Scientific and Practical Cyber Security Journal (SPCSJ) 4(1): 65 – 70 ISSN 2587-4667 Scientific Cyber Security Association (SCSA)

HIGH-SPEED AND SECURE HASH FUNCTION FOR BLOCKCHAIN SECURITY MECHANISMS

Anatoliy Hrytsak[1], Vasyl Kinzeryavyy[2], Dmytro Prysiazhnyi[1], Yuliia Burmak[3], Yevhen Samoylik[2]
[1]Vinnytsia National Technical University, Vinnytsia, Ukraine
[2]National Aviation University, Kyiv, Ukraine
[2]Kyiv College of Communication, Kyiv, Ukraine

ABSTRACT: Information communication technologies development and the emergence of new attack types leads to increasing the amount of existing hash functions vulnerabilities and other disadvantages. In every blockchain security mechanisms each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. New hash function development is very actual and value research task. Thus, in this paper a new hash function was proposed, which was based on well-known hash function. Improvements involved a number of changes: increased the size of words and an increase in the message digest; at the pre-processing stage, the incoming message is supplemented by a pseudo-random sequence; the numbers of nonlinear functions are increased. The proposed changes allow reducing the number of rounds in the compression function, which will guarantee at least similar security indicators with simultaneous increase in data processing speed.

KEYWORDS: Information Communication Technologies, Cybersecurity, Blockchain Security, High-Speed, Hash Function, Data Processing, Confidentiality and Integrity.

I. Introduction

Every year the level of information security is increasing in organizations of different forms ownership. First of all, this is due to an increase in the flow of information that is provided in real time with the help of Internet resources. Through the web-portals of the organization highlight results of its activities, provide online services and financial services, conduct financial transactions and etc. The lion's share of the information circulating in the above-mentioned processes needs to be adequately protected. According to this, another important task is to ensure the proper protection of information during the exchanging data at the expense of Internet resources. One of the most common methods of such protection is the using of cryptographic certificates – digital certificates that ensure the confidential exchange of data between the client and using public key encryption. However, the certificate is not only an open key with information, but also a digital signature of a server or a web portal that is implemented using hash function. But, the number of cyberattacks, especially on web portals, has increased in geometric loopholes: blocking access, stealing confidential data, monitoring traffic, etc. At the intelligence stage, hackers monitor the network to identify weaknesses, where you can get the access to users working machines and ultimately penetrate into the network. Improvement of efficiency of digital certificates, as one of the most common methods of protecting information in the process of exchange and connection, is relevant and needs improvement. Such cyberattacks as DROWN (a vulnerability that allows decrypting ciphertext without a private key) [1], FREAK (vulnerability that allows you to penetrate into the installed encrypted connection and analyze a traffic) [2], LOGJAM (vulnerability that allows reading and modification of data transmitted over a secure communications channel [3]) caused large losses to many web resource owners, including such giants as Google, Mozilla, Yahoo, etc. and put under the question reliability of digital certificates. Besides, in every blockchain security mechanisms each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. Therefore, increasing the reliability of digital certificates, as the most common methods for protecting the exchange of data through communication channels, is relevant and needs to be improved. The purpose of this work is high-speed and secure hash function development for using the information security systems and blockchain technologies.

2. Related papers

One of the most common cryptographic algorithms is hash function. They are necessary for "compressing" information into message digest that represent a bit combinations of fixed length. Hash functions of SHA-2 are very popular in applications related to the systematization, search and protection of information. Digital protocols use public key encryption to authenticate the client and server. At the confirmation stage, the hash function plays the role of the identification mark and is used to ensure the integrity of the data during transmission. That is, the hash function has to make it impossible to fake the certificate, while leaving the same signature of the verification center. Till recently, the SHA-1 hash function was used in digital certificates. In connection with the detection of numerous collisions in SHA-1 [4,5] and in the most digital certificates [5-7], Microsoft, Google and others initiated a decision to replace the hash function [8]. Starting in 2016 the SHA-2 hash function is used in the SSL certificate. However, technologies continue to improve, the power of technology is increasing, and today many works are devoted to the investigation of cryptographic strength of SHA-2, in particular, the following shortcomings were identified in works [9-10]: collisions for truncated variants SHA-512; finding the first and second prototypes; a birthday attack. The paper proposes a new method for constructing a hash function; it is the prototype of SHA-512. In our opinion, this hash function can allow to improve the efficiency of cryptographic protection of digital certificates when it is applied.

