^{32.5}$ calls, respectively [18], and this was the first time that a colliding message pair for 24-step SHA-512 was provided. In 2009, Indesteege *et al.* found collisions on the 24-step SHA-256 and SHA-512 with $2^{28.5}$ calls and 2^{53} calls, respectively, and a local collision on 31-step SHA-256 with 2^{32} [24]. Also in 2009, Aoki *et al.* presented full preimage attacks on up to 43-step SHA-256 and SHA-512 with the time complexities of $2^{254.9}$ and $2^{511.5}$ compression function operations for full preimages, respectively [25]. Since 2011, Mendel *et al.* have presented a collision on 27-step SHA-256 and a semi-free-start collision on 32-step SHA-256 with practical complexity [27]. Biryukov1 *et al.* presented a second-order differential collision for the SHA-256 compression function on 47 out of 64 steps, which have practical complexity based on a rectangle/boomerang approach [28].

Almost all of the currently known cryptanalyses of SHA have attempted to find collisions on a differential path. However, the design of each component such as algorithms for message scheduling and hash loop body and the function parameters, affects the possibility that a path for collisions (using differential path cryptanalysis) will be found. A fairly large body of literature exists regarding methods of improving hash algorithms. However, there is a surprising lack of information regarding the design and selection of function parameters. This paper addresses this deficiency.

The purpose of the research presented in this article is to examine the relationship between the security of a hash function and its function parameters. In this regard, two issues that need to be resolved are (a) how to assess the security fitness of a given set of function parameters, and (b) how to find the optimal function parameter set. Specifically, this paper proposes a novel view of complexity (hence security fitness) of SHA-2-XOR functions proposed in [10], by counting the terms involved in each equation, instead of analyzing the probability of finding collisions within an SHA-256-XOR hash function. Our experiments have shown that the parameter set in each equation of a message sched-

ule plays an important role in security fitness, but it is very hard to find the optimum parameter values. We apply genetic algorithms to find the optimal message schedule parameter sets that enhance the complexity 4 times for SHA-1 and 1.5 times for SHA-256-XOR, when compared to original SHA-1 and SHA-256-XOR functions. The analysis results would be interesting for designers who are interested in the security of modular addition free hash functions, which are good for hardware implementation with lower gate counts. Moreover, the found message schedule parameter sets would be a good reference for further improvement of SHA functions.

The remainder of this paper is organized as follows. Section 2 briefly introduces SHA-, SHA-256, and genetic algorithms. Section 3 proposes our security evaluation criterion for SHA message scheduling, and finds the best parameter sets for SHA-1 using a brute force approach. Section 4 applies it to find the nearly optimal set for SHA-256-XOR and describes the experimental results. Section 5 discusses the results and concludes the paper. Table 2 lists the nomenclature used throughout the paper.

Table 2. Legends.

Symbol	Definition
M	Message with arbitrary length as the input of a hash function.
W_t	The tth message word.
m	The number of output words.
n	The length of one word.
l	The length of the input message, $l = \|M\|$.
$ROTL^{\{i\}}()$	Left-rotation operation for i bits.
r	The value of $m \times n$, $r = m \times n$.
$H(M)$	The hash function $H()$ with input M.
$\{t, A, B, C, D\}$	The parameter set of W_t equation in message scheduling.
$SHR^{\{i\}}()$	Right-shift operation for i bits.
$M^{(i)}$	Message block i with a size of 512 bits.
$M_j^{(i)}$	The jth word of the i^{th} message block.
M_j^n	M_j^n indicates message word M_j doing n-bitwise rotation.

2. RELATED WORKS

2.1 Overview of SHA-1 and SHA-256 Algorithms

SHA-1 [4] takes a message M with a length of l bits, where $0 \le l < 2^{64}$, as the input, and outputs a 160-bit hash value. The hash function parses the padded message into 512-bit blocks. Each block passes an 80-round compression function and outputs a 160-bit hash value.

SHA-1 processing involves the following 3 steps:

Step 1: Padding message: pad the input message making it a multiple of 512 bits.

Step 2: Parsing the padded message: parse the padded message into N 512-bit blocks,

$M^{(1)}$, $M^{(2)}$, …, $M^{(N)}$. Each block $M^{(i)}$ is divided into sixteen 32-bit words, $M_0^{(i)}$, $M_1^{(i)}$, …, $M_{15}^{(i)}$.

Step 3: Message scheduling for each message block $M^{(i)}$.

- The message words, $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)}, & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}), & 16 \le t \le 79 \end{cases} \quad (1)$$

  where $ROTL^{\{i\}}$ indicates left rotation operation by i bits.

- Message expansions are performed for 80 rounds. Algorithm 1 defines these steps in detail. Table 3 summarizes the Boolean function f_t that appeared in the SHA-1 step function.

