Journal of Theoretical and Applied Information Technology
20th November 2013. Vol. 57 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

NEW COMPRESSION FUNCTION TO SHA-256 BASED ON THE TECHNIQUES OF DES.

[1]ZAKARIA KADDOURI, [2]FOUZIA OMARY, [3]ABDOLLAH ABOUCHOUAR, [4]MOHSSIN DAARI, [5]KHADIJA ACHKOUN.

LRI Laboratory (Ex: Networks and Data Mining Laboratory)
Department of Computer Science, Faculty of Sciences, Mohammed V University-Agdal, Rabat . Morocco.

E-mails : [1]kaddouri.zakaria@gmail.com, [2]omary@fsr.ac.ma, [3]abdollah.abouchouar@gmail.com,
[4]daari.mohssin@gmail.com, [5]khadija.achkoun@gmail.com

ABSTRACT

In this paper, we propose a new hash function based on the SHA-256 diagram and techniques of DES, that we call SHA-SBOX. This proposed hash function produces a 256-bit fingerprint exactly like the classic version. Its effectiveness in the resistance to differential and linear attacks is widely comparable to SHA-256. In light of recent attacks on MD4, MD5, SHA-0 and SHA-1, there is a great need to consider other strategies to design hash functions. We present a new compression function for SHA-256 with an internal structure based entirely on the techniques of DES.

Keywords: *SHA-256, Permutation, Substitution, Linear Attacks, Differential Attacks.*

1. INTRODUCTION

In 2002 NIST standardized a new family of hash functions, SHA-2 [13]. Four versions of this algorithm allow to calculate fingerprints 224, 256, 384 and 512 bits. Like its predecessors, SHA2 uses the algorithm extension field of Merkle-Damgard and the construction of Davies-Meyer. The size of the chaining variables is 256 bits for the first two versions and 512 bits for the last two. Message blocks corresponding to respective sizes were 512 and 1024 bits. Generally, SHA-224 and SHA-256 use 32-bit registers, then, SHA-384 and SHA-512 use 64-bit registers. The overall structure of operations carried out between these four functions is similar.

SHA-256 became the new recommended standard for cryptographic hash after the attacks on MD5 and SHA-1. Cryptanalysis of other members of the SHA family was relatively low compared to SHA-0 and SHA-1. In 2003, Helena Handschuh and Henri Gilbert published an analysis of SHA-256, 384 and 512. Their study shows that the other members of SHA are not affected by the attacks that were made and tested on other hash functions (MD4, MD5 and SHA-1 and others) [13]. Linear and differential attacks do not apply.

In contrast, the two cryptologists have highlighted significant weaknesses on modified versions [13]. By changing the constants or the initialization parameters to make them symmetric while replacing modular addition by XOR, we get a hash that produces a symmetrical fingerprint if the incoming message is symmetrical too.

The weaknesses discovered on MD-5 and SHA-1 brought to mind the fragility of SHA-2 which is built on the same pattern and the same operations. Therefore, the NIST (National Institute of Standards and Technology) has launched an international competition to select a hash function of new generation. On the 2nd of October, 2012, NIST designated Keccak as the winning algorithm, which it is not intended to replace SHA-2, but to provide an alternative.

The objective of our work is to develop a new efficient variant of SHA-256 ensures the same properties as the classic model including resistance to differential and linear attacks.

In the next section of this paper, we present a preliminary related to hash functions. The third section describes our new function by analyzing its security and its performance.

2. PRELIMINARY

2.1 The Definition and the General Model of the Hash Functions

Hash functions (more exactly cryptographic hash functions) are an important primitive in cryptography in general based on the diagram of Merkle–Damgård [1].

Hash functions that map bit-strings of arbitrary finite length into strings of fixed length. This output is commonly called a hash value, a message digest, or a fingerprint. Given h with an input x, computing h(x) must be easy. A one-way hash function must satisfy the following properties.