3. High-speed and secure hash function development

Pre-processing step. At the pre-processing step, an incoming message M ($M \in V_N$, $V_N \in \{0,1\}^N$, N is message length M in bits, $N \in Z$, $N < 2^{128}$) are complemented by additional sequence D_l (message length M) and pseudorandom sequence $salt$ (is determined on the basis of M), so that the resulting message length is a multiple of the length data blocks L ($L = 1024 \cdot t'$ bits, $t' \in N$):

$$M_{rez} = \left(M,\ D_l, salt \right) \tag{1}$$

where $M_{rez} \in V_{NN}$, $V_{NN} = N + 128 + N_{salt}$, $D_l = H_{Dl}(M)$, $D_l \in V_{128}$, $salt = H_{Gen}(M)$, $salt \in V_{Nsalt}$, $N_{salt} = 2L - \left((N + 128) \bmod L \right)$, as a function H_{Gen} could be any function of generating a pseudorandom sequence which is based on M, H_{Dl} is function of length M. Based on the completed message M_{rez} will determine the hash value of the message M.

Message M_{rez}, $M_{rez} \in V_{NN}$, broken into k L - bit blocks: $M_{rez} = (m_1, m_2 .... m_k)$ where $m_i \in V_L$, $i = \overline{1,k}$, $k = (NN)/L$.

Determination step of a hash. The digest of the message is iteratively calculated, processing each one m_i block messages M_{rez}, $m_i \in V_L$, $i = \overline{1,k}$ compression function F_g (2), to get the resulting hash (3):

$$h_i = F_g\left(h_{i-1}, m_i\right),\ i = \overline{1,k} \tag{2}$$

$$H\left(IV, M_{rez}\right) = h_k \tag{3}$$

where $h_0 = IV$, IV is initialization vector, $IV \in V_{L/2}$, h_i is intermediate values of the messages digest $h_i \in V_{L/2}, i = \overline{1,k}$; H is resulting hash, $H \in V_{L/2}$; F_g is the compression function uses in the hash function.

The compression function F_g is performed in three stages: splitting blocks into words (1), initialization of variables (2), compression (3).

<u>*Step 1*</u>. Each m_i data block M_{rez}, $m_i \in V_L$, $i = \overline{1,k}$, decomposes into 16 words (4):

$$m_i = \left(W_0^i, ..., W_{15}^i \right) \tag{4}$$

where $W_j^i \in V_{L/16}$, $j = \overline{0,15}$.

On the basis of words which are received W_j^i, $j=\overline{0,15}$, words are calculated $W_u^i(5)$ $W_u^i \in V_{L/16}$, $u=\overline{16,63}$:

$$W_u = W_{u-16} + Delta0(W_{u-15}) + W_{u-7} + Delta1(W_{u-2}), \quad (5)$$

where $Delta0(W_u) = Rotr(W_u,1) \oplus Rotr(W_u,8) \oplus SHR(W_u,7)$,
$Delta1(W_u) = Rotr(W_u,19) \oplus Rotr(W_u,61) \oplus SHR(W_u,6)$, $Rotr(x,l)$ is right bitwise cyclic shift of argument x for $l-$bits; $SHR(x,l)$ is left shift argument x for $l-$bits.