Algorithm 1: SHA-1 step function

1: **FOR** t = 1 to 80
2: $e_t = d_{t\text{-}1}$
3: $d_t = c_{t\text{-}1}$
4: $c_t = \text{ROTL}^{30}(b_{t\text{-}1})$
5: $b_t = a_{t\text{-}1}$
6: $a_t = \text{ROTL}^5(a_{t\text{-}1}) + f_t(b_{t\text{-}1}, c_{t\text{-}1}, d_{t\text{-}1}) + e_{t\text{-}1} + K_t + W_{t\text{-}1}$
7: **End FOR**

Table 3. Boolean function used in SHA-1.

Round t	Boolean function $f_t(x, y, z)$
$01 \le t \le 20$	$(x \wedge y) \vee (\neg x \wedge z)$
$21 \le t \le 40$	$x \oplus y \oplus z$
$41 \le t \le 60$	$(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$
$61 \le t \le 80$	$x \oplus y \oplus z$

SHA-256 takes a message M with a length of l bits, where $0 \le l < 2^{64}$, as the input, and outputs a 256-bit hash value. The hash function parses the padded message into 512-bit blocks. Each block passes a 64-round compression function and outputs a 256-bit hash value.

The SHA-256 contains steps that are similar to SHA-1, except that it sets different initial values and constants, and uses different functions. The following is a description of the message block processing step.

Step 4: Message scheduling for each message block $M^{(i)}$.

- The message words, $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)} & , 0 \le t \le 15 \\ \sigma_0^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_1^{\{256\}}(W_{t-15}) + W_{t-16} & , 16 \le t \le 63 \end{cases} \quad (2)$$

$$\sigma_0^{\{256\}}(x) = ROTL^7(x) \oplus ROTL^{18}(x) \oplus SHR^3(x)$$
$$\sigma_1^{\{256\}}(x) = ROTL^{17}(x) \oplus ROTL^{19}(x) \oplus SHR^{10}(x) \quad (3)$$

where $SHR^{\{i\}}$ indicates right shift operation by i bits.

- Message expansions are performed for 64 rounds. Algorithm 2 defines these steps in detail. Table 4 summarizes the Boolean function f_t used in each round.

Algorithm 2: SHA-256 step function

1: **FOR** $t = 1$ to 64
2: $T_1 = h_{t-1} + f_1(e_{t-1}) + f_3(e_{t-1}, f_{t-1}, g_{t-1}) + K_t + W_{t-1}$
3: $T_2 = f_2(a_{t-1}) + f_4(a_{t-1}, b_{t-1}, c_{t-1})$
4: $h_t = g_{t-1}$
5: $g_t = f_{t-1}$
6: $f_t = e_{t-1}$
7: $e_t = d_t + T_1$
8: $d_t = c_{t-1}$
9: $d_t = c_{t-1}$
10: $c_t = b_{t-1}$
11: $b_t = a_{t-1}$
12: $a_t = T_1 + T_2$
13: **End FOR**

Table 4. Boolean function used in SHA-256.

Boolean function f_t
$f_1(x) = \text{ROTL}^{(2)}(x) \oplus \text{ROTL}^{(13)}(x) \oplus \text{ROTL}^{(22)}(x)$
$f_2(x) = \text{ROTL}^{(6)}(x) \oplus \text{ROTL}^{(11)}(x) \oplus \text{ROTL}^{(25)}(x)$
$f_3(x) = (x \wedge y) \oplus (\neg x \wedge z)$
$f_4(x) = (x \wedge y) \oplus (x \wedge z) \vee (y \wedge z)$

2.2 Genetic Algorithm

The genetic algorithm is the most popular type of evolutionary algorithm that use techniques inspired by evolutionary biology. As stated by John H. Holland in 1975, "The genetic algorithm has a wide scope of applications, including economics, engineering, machine learning, genome biology, game theory, neural networks, etcetera" [23]. A genetic algorithm provides a highly efficient method for ensuring convergence to near-optimal or optimal solutions.

Fig. 1 shows the steps of the genetic algorithm, which are described as follows:

(1) Initialization of population.
(2) Choice of a fitness function and evaluation of the fitness value of each individual in the population.
(3) Selection of better ranked part to be reproduced.
(4) Breeding new generation's population by crossover and mutation.

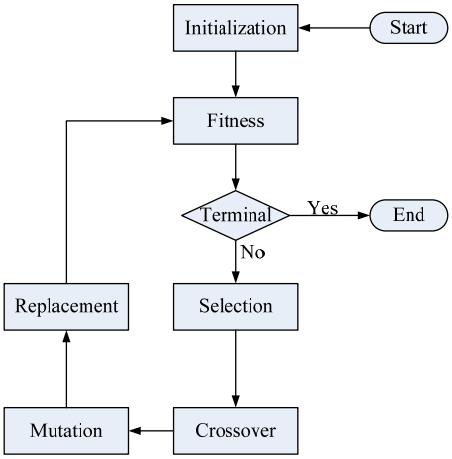

Fig. 1. Flowchart of genetic algorithm.

(5) Replacement of the worst ranked part of the population with the new generation's population.
(6) Repeating this generational process until the termination condition has been reached.

The Genetic Algorithm Utility Library (GAUL) developed by AI Foundry [22] is a flexible programming library designed to aid in the development of applications that use genetic or evolutionary algorithms. It provides data structures and functions for handling and manipulating the data required for serial and parallel evolutionary algorithms.