- preimage resistance : it is computationally infeasible to find any input which hashes to any pre-specified output. That is, given a *y* in the image of *h*, it is computationally infeasible to find an input *x* such that h(*x*) = *y*.

- second preimage resistance : it is computationally infeasible to find any second input which has the same output as any specified input. That is, given a *x* in the image of *h(x)*, it is computationally infeasible to find an input x_0 x such that $h(x) = y$.

A cryptographically useful hash function must satisfy the following additional property:

- collision resistance : it is computationally infeasible to find a collision. That is, it is computationally infeasible to find a pair of two distinct inputs *x* and x_0 such that $h(x) = h(x_0)$.

2.2 S-BOX Principle

Substitution tables take in general an m-bit variable in the input and produce an output of n bits, inputs and outputs are not necessarily the same size. The tables are often defined in advance. The presented values should be chosen to avoid attacks by various means such as the use of bent functions. In the case of DES, it has been proven that S-BOX tables [5] were designed to resist differential and linear cryptanalysis.

The S-Box of DES algorithm are defined as: S1, S2 ,... , S8 where Each of the unique selection functions, takes a 6-bit block as input and yields a 4-bit block as output.

Let Si be a function and B is a block of 6 bits, then Si(B) is determined as follows: The first and last bits of B represent in base 2 a number in the range 0 to 3. Let that number be k. The middle 4 bits of B represent in base 2 a number in the range 0 to 15. Let that number be j. relatively in the table the number in the k'th row and j'th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output Si(B) of Si for the input B. For example, for input 011011 the row is 01, that is row 1, and the column is determined by 1101, that is column 13. In row 1 column 13 appears 5 so that the output is 0101 [10].

2.3 Description of SHA-256

Published in 2002 [8], SHA-256 is one of the last members of the family dated MD-SHA. In addition to its output size, it contains several new features compared to its predecessors.

For example, the message expansion is much more complex, and corrects previous errors of SHA-0 and SHA-1. In addition, the step function updates two registers both for better dissemination.

We note these registers in Ai and Bi. As its name suggests, SHA-256 hashes product n = 256 bits.

It therefore maintains an internal state of r = 8 registers and w = 32 bits each, initialized by the chaining variable input:

$A_{-3} = h_3 \quad A_{-2} = h_2 \quad A_{-1} = h_1 \quad A_0 = h_0$

$B_{-3} = h_7 \quad B_{-2} = h_6 \quad B_{-1} = h_5 \quad B_0 = h_4$

On each call, m = 16 message words will be treated with s = 64 steps (we note t = 1 and u = 64 in our formalism).

The message expansion is much more complex than in other versions of SHA, but always uses a recurrence formula:

For $0 \leq i \leq 15$; $W_i = M_i$,

For $16 \leq i \leq 63$; $W_i = \sigma_1 (W_{i-2}) + W_{i-7} + \sigma_2 (W_{i-15}) + W_{i-16}$,

With:

$\sigma 1 (x) = (x^{>>>7}) \oplus (x^{>>>18}) \oplus (x^{>>3})$

$\sigma 2 (x) = (x^{>>>17}) \oplus (x^{>>>19}) \oplus (x^{>>10})$.

During each step i, the target register A_{i+1} and B_{i+1} are updated by the functions f and g, respectively:

$A_{i+1} = f(A_i, A_{i+1}, A_{i-2}, A_{i-3}, B_i, B_{i-1}, B_{i-2}, B_{i-3}, W_i, K_i)$
$= \Sigma_0 (A_i) + Maj(A_i, Ai-1, Ai-2) + Bi-3 + \Sigma 1 (Bi) + Ch(Bi, Bi-1, Bi-2) + Wi + Ki,$

$B_{i+1} = g(A_i, A_{i-1}, A_{i-2}, A_{i-3}, B_i, B_{i-1}, B_{i-2}, B_{i-3}, W_i, K_i)$
$= A_{i-3} + B_{i-3} + \Sigma_1(B_i) + Ch(B_i, B_{i-1}, B_{i-2}) + W_i + K_i,$

Where K_i are predefined constants for each step and Maj functions and Ch are defined Boolean functions

$Maj(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$

$Ch(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G)$

$\Sigma_0(x) = (x^{>>>2}) \oplus (x^{>>>13}) \oplus (x^{>>22})$

$\Sigma_1(x) = (x^{>>>6}) \oplus (x^{>>>11}) \oplus (x^{>>25})$.

At the end of the 64 steps, the words of the output of the compression function are calculated by:

$h'_0 = A_{64} + A_0$

$h'_1 = A_{63} + A_{-1}$

$h'_2 = A_{62} + A_{-2}$

$h'_3 = A_{61} + A_{-3}$

$h'_4 = B_{64} + B_0$

$h'_5 = B_{63} + B_{-1}$
$h'_6 = B_{62} + B_{-2}$
$h'_7 = B_{61} + B_{-3}$

3. NEW COMPRESSION FUNCTION

3.1 Description

This new function is an iterated hash function using the same SHA-256 standard diagram, where the compression function is based on the principle of substitution and permutation. Its design is transparent and based on similar operations to those used in the SHA family, the difference is in the compression functions used. Boolean functions Ch and Maj have been replaced by functions of substitution and permutation as we designate F_{PS}.

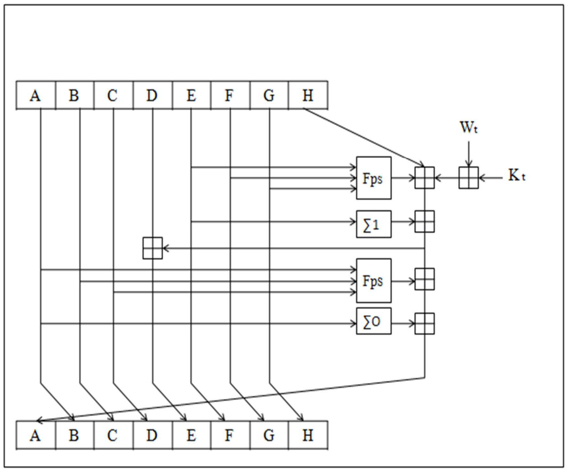

Figure.1: An SHA-SBOX operation.

Permutations used in the compression function are well defined, the S-box used are identical to those used in the DES encryption algorithm, which ensures a strong confusion.

The processing of the function Fps is defined by these steps:

- Fps takes in the input three blocks of 32 bits, concatenates and divides them into two blocks of 48 bits.
- Application of the substitution function S-Box (DES) on the 48 blocks giving two output blocks of 32 bits.
- Application of the permutation P on the two blocks.
- Application of the XOR to the two blocks of 32 bits.

Result: a single block of 32 bits.

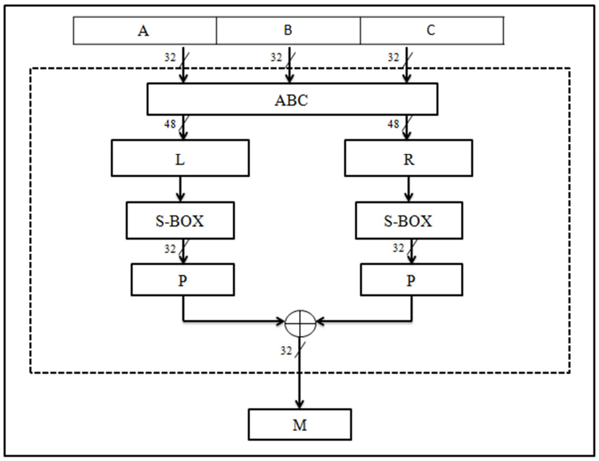

Figure.2: Pattern of our compression function Fps

3.2 Formalization:

On the basis of the diagram of SHA-256, which uses a compression function based on Boolean functions Ch and Maj, this new function is an iterated hash function in which Ch and Maj functions have been replaced by substitution and permutation.

Substitution used in this function operates the S-Box used in the DES encryption algorithm, which ensures a very strong confusion.

This function F_{PS} is defined from $\mathbb{F}_2^{96}$ to $\mathbb{F}_2^{32}$ where three blocks are concatenated, each of 32 bits, let A, B and C, these three blocks. Then, the concatenated block ABC is divided into two parts, right Ri and left Li relative to the ith iteration as:

for $0 \leq j \leq 47$; Li = ABCj
for $48 \leq j \leq 95$; Ri = ABCj

Then, each part undergoes the substitution S using the S-box and the permutation P in order to generate new blocks Li' and Ri',
where Li' = P∘S (Li) and Ri'= P∘S(Ri).
To result in the final M block with a size of 32 bits by adding the blocks Li' and Ri'.
Hence, Finally:

$$F_{PS} : \mathbb{F}_2^{96} \rightarrow \mathbb{F}_2^{32}$$
$$ABC \longrightarrow F_{PS}(ABC) = M = P \circ S\,(Li) \oplus P \circ S(Ri).$$

3.3 Security and Performance

The design of our function is transparent and based on the same operations used in SHA-256 and the same substitution tables (S-BOX) and permutations of DES. The analysis of the security of SHA-BOX can be made for the security analysis of SHA-256 (which is well studied) and DES.

- The role of S-box is to minimize the probability of "elementary" approximation, in fact, it is a nonlinear operation specifically designed to resist differential and linear cryptanalysis.

- Our function ensures a strong avalanche effect, it causes more and more important changes progressively, and the data propagate through the structure of the algorithm. Thereby disrupting a single input bit, we ideally get an output totally different.

- Fps operations are less costly in terms of execution time.
In fact, the direct access operations to tables of substitutions and permutations are faster compared to Boolean operations.

- All Wang's attacks [9] on the SHA family on the intermediate hashed differential and linear types cannot be applied (substitutions and permutations can avoid attacks) [11].

- The modular addition applied to the results from permutations and substitution (S-Box) at every round, increases the level of security and complicates the task of attack by collision.

✓ Security level (birthday attack) of SHA-SBOX is: 2^{128} bits.

3.4 Experimentation

Comparing Avalanche effect is the main indicator of the performance of the new function. The table and figure below compare the Avalanche effect of the SHA-256 and SHA-SBOX.

Table1: Avalanche effect of SHA-256 and SHA-SBOX

Size of Plaintext by character	**Avalanche**	
	SHA-256	**SHA-SBOX**
1000	183	209
10000	194	213
15000	200	202
20000	185	184
25000	244	225
30000	166	205
35000	189	191
40000	200	223
45000	197	209

The figure below shows a graph detailing the results of the avalanche effect testing for texts of different sizes.

We can notice that the avalanche effect of SHA-SBOX is often higher compared to SHA-256.

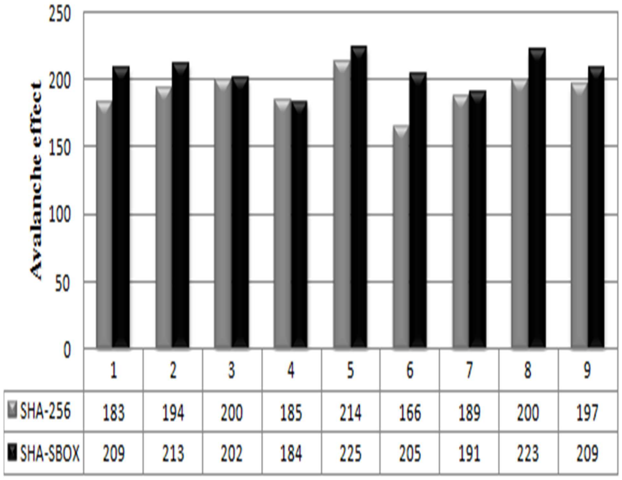

Figure.3: Graphical representation of the avalanche effect of SHA-256 and SHA-SBOX.

According to the Table.1 and the Figure.3, we can say that our contribution gives better results.

4. CONCLUSION

A hash function depends mainly on its compression function, we have presented in this work a kind of hybridization between SHA-256 and DES, in order to improve the compression function of the classic version, for this, we designed a new function whose internal operations are mainly based on permutation and substitution (S_BOX), which makes our hash function more efficient, fast and also resistant to attack.

The proposed compression function can be applied to different variants of the SHA family that generate fingerprints of different sizes.

REFRENCES:

[1] I.B. Damgard, "A design principle for hash functions," Advances in Cryptology,,Proc. Crypto'89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 416-427.

[2] R. Merkle, "One way hash functions and DES," Advances in Cryptology, Proc.,Crypto'89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428-446.

[3] Preneel, Cryptographic Hash Functions, Kluwer Academic Publishers, to ap-Pear

[4] R. Anderson, The classi_cation of hash functions," Proc. of the IMA Conference on Cryptography and Coding, Cirencester, December 1993, Oxford University Press, 1995, pp. 83-95.

[5] Kaisa Nyberg : Perfect Nonlinear S-Boxes. Dans Donald W. Davies, éditeur :EUROCRYPT'91, volume 547 de Lecture Notes in Computer Science, pages 378-386. Springer, 1991.

[6] Kaisa Nyberg : Perfect Nonlinear S-Boxes. Dans Donald W. Davies, éditeur :EUROCRYPT'91, volume 547 de Lecture Notes in Computer Science, pages 378-386. Springer, 1991.

[7] John Kelsey : How to Choose SHA-3. http://www.lorentzcenter.nl/lc/web/2008/309/presentations/Kelsey.pdf, 2008.

[8] Henri Gilbert, Helena Handschuh : *Security Analysis of SHA-256 and Sisters*. Selected Areas in Cryptography 2003: 175-193.

[9] Xiaoyun Wang, Yiqun Lisa Yin et Hongbo Yu : Finding Collisions in the Full SHA-1. Dans Victor Shoup, editeur : CRYPTO'05, volume 3621 de Lecture Notes in Computer Science, pages 17-36, 2005.

[10] Kaisa Nyberg : Perfect Nonlinear S-Boxes. Dans Donald W. Davies, editeur :

[11] EUROCRYPT'91, volume 547 de Lecture Notes in Computer Science, pages 378-386. Springer, 1991

[12] Resistance of Balanced S-boxes to Linear and Differential Cryptanalysis.

[13] FIPS 180-2: Secure Hash Standard, Août 2002. Cf. http://csrc.nist.gov. 32

[14] FIPS 180-4: Secure Hash Standard, Mars 2012. Cf. http://csrc.nist.gov. 32

[15] F.Omary, A.Tragha, A.Lbekkouri, A.Bellaachia, A.Mouloudi « A New Ciphering Method Associated with Evolutionary Algorithm». Lecture Notes in Computer Science – Publisher : Springer Berlin / Heidelberg –ISSN: 0302-9743 –Subject :Computer Science-Volume 3984/ 2006.

[16] F.Omary, A.Tragha, A.Bellaachia, A.Mouloudi. « Design and Evaluation of Two Symmetrical Evolutionist-Based Ciphering Algorithms ». International Journal of Computer Science and Network Security (IJCSNS) February 28, 2007 pp 181-190.

[17] Z.Kaddouri, F.Omary and A.Abouchouar "Binary Fusion Process to the Ciphering System “Sec Extension to Binary Blocks”, Journal of Theoretical and Applied Information Technology, ISSN: 1992-8645, Volume 48, n°1, pages 067 - 075, 10th February2013.