Step 2. Re-initialization of internal state vectors is performed T (6), $T=(T_1,...,T_8)$, $T_z \in V_{L/16}$, $z=\overline{1,8}$:

$$T_z = h_{i-1}^z \quad (6)$$

where $h_{i-1}=(h_{i-1}^1,...,h_{i-1}^8)$, h_{i-1} is the previous value of the digest, which is fed to the input of the function F_g, $h_{i-1}^z \in V_{L/16}$, $z=\overline{1,8}$.

Step 3. At this step, there is a direct compression of the data block $m_i \in V_L$, $i=\overline{1,k}$, $k = NN/L$, the value of the vectors of the internal state will change each 64 rounds $T=(T_1,...,T_8)$, $T_z \in V_{L/16}$, $z=\overline{1,8}$, through their mixing with vectors W_j and constants K_j $j=\overline{0,63}$.

For each j round, the mathematical actions given in the formulas will be executed (7) - (11), $j=\overline{0,63}$:

$$F_{g_1} = T_8 \oplus Sigma1(T_5) \oplus Ch(T_5,T_6,T_7) + W_j + K_j \quad (7)$$

$$F_{g_2} = Sigma0(T_1) \oplus Maj(T_1,T_2,T_3) \quad (8)$$

$$F_{g3} = JQ(T_3,T_6) \oplus Maj(T_2,T_3) \quad (9)$$

$$F_{g_4} = SH(T_8,T_7) \oplus Sigma(T_8) \quad (10)$$

$$T_8 = T_7 + F_{g_4};\ T_7 = T_6;\ T_6 = T_5;\ T_5 = T_4 + F_{g_1};\ T_4 = T_3;\ T_3 = T_2 + F_{g_3};\ T_2 = T_1;\ T_1 = F_{g_1} + F_{g_2} \quad (11)$$

where T_z are vectors of the internal state, $T_z \in V_{L/16}$, $z=\overline{1,8}$; W_j are words which are broken from m_i block; K_j are predetermined constants (if necessary, may change), $K_j \in V_{L/16}$;
$Ch(x,y,z)$, $Maj(x,y,z)$, $Sigma0(x)$, $Sigma1(x)$, $Delta0(x)$, $Delta1(x)$, $JQ(x,y)$ and $SH(x,y)$ are nonlinear functions that are described in (12) - (19):

$$Sigma0(x) = Rotr(x,28) \oplus Rotr(x,34) \oplus Rotr(x,39) \quad (12)$$

$$Sigma1(x) = Rotr(x,14) \oplus Rotr(x,18) \oplus Rotr(x,41) \quad (13)$$

$$Ch(x,y,z) = (x+y) \oplus (\bar{x}+z) \quad (14)$$

$$Maj(x,y,z) = (x+y) \oplus (x+z) \oplus (y+z) \quad (15)$$

$$Delta0(x) = Rotr(x,1) \oplus Rotr(x,8) \oplus SHR(x,7) \quad (16)$$

$$Delta1(x) = Rotr(x,19) \oplus Rotr(x,61) \oplus SHR(x,6) \quad (17)$$

$$JQ(x,y) = (\bar{x}+y) \oplus Rotr(x,13) \oplus SHR(\bar{y},17) \quad (18)$$

$$SH(x,y) = SHR(x,7) \oplus Rotr(y,8) \oplus Rotr(\bar{x},y) \quad (19)$$

where F_g is intermediate compression function value, $F_{g_o} \in V_{L/16}$, $o = \overline{1,4}$, $Ch(x,y,z)$, $Maj(x,y,z)$, $Sigma0(x)$, $Sigma1(x)$, $Delta0(x)$, $Delta1(x)$, nonlinear functions that were used in the original SHA-2. $JQ(x,y)$ and $SH(x,y)$ are new nonlinear functions that were proposed in this hash function.

