ISSN 2320 - 2602
Volume 7 No.6, June 2018
International Journal of Advances in Computer Science and Technology
Available Online at http://www.warse.org/IJACST/static/pdf/file/ijacst04762018.pdf
https://doi.org/10.30534/ijacst/2018/04762018

Algorithm of without key hash–function based on Sponge-scheme

M.M. Aripov[1], D.M.Kuryazov[2]
[1]Dr., Prof., of National University of Uzbekistan, Uzbekistan, mirsaidaripov@mail.ru
[2]Ph.D., competitor of the National University of Uzbekistan, Uzbekistan, kuryazovdm@mail.ru

ABSTRACT

In given article is offered a new algorithm of without key hash-function, founded on cryptographic Sponge scheme.

Key words: Hash-function, Sponge scheme, message authentication code algorithm.

1. INTRODUCTION

Nowadays, according to the results of projects such as SHA-3, NESSIE and Stribog, Sponge Scheme based Keccak [1-3], Miaguchi Prinel scheme based Whirpool [5] algorithms were found as a winners and Merkel Damgard scheme based hash algorithm has established as a new standard GOST R 34.11–2012 [2].

Analysis of modern hash algorithms is presented in dissertation [4], their designing stages are described in [7] work, key hash-function and flow encryption algorithm based on Sponge-scheme are given in [6, 8].

According to the analysis of the literature in this area, possible attacks (collision detection options) to hash function algorithms are divided into three types:

1. Attack to determine collisions. In other words, finding different texts m_1 and m_2 that have the equal hash values ($hash(m_1)=hash(m_2)$).
2. Attack to determine first sample of text. Finding the text m which satisfies hash(m)=h equation based on given h – hash value.
3. Attack to determine second sample of text. Finding m_2 text, that differ from given m_1 text, and which hash values are equal ($hash(m_1)=hash(m_2)$).

Key hash function algorithms and possible attacks to them are described in [6].

2. THE MAIN PART

In the article, considering to recognized requirements and criteria for creating new algorithms of hash function, algorithm of Sponge scheme based hash function has been proposed. Sponge scheme is presented in 1-figure.

The Scheme works on the following sequence:

- The Phase “absorbing” is executed by XOR operation with current block of the source message and the first part of condition S_1 which size is r bit. Result saves in S_1. Rest part – S_2 remains unaffected. Then new S condition ($S=S_1||S_2$) is processed in f-function. This operations are continued until the end of message M.

- In the “squeezing” phase condition S is given to *f*-function and a part of S_1 is given to output. It repeats until achieving necessary length.

Transformations of new hash algorithm are carried out on A[8x8] massive consisting of elements with L bits or vector with b bit, in other words interior state of Sponge is S=b=64*L bit. Each element is $L=2^z$, (z=0,1,2,3,4,5). When L=32 bit S=2048 and S consists of 2 parts – $S=S_1||S_2$, S_1 part r=768 bit, S_2 part c=1280. The initial state of S is equal to 0^{2048} (2048 zeros).

The process of calculating hash value by given data consists of following stages:

Stage 1. Adding filling bits.

Regardless of length, the data, for which hash is calculated, is separated on 768 bit blocks. If the length of final block is less than 768 bit, it is filled up as a showed in following formula: $M_n||80_{16}||0_{16}\ldots0_{16}$.

Stage 2. Adding the length of data.

Result of 1-nd stage is concatenated with 768 bit value, represented the length of given data. Value of block is calculated as length of the data mod 2^{768}.

Stage 3. Adding the control sum.

Result of 2-nd stage is concatenated with 768 bit control sum of given data. After the last incomplete block is filled with zero, each 768 bit block divided into 24 32-bit words. Each 32-bit word is added with corresponding 32-bit word of another 768 bit block with mod 2^{32} operation. After all 32-bit words of all 768 bit blocks are added, obtained 32-bit words are concatenated. Result is 768 bit data block M.