GAUL is an open-source programming library, which was released under the GNU General Public License. It is designed to assist in the development of code that requires evolutionary algorithms.

3. SHA MESSAGE SCHEDULING EVALUATION CRITERION

This section proposes an evaluation criterion of SHA message scheduling. The number of terms involved in the message schedule is treated as an evaluation criterion of SHA message scheduling. This study uses SHA-0 and SHA-1 as examples to show that SHA-1 is more secure than SHA-0 by comparing their message scheduling equations.

3.1 Local Collision

A local collision appearing on all the SHA families is a collision within intermediate steps of the hash function [14]. The starting point for hash function collision attacks is a local collision. Local collisions are found using linear approximations of Boolean functions that are used in various rounds in message scheduling (and other conditions as defined in [14]). The first observation is that SHA-0 has a 6-step local collision that can start at any step i. The differential path is a sequence of grouped local collisions with possible overlaps [20]. Wang [14] tried to find a set of starting steps for each local collision to construct such a path. The disturbance vector is applied to satisfy the recursion

defined by the message expansion. Once a local collision is found, an attempt is made to consider the message expansion and other non-linear designs to find a collision for the full hash function. For SHA-0, 3 vectors are found successfully for three conditions in [14]. However, it is more complicated to find a good disturbance vector due to the large search space on SHA-1, and the probability of n interleaved local collision complexities increases exponentially with n for SHA-256 [16].

Mendel provides an approach for collision searches as follows [16]:

(1) Identify local collisions in each round of transformation.
(2) Search for disturbance vectors that need to satisfy some additional properties.
(3) Build the difference vector by interleaving the local collisions.
(4) The complexity of the collision search is related to the characteristic within these interleaved local collisions.
(5) Adjusting message bits for the chosen characteristic reduces the computational cost for the collision search.

The issue that arises is how to reduce the number of local collisions in an expansion process. Our study applies a genetic approach to find the optimal parameter set of the SHA family message expansion function based on the evaluation criterion with the lowest number of local collisions.

3.2 Local Collision in SHA-0 and SHA-1

In [19], it is pointed out that SHA-1 is safer than SHA-0 because of a single bitwise rotation in SHA-1 that affects the local collisions existing in SHA-0. Table 5 shows the SHA-0 and SHA-1 equations.

Table 5. SHA-0, SHA-1, and SHA-256-XOR equations.

Algorithm	Equation
SHA-0	$W_t = \begin{cases} M_t^{(i)} & , 0 \le t \le 15 \\ W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16} & , 16 \le t \le 79 \end{cases}$
SHA-1	$W_t = \begin{cases} M_t^{(i)} & , 0 \le t \le 15 \\ ROTL^1\left(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}\right) & , 16 \le t \le 79 \end{cases}$
SHA-256-XOR	$W_t = \begin{cases} M_t^{(i)} & , 0 \le t \le 15 \\ \sigma_0^{\{256\}}\left(W_{t-2}\right) \oplus W_{t-7} \oplus \sigma_1^{\{256\}}\left(W_{t-1}\right) \oplus W_{t-16} & , 16 \le t \le 63 \end{cases}$

The following are examples that compare the terms involved in W_{27} in both SHA-0 and SHA-1, and that in W_{20} in SHA-256-XOR where M_j^n (or W_j^n) indicates that the message block M_j (or intermediate message word W_j) undergoes an n-bitwise left rotation. Each message word W_t is obtained by recursively computing other words with lower indices and being replaced by message blocks until $t \le 15$.

Fig. 2 represents the number of terms involved in full SHA-0, SHA-1, and SHA-256-XOR.

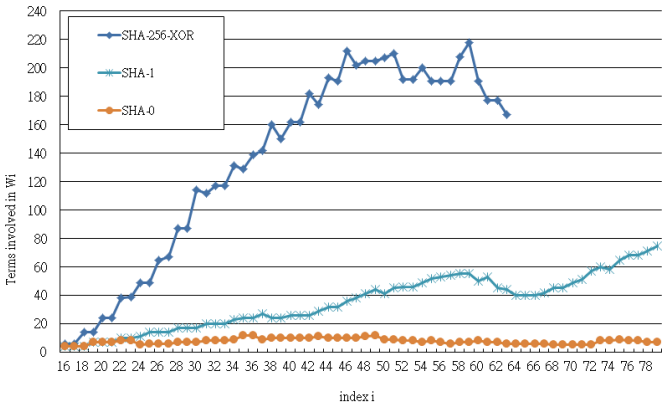

Fig. 2. Comparison of the number of terms involved in each W_t in message scheduling for SHA-0, SHA-1 and SHA-256.