After completing the last round, the values of the vectors of the internal state $T = (T_1,...,T_8)$, $T_z \in V_{L/16}$, $z = \overline{1,8}$, completely changed as follows:

$$T_z = T_z \oplus h_{i-1}^z, \tag{20}$$

where h_{i-1} is the previous value of the digest, which is fed to the input of the function F_g $h_{i-1} = \left(h_{i-1}^1,...,h_{i-1}^8\right)$, $h_{i-1}^z \in V_{L/16}$, $z = \overline{1,8}$. The output of the function will be given to the final values of the internal state vectors.

In our opinion, the method for constructing a hash function is developed by adding a pseudorandom sequence $salt$ to an incoming message at the pre-processing and non-linear operations $JQ(x,y)$ and $SH(x,y)$ at the step of determining the hash values. It is possible to reduce the total number of rounds in the compression function with similar or better performance and security indicators data in the aspect of resistance to various attacks and neutralization of known vulnerabilities compared with the SHA-2. Theoretical and experimental researches will be conducted for verification of this statement and cryptanalysis performing in the further works.

4. Experiments and discussion

For the experimental study on the basis of the proposed method, three hash functions with such parameters were constructed: $t' = 1$, $L = 1024 \cdot t' = 1024$, $H \in V_{L/2} = V_{512}$ for BK_1; $t' = 2$, $L = 1024 \cdot t' = 2048$, $H \in V_{L/2} = V_{1024}$ for BK_2; $t' = 3$, $L = 1024 \cdot t' = 3072$, $H \in V_{L/2} = V_{1536}$ for BK_3. As a function H_{Gen} for BK_i, $i = \overline{1,3}$ cryptographic algorithm SNOW 2.0 was selected. The software implementation of proposed hash functions was carried out as console tool using the programming language C++. Development environment was Microsoft Visual Studio 2013 (Release Version).

Therefore, to study the statistical characteristics of hash functions, these were investigated in the statistical test NIST STS [11]. Also proposed hash functions were compared with the results of the benchmark generator of pseudo-random sequences BBS and some block symmetric ciphers (Kalyna, Luna, Neptun), which worked in counter mode. Note that for this research, based on the developed hash functions and the SHA-512 function, stream ciphers were constructed to generate required length files for NIST STS statistical tests.

In Fig. 1 the statistical portrait of the passage of statistical tests is given for BK_1 (similar portraits can be built for BK_2 and BK_3), and in Table 1 the results of the study was showed. It showed that the proposed functions of healing passed a comprehensive control in accordance to NIST STS.

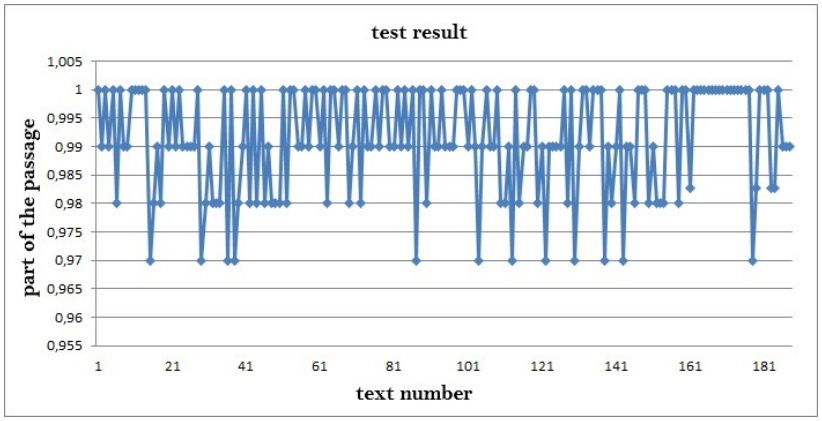

Fig. 1. The statistical portrait of the stream cipher on the basis of BK_1

Table 1. Sequence testing results

Generator	The number of tests in which the testing was completed	
	99%	96% sequence
BBS	133	188 (100%)
Kalyna	136	188 (100%)
Luna	141	188 (100%)
Neptun	140	188 (100%)
SHA-512	137	188 (100%)
BK_1	141	188 (100%)
BK_2	140	188 (100%)
BK_3	142	188 (100%)

Also, the study of the rate characteristics for developed hash functions BK_i, $i=\overline{1,3}$ was carried out. To do this randomly selected several files of different sizes and for each file was scanned hash code, while measuring the time hash code, see Table 2.