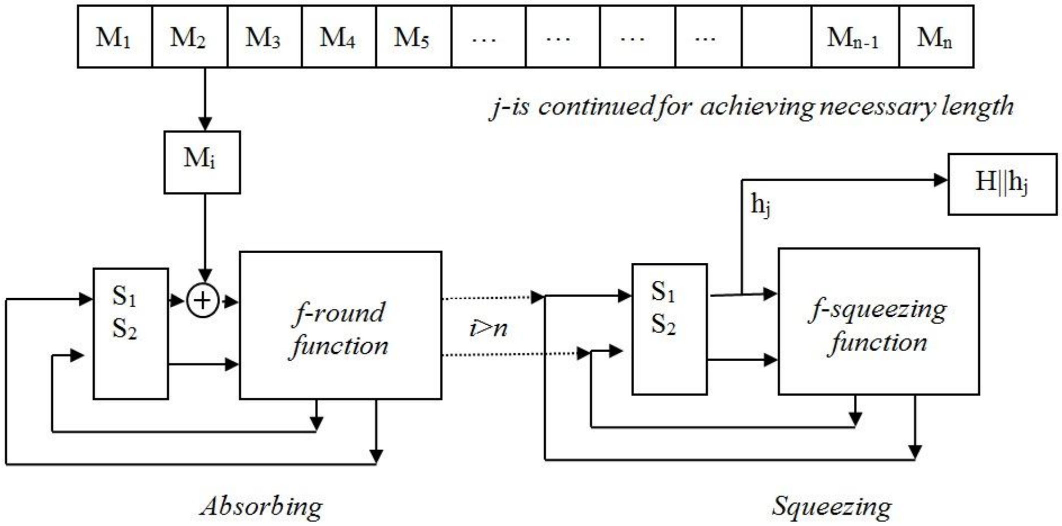

Figure-1: Sponge scheme

Stage 4. Separating information by 768 bit data block and processing.

After the completion 4 stages mentioned above, extended information is divided by 768 bit blocks M_1, M_2, ..., M_N.

Dividing information by 768 bit blocks and processing is carried out using the following function *f*:

1. M_x block is divided into 24 … 32-bit words, $M_x=m_0||m_1||\ldots||m_{23}$, using the formula (1) computes sum of m_k by XOR operation.

$$P_x = \sum_{k=0}^{23} m_k = m_0 \oplus m_1 \oplus \ldots \oplus m_{23},\ k = 0,\ldots,23 \qquad (1)$$

2. The first 24 elements of array A[8x8] is added (2) by XOR with m_k and cyclic shifted on 11 bits P_k.

$$A_{i,j}=A_{i,j}\oplus m_{(8i+j)}\oplus P_k<<<11,\ i=0,\ldots,2,\ j=0,\ldots,7 \qquad (2)$$

3. Each element of *A* array computes using formula (3).

$$A_{i,j} = A_{i,j} \oplus \begin{cases} A_{k,j},\ if\ \ k \neq i; \\ A_{i,k},\ if\ \ k \neq j; \end{cases} \qquad (3)$$

in here these are i=0…7, j=0…7, k=0…7.

4. For hashing elements uses Buff array that defined by formula (4). *Buf[0...63], buf[s]=32 bit, s=0...63.* As an index of Buff array uses element of OY array.

$$Buf[OY[8i+j]]=A_{i,j}, \qquad (4)$$

Where OY is dimensional array:
OY[0…63]={ 54, 21, 34, 9, 58, 19, 32, 7, 35, 10, 55, 20, 33, 8, 57, 18, 22, 53, 64, 59, 56, 45, 6, 31, 11, 36, 49, 46, 63, 60, 17, 44, 50, 23, 52, 61, 40, 47, 30, 5, 37, 12, 25, 48, 27, 62, 43, 16, 24, 51, 2, 39, 14, 41, 4, 29, 1, 38, 13, 26, 3, 28, 15, 42}.

The tasks of generating array elements are described in [6] work.

5. Linear transformation is presented on the (5) formula. This transformation uses AND, NOT, XOR and cyclical shift operations to process elements of Buff array and push them to A array.

$$A_{i,j}=A_{i,j}\oplus\sim Buf_{(8i+j)} \oplus((Buf_{((8i+j)+1)\bmod 64}\& Buf_{((8i+j)+2)\bmod 64})<<<11) \qquad (5)$$

there *i*=0…7, j=0…7.

6. Adding round constant is carried out using a formula (6).

$$A_{1,1}=A_{1,1}\oplus L_r \qquad (6)$$

there L_r – round constant (Table - 1)

Table 1: round constants

L_r[1]	$b1085bda_{16}$	L_r[6]	$2e45d016_{16}$	L_r[11]	$a2422a08_{16}$	L_r[16]	$f2a64507_{16}$
L_r[2]	$ebcb2f81_{16}$	L_r[7]	$714eb88d_{16}$	L_r[12]	$a460d315_{16}$	L_r[17]	$6fa3b58a_{16}$
L_r[3]	$c0657c1f_{16}$	L_r[8]	$7585c4fc_{16}$	L_r[13]	05767436_{16}	L_r[18]	$a99d2f1a_{16}$
L_r[4]	$1ecadae9_{16}$	L_r[9]	$4b7ce091_{16}$	L_r[14]	$cc744d23_{16}$	L_r[19]	$4fe39d46_{16}$
L_r[5]	$2f6a7643_{16}$	L_r[10]	92676901_{16}	L_r[15]	$dd806559_{16}$	L_r[20]	$0f70b5d7_{16}$

After completion 20 rounds returns to 7-th stage, otherwise returns to 2-th stage.

If $x<N$, i.e. if there is processed block, returns to 1-step of 4-th stage. And the next block is processed as a mentioned above, otherwise absorbing part ends up and beginning next step – squeezing.

7. Element $A_{0,0}$ of A array concatenates with $H(M)$ variable (7), there $H(0)=0$.

$$H(M)= H(M)||A_{0,0} \quad (7)$$

Elements of A array are processed by (2), (3), (4), (7) formulas. If $H(M)$ has not desired length, returns to 8-th stage, otherwise to 7-th stage.

8. As a result we have $H(M)$ – hash value.

3. CONCLUSION

The advantage of chosen Sponge scheme is that we may create key hash algorithm, flow encryption algorithm or authentification protocol without changing round function. Detailed consideration of this question is a subject of another works.

REFERENCES

1. Евсеев С.П, Король О.Г., Кунницина Е.А. **Конкурс на алгоритм хэширования SHA-3**. *// Украина. Известия Харьковского национального университета радиоэлектроники, 2010 г., №21.*
2. Национальный стандарт Российской Федерации ГОСТ Р 34.11-2012 «Информационная технология. Криптографическая защита информации. Функция хэширования». *Москва. Стандартинформ 2012г.*
3. Guido Berton, Joan Deamen, Michael Peeters, Gillan Van Assche. **Keccak sponge function family main document** // *June 19, 2010, pp. 121*.
4. Krystian Matusiewicz, M.Sc. **Analysis of Modern Dedicated Cryptographic Hash Functions**. // *Macquarie University, 2007, pp. 153.*
5. http://www.intuit.ru/studies/courses/553409/lecture/5239?page=1#keyword1.
6. Курьязов Д.М. **Алгоритм кода аутентификации на основе схемы Губка** *// Вестник Ташкентского университета информационных технологий, 2015 г., №4, стр. 127-131.*
7. Курьязов Д.М. **Этапы проектирования современных хэш алгоритмов.** *// Сборник тезисов и докладов Республиканского семинара: «Информационная безопасность в сфере связи и информатизации. Проблемы и пути их решения» Ташкент, 28 октября, 2015г., стр. 31-35.*
8. Курьязов Д.М. **Поточный алгоритм, основанный на схеме Губка.** // *Журнал Инфокоммуникации: Сети, технологии, решения. 2015г., №4. стр.11-15.*