[SHA-0]

$$\begin{aligned} W_{27} &= W_{24} \oplus W_{19} \oplus W_{13} \oplus W_{11} \\ &= (W_{21} \oplus W_{16} \oplus W_{10} \oplus W_8) \oplus W_{19} \oplus W_{13} \oplus W_{11} \\ &= \ldots \\ &= M_{15} \oplus M_4 \oplus M_2 \oplus M_7 \oplus M_8 \oplus M_3 \\ &\Rightarrow \text{ 6 terms are involved.} \end{aligned}$$

[SHA-1]

$$\begin{aligned} W_{27} &= {W_{24}}^1 \oplus {W_{19}}^1 \oplus {W_{13}}^1 \oplus {W_{11}}^1 \\ &= ({W_{21}}^2 \oplus {W_{16}}^2 \oplus {W_{10}}^2 \oplus {W_8}^2) \oplus {W_{19}}^1 \oplus {W_{13}}^1 \oplus {W_{11}}^1 \\ &= \ldots \\ &= {M_{15}}^4 \oplus {M_{10}}^4 \oplus {M_4}^4 \oplus {M_2}^4 \oplus {M_{13}}^3 \oplus {M_7}^3 \oplus {M_5}^3 \oplus {M_{10}}^2 \oplus {M_8}^2 \oplus {M_{11}}^2 \oplus \\ &\quad {M_5}^2 \oplus {M_3}^2 \oplus {M_{13}}^1 \oplus {M_{11}}^1 \\ &\Rightarrow \text{ 14 terms are involved.} \end{aligned}$$

[SHA-256-XOR]

$$\begin{aligned} W_{20} &= \sigma_0(W_{18}) \oplus W_{13} \oplus \sigma_1(W_{15}) \oplus W_4 \\ &= {W_{18}}^7 \oplus {W_{17}}^{18} \oplus W_{13} \oplus {W_5}^{17} \oplus {W_5}^{19} \oplus W_4 \\ &= ({W_{14}}^{14} \oplus {W_{14}}^{25} \oplus {W_9}^7 \oplus {W_1}^{24} \oplus {W_1}^{26} \oplus {W_6}^7 \oplus {W_{14}}^4 \oplus {W_9}^{18} \oplus {W_1}^3 \oplus \\ &\quad {W_5}^5 \oplus {W_0}^{18} \oplus W_{11} \oplus {W_3}^{17} \oplus {W_3}^{19} \oplus W_2)^7 \oplus ({W_{14}}^{14} \oplus {W_{14}}^{25} \oplus {W_9}^7 \oplus \\ &\quad {W_1}^{24} \oplus {W_1}^{26} \oplus {W_6}^7 \oplus {W_{14}}^4 \oplus {W_9}^{18} \oplus {W_1}^3 \oplus {W_5}^5 \oplus {W_0}^{18} \oplus W_{11} \oplus {W_3}^{17} \\ &\quad \oplus {W_3}^{19} \oplus W_2)^{18} \oplus {W_5}^{17} \oplus {W_3}^{19} \oplus W_4 \\ &= {M_{14}}^{21} \oplus {M_{14}}^{32} \oplus {M_9}^{14} \oplus {M_1}^{31} \oplus {M_1}^1 \oplus {M_0}^{14} \oplus {M_{11}}^7 \oplus {M_3}^{24} \oplus {M_3}^{26} \oplus \\ &\quad {M_2}^7 \oplus {M_{14}}^{11} \oplus {M_{14}}^{22} \oplus {M_9}^4 \oplus {M_1}^{21} \oplus {M_0}^0 \oplus {M_{11}}^{18} \oplus {M_3}^3 \oplus {M_3}^5 \oplus {M_2}^{18} \\ &\quad \oplus {M_5}^{17} \oplus {M_5}^{19} \oplus M_4 \\ &\Rightarrow \text{ 22 terms are involved.} \end{aligned}$$

3.3 Lee's Conjecture

Matusiewicz *et al.* proved that the functions (Σ and σ) or the message expansion are

essential for the security of SHA-256 by showing that it is possible to find collisions with a complexity of 2^{64} hash operations for a variant without them [29].

We proposed that message scheduling of the SHA algorithm has higher security complexity (or fitness) if each message word (W_t) involves more message blocks (M_i) in each round.

Chabaud and Joux showed that SHA-1 is more secure than SHA-0 [19]. Furthermore, Wang found collisions in full SHA-0 and SHA-1 hash operations with complexities less than 2^{39} [20] and 2^{69} [14], respectively.

Consider the analyses of the terms involved in each message block. Fig. 2 clearly shows that the number of terms involved in SHA-1 is more than that in SHA-0, taking W_{27} as an example (14 > 6). Therefore, SHA-1 has a higher security complexity (hence security fitness) than SHA-0. In this paper, we use the term "security fitness" to evaluate the security of each possible W_t in message scheduling.

3.4 The Best Setting of Message Scheduling Equation in SHA-1

The message scheduling equation in SHA-1 can be generalized as

$$W_t = \begin{cases} M_t^{(i)} & , 0 \le t \le 15 \\ ROTL^1\left(W_{t-A} \oplus W_{t-B} \oplus W_{t-C} \oplus W_{t-D}\right) & , 16 \le t \le 79 \end{cases}. \tag{4}$$

The best four variables are produced by the brute force (or exhaustive) approach, and the values found are $\{A, B, C, D\} = \{1, 2, 11, 16\}$. The best complexity occurs in round 60 when 212 terms are involved. The modified equation is

$$W_t = \begin{cases} M_t^{(i)} & , 0 \le t \le 15 \\ ROTL^1\left(W_{t-1} \oplus W_{t-2} \oplus W_{t-11} \oplus W_{t-16}\right) & , 16 \le t \le 79 \end{cases} \tag{5}$$

and called optSHA-1 [21]. Fig. 3 compares the number of terms involved in SHA-1 and optSHA-1.

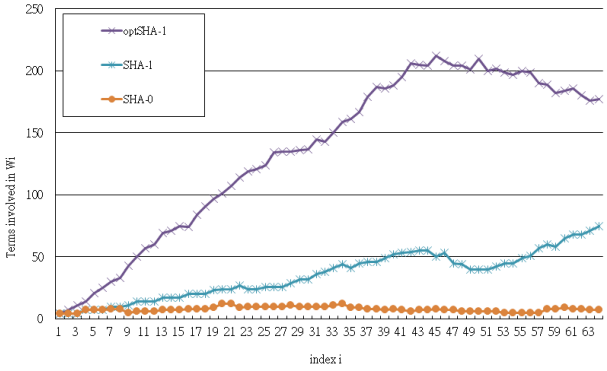

Fig. 3. Comparison of the number of terms involved in each W_t in message scheduling for SHA-1 and SHA-1-OPT.

4. IMPROVING SHA-256-XOR VIA GENETIC ALGORITHMS

4.1 Specialized GA for SHA-256-XOR

To find optimum parameters, the message scheduling equation in SHA-256-XOR can be generalized as

$$W_t = \begin{cases} M_t^{(i)} & ,0 \le t \le 15 \\ \sigma_0^{\{256\}}(W_{t-A}) \oplus W_{t-B} \oplus \sigma_1^{\{256\}}(W_{t-C}) \oplus W_{t-D} & ,16 \le t \le 63 \end{cases}, \quad (6)$$

and

$$\sigma_0^{\{256\}}(x) = ROTL^7(x) \oplus ROTL^{18}(x) \oplus SHR^3(x),$$
$$\sigma_1^{\{256\}}(x) = ROTL^{17}(x) \oplus ROTL^{19}(x) \oplus SHR^{10}(x). \quad (3)$$

Consider two operations, *ROTL* and *SHR*. A bitwise rotation operation, *ROTL*, is a circular shift operation that is a permutation of the entries in a tuple where the last element becomes the first element and all of the other elements are shifted. The shift operation, $SHR^n(x)$, which sets 0 as the first element, does not influence the experimental results because $SHR^n(x)$ and $ROTR^n(x)$ produce different results. Based on this assumption, the generalized form is modified to

$$\sigma_0^{\{256\}}(x) = ROTL^7(x) \oplus ROTL^{18}(x),$$
$$\sigma_1^{\{256\}}(x) = ROTL^{17}(x) \oplus ROTL^{19}(x). \quad (7)$$

In the previous section, the optimal values are calculated using the brute force approach in otpSHA-1. To find the optimum parameters using the brute force approach for SHA-256-XOR, we would need to test 2^{64} possible combinations of $\{A, B, C, D\}$ for each round t ($16 \le t \le 63$), and to perform up to 48×2^{94} operations in the whole experiment. We applied genetic algorithm operators of recombination and perturbation to reduce the number of infeasible solutions needed to find the near optimal variable set $\{A, B, C, D\}$.

The design of the GA involves some main components: genetic representation, population initialization, fitness function, selection scheme, crossover, and mutation. Each component is described as follows, and the parameters used with GAUL are listed in Table 6:

Table 6. Genetic algorithm parameters.

Parameter	Value
Library	GAUL
Population size	500
Number of chromosomes	1
Length of each chromosome	5
Evolutionary mode	GA_SCHEME_DARWIN
Elitism mode	GA_ELITISM_PARENTS_SURVIVE
Crossover ratio	0.9
Mutation ratio	0.1
Fitness function	# of terms involved in W_t equation

- Genetic Representation: The genes represent the input variables, A, B, C, D, t, of the generalized SHA-256-XOR, and each chromosome represents a possible solution. In the simulation, the length of each chromosome is 5.
- Population Initialization: Each chromosome presents a potential solution for the problem in genetic algorithms. The initial population is randomly generated and the size is set to 500.
- Fitness Function: The fitness function counts the number of terms in the equation for W_t. After the process of selection, crossover, and mutation, the optimal chromosome indicates the maximum number of terms involved in the equations.

$$F(t) = \#\text{ of different terms involved in } W_t \text{ equation} \tag{8}$$

- Selection Scheme: Selection is a genetic operator that chooses a chromosome from the current generation's population for inclusion in the next generation's population. We adopt the binary tournament selection based on the fitness value in the simulation.
- Crossover and Mutation: Crossover enables genetic algorithms to extract the best genes from different individuals, and to produce potentially superior children. The mutation operation randomly modifies the gene to prevent the falling of all solutions into a local optimum, and extends the search space. In the simulation, we adopt the one-point crossover with a ratio 0.9, and a single-point mutation with a ratio 0.1.

4.2 Experiment Results

Table 7 lists 10 generations of the simulation results for $\{A, B, C, D\}$. The simulation requires heavy computational times for each t. Appendix A lists more of our experimental results. We have not generated optimum parameters for additional rounds because of the computational requirements. However, we believe that we have demonstrated the basis of our contribution, which is a possible approach for the selection of optimal message scheduling parameters and the analysis of the security fitness.

Table 7. The last 10 generations of the simulation.

Generation	A	B	C	D	Fitness
41	8	1	1	16	238
42	4	1	1	16	259
43	4	1	1	16	265
44	4	1	1	16	265
45	4	1	1	16	265
46	4	1	1	16	270
47	4	1	1	16	270
48	4	1	1	16	270
49	4	1	1	16	270
50	4	1	1	16	270

The values for the 5 variables converge after 42 generations. It appears that the approximate optimal values are $\{A, B, C, D\} = \{4, 1, 1, 16\}$. Thus, the best equation for W_t of SHA-256-XOR, named optSHA-256-XOR, should be

$$W_t = \begin{cases} M_t^{(i)} & , 0 \le t \le 15 \\ \sigma_0^{\{256\}}(W_{t-4}) \oplus W_{t-1} \oplus \sigma_1^{\{256\}}(W_{t-1}) \oplus W_{t-16} & , 16 \le t \le 63 \end{cases}. \quad (9)$$

Fig. 4 compares SHA-256-XOR with optSHA-256-XOR by showing clearly that otpSHA-256-XOR is indeed more secure than SHA-256.

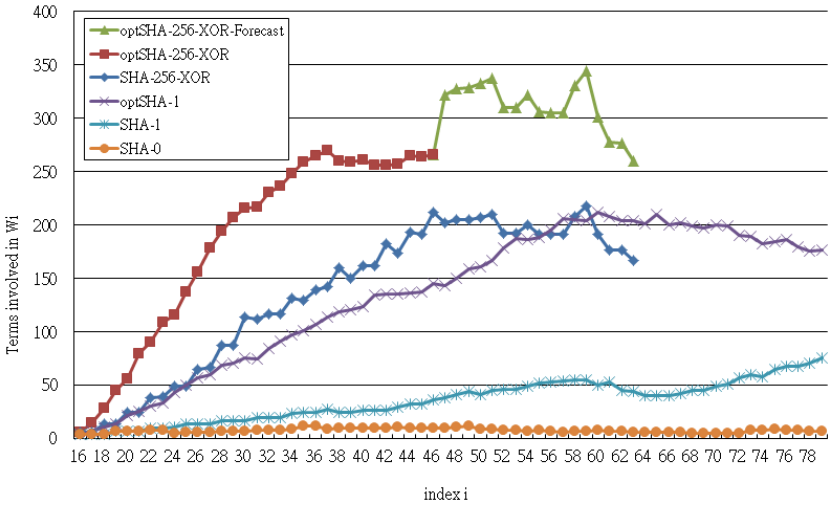

Fig. 4. Comparison of the number of terms involved in each W_t in message scheduling for SHA-256-XOR and optSHA-256-XOR.

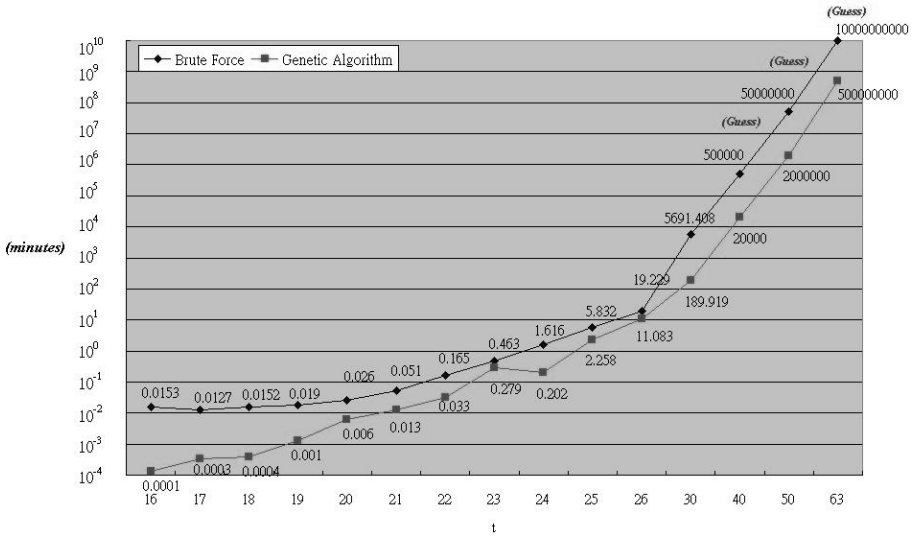

Fig. 5. Comparison of the running time in W_t between genetic algorithm and brute force.

With regards to the performance, Fig. 5 compares the running time for each W_t ($16 \le t \le 30$) in brute force and genetic algorithms. We have not yet been able to complete

the simulation for every W_t. The later items in the experiment will consume additional time because the equation is a recursive function.

5. CONCLUSIONS

Since its introduction in 1993, the secure hash function family has become an important standard in cryptography. This paper proposes a novel view of complexity (and hence security fitness) by counting the number of terms involved in each equation, instead of analyzing the probability of finding collisions within hash functions. This study identified the near optimal versions, optSHA-1 and optSHA-256-XOR, using brute force and genetic approaches of SHA-1 and SHA-256-XOR, respectively, and the latter was found to be more computationally efficient. This analysis would be interesting for designers interesting in the security of modular addition free hash functions, which are good for hardware implementation with lower gate counts. Furthermore, the obtained message schedule parameter sets will be a good reference for further improvement of SHA functions.

Finally, we again review the equations of SHA-256-XOR (Table 8). In our future studies, we aim to combine these new variables $\{E, F, G, H\}$ with $\{A, B, C, D\}$ to enhance the algorithm using genetic algorithms. Furthermore, this approach can be applied to other SHA-2-XOR family members to determine the relationship between the terms involved and the complexity (or security fitness).

Table 8. Generalized SHA-256-XOR with more variables.

Algorithm	Equation
Generalized SHA-256-XOR with more variables	$W_t = \begin{cases} M_t^{(i)} & , 0 \le t \le 15 \\ \sigma_0^{\{256\}}(W_{t-A}) \oplus W_{t-B} \oplus \sigma_1^{\{256\}}(W_{t-C}) \oplus W_{t-D} & , 16 \le t \le 63 \end{cases}$ $\sigma_0^{\{256\}}(x) = ROTL^E(x) \oplus ROTL^F(x) \oplus SHR^3(x)$ $\sigma_1^{\{256\}}(x) = ROTL^G(x) \oplus ROTL^H(x) \oplus SHR^{10}(x)$

REFERENCES

1. S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "Cryptographic hash functions: A survey," Technical Report 95-09, Department of Computer Science, University of Wollongong, 1995.
2. Secure Hash Standard, FIPS PUB 180, 1993.
3. Secure Hash Standard, FIPS PUB 180-1, 1995.
4. Secure Hash Standard, FIPS PUB 180-2, 2001.
5. Secure Hash Standard, FIPS PUB 180-2, 2002.
6. Data Encryption Standard (DES), FIPS PUB 46-3, 1999.
7. P. Pal and P. Sarkar, "PARSHA-256 – A new parallelizable hash function and a multithreaded implementation," in *Proceedings of the 10th International Workshop on Fast Software Encryption*, LNCS 2887, 2003, pp. 347-361.
8. H. Handschuh and D. Naccache, "SHACAL," *NESSIE*, 2001.

9. J. Lu, J. Kim, N. Keller, and O. Dunkelman, "Related-key rectangle attack on 42-round SHACAL-2," in *Proceedings of the 9th Information Security Conference*, LNCS 4176, 2006, pp. 85-100.
10. H. Yoshida and A. Biryukov, "Analysis of a SHA-256 variant," in *Proceedings of the 12th Annual Workshop on Selected Areas in Cryptography*, LNCS 3897, 2005, pp. 245-26.
11. D. R. Stinson, *Cryptography Theory and Practice*, 3rd ed., CRC Press, US, 2005.
12. B. Schneier, *Applied Cryptography*, 4th ed., John Wiley & Sons, Inc., US, 1996.
13. V. Rijmen and E. Oswald, "Update on SHA-1," in *Proceedings of RSA*, LNCS 3376, 2005, pp. 58-71.
14. X. Wang, H. Yu, and Y. L. Yin, "Finding collision in the full SHA-1," in *Proceedings of Crypto*, LNCS 3621, 2005, pp. 17-36.
15. H. Gilbert and H. Handschuh, "Security analysis of SHA-256 and sisters," in *Proceedings of the 6th Information Security Conference*, LNCS 3006, 2003, pp. 175-193.
16. F. Mendel, N. Pramstaller, C. Rechberger, and V. Rijmen, "Analysis of step-reduced SHA-256," in *Proceedings of the 13th Annual Fast Software Encryption Workshop*, LNCS 4047, 2006, pp. 126-143.
17. I. Nikolić and A. Biryukov, "Collisions for step-reduced SHA-256," in *Proceedings of the 15th Annual Fast Software Encryption Workshop*, LNCS 5086, 2008, pp. 1-15.
18. S. K. Sanadhya and P. Sarkar, "New collision attacks against up to 24-step SHA-2," in *Proceedings of the 9th International Conference on Cryptology in India*, LNCS 5365, 2008, pp. 91-103.
19. F. Chabaud and A. Joux, "Differential collisions in SHA-0," in *Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, 1998, pp. 56-71.
20. X. Wang, H. Yu, and Y. L. Yin, "Efficient collision search attacks on SHA-0," in *Proceedings of the 25th Annual International Cryptology Conference*, LNCS 3621, 2005, pp. 1-16.
21. Y. S. Yeh, T. Y. Huang, I. T. Chen, and S. C. Chou, "Analyze SHA-1 in message schedule," *Journal of Discrete Mathematical Sciences and Cryptography*, Vol. 10, 2007, pp. 1-7.
22. S. Adcock, "Genetic algorithm utility library," http://gaul.sourceforge.net.
23. J. H. Holland, *Adaptation in Natural and Artificial System*, The University of Michigan Press, US, 1975.
24. S. Indesteege, F. Mendel, B. Preneel, and C. Rechberger, "Collisions and other non-random properties for step-reduced SHA-256," in *Proceedings of Selected Areas in Cryptography*, LNCS 5381, Springer-Verlag, 2009, pp. 276-293.
25. K. Aoki, J. Guo, K. Matusiewicz, Y. Sasaki, and L. Wang, "Preimages for Step-Reduced SHA-2," in *Proceedings of ASIACRYPT*, 2009, pp. 578-597.
26. E. A. Grechnikov, "Collisions for 72-step and 73-step SHA-1: Improvements in the Method of Characteristics," Cryptology ePrint Archive, Report 2010/413, 2010.
27. F. Mendel, T. Nad, and M. Schläffer, "Finding SHA-2 characteristics: Searching through a minefield of contradictions," in *Proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security*, LNCS 7073, 2011, pp. 288-307.
28. A. Biryukov, M. Lamberger, F. Mendel, and I. Nikolić, "Second-order differential

collisions for reduced SHA-256," in *Proceedings of the 17th International Conference on the Theory and Application of Cryptology and Information Security*, LNCS 7073, 2011, pp. 270-287.
29. K. Matusiewicz, J. Pieprzyk, N. Pramstaller, C. Rechberger, and V. Rijmen, "Analysis of simplified variants of SHA-256," in *Proceedings of Western European Workshop on Research in Cryptology*, LNI, P-74, 2005, pp. 123-134.

Chu-Hsing Lin (林祝興) received his Ph.D. degree in computer sciences from National Tsing Hua University, Taiwan. Now he is a faculty at the Computer Science Department, Tunghai University. Dr. Lin has ever been the Director of the Computer Center from 1995 to 1999, and the Chair of the CS Department from 2004 to 2007. He has also been one of the Board Directors of the Chinese Information Security Association (CCISA) from 2001 till now. Dr. Lin has published over 150 papers in academic journals and international conferences. He was granted over twenty research projects by government departments and private companies in recent years. In 2006 and 2008, he was awarded the Outstanding Instructor Award of Master & Ph.D. Thesis, respectively, by the IICM (Institute of Information & Computing Machinery). His current research interests include multimedia information security, wireless ad hoc networks, embedded systems applications.

Chen-Yu Lee (李鎮宇) received his MS degree in computer science and information engineering from Tunghai University, Taiwan in 2000. He is currently pursuing his Ph.D. degree in Computer Science from National Chiao Tung University, Taiwan. His research interests include cryptography, information security, and DRM system.

Krishna M. Kavi is currently a Professor of Computer Science and Engineering at the University of North Texas and the Director of NSF Industry/University Cooperative Research Center on Net-Centric Software and Systems. Between 2001-2009 he served as the Chair of CSE Department at UNT. Previously he was on the faculty of the University of Texas at Arlington and the University of Alabama in Huntsville. He was a Distinguished Visiting Professor at National Chiao Tung University in 2009-2010. He is currently serving as an Associate Editor of the Journal of Information Science and Engineering. Kavi's research is primarily in

computer systems architecture including dataflow architectures, cache memories and 3D DRAMs. He also works in QoS based service composition and security vulnerability ontologies. His record includes more than 150 technical publications.

Deng-Jyi Chen (陳登吉) is now a Professor at Computer Science and Information Engineering Department of National Chiao Tung University, Hsinchu, Taiwan. Prior to joining the faculty of National Chiao Tung University, he was with National Cheng Kung University, Tainan, Taiwan. So far, he has been publishing more than 130 related papers in the area of software engineering (software reuse, object-oriented systems, visual requirement representation), multimedia application systems (visual authoring tools), e-learning and e-testing system, performance and reliability modeling and evaluation of distributed systems, computer networks. Some of his research results have been technology transferred to industrial sectors and used in product design. So far, he has been a chief project leader of more than 10 commercial products. Some of these products are widely used around the world. He has been received both research awards and teaching awards from various organizations in Taiwan and serves as a committee member in several academic and industrial organizations.

Yi-Shiung Yeh (葉義雄) did his Ph.D. in Computer Science, (September 1981 – December 1985) MS in Computer Science, (September 1978 – June 1980) from the Department of EE and CS, University of Wisconsin-Milwaukee. Since August 1988 he is working as a Professor in the Institute of CS and IE, National Chiao Tung University. From July 1986 to August 1988 he worked as an Assistant Professor in the Department of Computer and Information Science, Fardham University. From July to December 1984 he was a doctorate intern at Johson Controls, Inc. From August 1980 to October 1981 he worked as a System Programmer in System Support Division, Milwaukee Country Government. His research interests were cryptography and information security, reliability and performance, DNA computation.