All experiments were performed using the system with following characteristics: Intel (R) Core (TM) i3-6100 processor, 3.7 GHz processor, and a 4 GB RAM based on the 64-bit Windows 7 Service Pack 1.

Table. 2 Results of the study for the speed characteristics of the hash functions

Hash function	File 1, 1 MB		File 2, 10 MB		File 2, 100 MB	
	t, s	v, MB /s	t, s	v, MB /s	t, s	v, MB /s
SHA-512	0.015	68,26	0,145	70,62	1,38	74,20
BK_1	0,012	85,33	0,101	101,38	0,926	110,58
BK_2	0,011	93,09	0,098	104,48	0,902	113, 52
BK_3	0,011	93,09	0,094	108,93	0,879	116,49

According to the obtained results of the developed hash functions' speed characteristics BK_i, $i=\overline{1,3}$, are better than well-known and widely used SHA-512 hash function.

5. Conclusions

The paper proposes a new method for secure hash function constructing, which can be used to improve the effectiveness of cryptographic protection of digital certificates in the future. It will provide a more reliable exchange of confidential information on the network. The method requires further research to test the performance on different platforms, the resistance to common methods of cryptanalysis. In the following works, it is planned to conduct the above-mentioned studies and compare them with the parameters of the hash functions of the SHA-2 series.

REFERENCES

1. N. Aviram, S. Schinzel, J. Somorovsky, "DROWN: Breaking TLS using SSLv2, Proceedings of the 25th USENIX Security Symposium", pp.18, 2016. [Online]. Available: https://drownattack.com/drown-attack-paper.pdf

2. M. Green, "Attack of the week: FREAK (or 'factoring the NSA for fun and profit')" [Online]. Available: https://blog.cryptographyengineering.com/2015/03/03/attack-of-week-freak-or-factoring-nsa/ | Date accesses: april 2018|.

3. B. Duncan, "Weak Diffie-Hellman and the Logjam Attack", [Online]. Available: (https://weakdh.org/ | Date accesses: april 2018|.

4. P. Karpman, T. Peyrin, M. Stevens, “Practical Free-Start Collision Attacks on 76-step SHA-1”, [Online]. Available: https://eprint.iacr.org/2015/530

5. S. Sanadhya, P. Sarkar, “22-Step Collisions for SHA-2” [Online]. Available: http://arxiv.org/abs/0803.1220

6. F. Kohlar, S. Schage, “On the Security of TLS-DH and TLS-RSA in the Standard Model1”, pp.50, 2013 [Online]. Available: http://eprint.iacr.org/2013/367.pdf

7. C. Meyer, J. Schwenk, “Chair for Network and Data Security Ruhr-University Bochum. Lessons Learned From Previous SSL/TLS Attacks A Brief Chronology Of Attacks And Weaknesses”, pp.15 [Online]. Available: http://eprint.iacr.org/2013/049.pdf

8. C. Castelluccia, E. Mykletun, “Improving Secure Server Performance by Re-balancing SSL/TLS Handshakes”. pp.11 (Published in “Proceeding ASIACCS '06 Proceedings of the 2006 ACM Symposium on Information, computer and communications security. pp 26-34”).

9. F. Mendel “Improving Local Collisions: New Attacks on Reduced SHA-256”, p.17 [Online]. Available: https://eprint.iacr.org/2015/350.pdf

10. C.Dobraunig, M. Eichlseder, “Analysis of SHA-512/224 and SHA-512/256”, p.30 [Online]. Available: https://eprint.iacr.org/2016/374.pdf

11. NIST Special Publication 800-22 “A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications” [Online]. Available:https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf