INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2022, VOL. 68, NO. 1, PP. 57-62
Manuscript received January 21, 2022; revised January, 2022. DOI: 10.24425/ijet.2022.139848

Analysis of the Possibility of Using Selected Hash Functions Submitted for the SHA-3 Competition in the *SDEx* Encryption Method

Artur Hłobaż

Abstract—**The paper presents analysis of the possibility of using selected hash functions submitted for the SHA-3 competition in the *SDEx* encryption method. The group of these functions will include the finalists of the SHA-3 competition, i.e. BLAKE, Grøstl, JH, Keccak, Skein. The aim of the analysis is to develop more secure and faster cryptographic algorithm compared to the current version of the *SDEx* method with SHA-512 and the AES algorithm. When considering the speed of algorithms, mainly the software implementation will be taken into account, as it is the most commonly used.**

Keywords—**secure communication; data encryption; data security; secure transmission; secure data exchange method; *SDEx* method; end-to-end data security**

I. Introduction

THE result of the author's scientific work to date was the development of an encryption method competitive to the current AES cryptographic standard, both in terms of security and speed. The developed method named *SDEx* (Enhanced Secure Data Exchange Method) was published and presented at conferences belonging to the leading conferences in the world related to networks and computer communications [1-5]. It is a block cryptographic algorithm, the security of which is based on the security of the used hash function and the Davies-Meyer scheme. The hash function used in the method acts as a dynamic generator of a sequence of pseudorandom bits. The *SDEx* method was developed using the hash functions of the SHA-2 family, i.e. SHA-256 and SHA-512, which are still considered the standard nowadays - whether for verification of message integrity or in a digital signature. Both of these functions are characterized in that the length of the input function block is twice as long as the hash that the applied function generates (more detailed description of the method can be found in Section 2).

In 2009, NIST (National Institute of Standards and Technology) announced a competition for a new SHA-3 hash function standard that would replace SHA-2. It was feared that SHA-2 would be broken or weakened, as happened with MD5 and SHA-1. Five hash functions qualified for the third final round of the SHA-3 competition: BLAKE, Grøstl, JH, Keccak, Skein [6]. As a result, the Keccak algorithm was the winner of the competition. In fact, choosing a winner was not easy as none of the finalists was the best for all applications, hardware or software, and offered no really convincing improvements over the SHA-2 family of algorithms. The currently used SHA-2, however, turned out to be more secure than assumed. Therefore, the replacement of SHA-2 with the new standard was postponed and continues to this day.

The choice of the Keccak algorithm as the winner of the competition for SHA-3 was justified, among others, by the fact that it has a large security margin, uses a new design approach and has excellent performance in hardware implementations (e.g. in FPGA). In the case of a software implementation, however, it is worse than SHA-2. Due to the fact that the speed of algorithms (of course, apart from the security aspect) is also a key element in the selection of algorithms, the article will analyse the possibilities of using all finalists of the SHA-3 competition in the *SDEx* method. The criteria that will be considered in the analysis are:

- structure and design,
- security (mainly evaluations relating to attack resistance),
- performance (computational efficiency, which refers to the speed of the algorithm).

The results of this analysis are presented in the paper.

The article is organized into five sections. In Section 2 the enhanced *SDEx* method is given. In Section 3 the SHA-3 finalists are presented. The analysis of the possibility of use of SHA-3 finalists in the *SDEx* method is presented in Section 4. The analysis takes into account elements such as structure and design, security analysis and performance comparison. Conclusions and future work are drawn in Section 5.

II. Enhanced Secure Data Exchange (*SDEx*) Method

The *SDEx* (Enhanced Secure Data Exchange) method was generally presented in [2] and its cryptanalysis was presented in [3]. The main idea of the developed method is the possibility of creating a fast and secure mechanism for end-to-end communication. Its security is based on security of hash function used and Davies-Meyer schema which makes hash function collision resistant [7]. All hash functions are iteratively constructed and divide the input data into a sequence of fixed size blocks M_1, M_2, ..., M_i. Message blocks are sequentially processed using a hash function to the intermediate state of constant size [8]. The hash function used

This work was supported by the grant from the Polish National Science Centre (no. 2020/04/X/ST6/00407).

Artur Hłobaż is with the Faculty of Physics and Applied Informatics, University of Lodz, Poland (e-mail: artur.hlobaz@uni.lodz.pl).

in *SDEx* encryption method (Fig. 1) acts as a dynamic pseudorandom string of bits generator.

The common symbols used on Figures 1-2 and equations (1-11):

- M_1, M_2, ... M_i - plaintext blocks,
- C_1, C_2, ... C_i - ciphertext blocks,
- IV - initialization vector (session key),
- h_0 - initialization hash – predetermined value for the hash function,
- h_1, h_2, . . . h_k - particular iterations of hash computation,
- H_{IV} - hash from the initialization vector,
- H_U - hash from user password,
- $\oplus$ - XOR operation,
- $+\!\!+$ - concatenation of two strings.

The equations that describe individual encryption steps take the following form:

$$C_1 = M_1 \oplus H_{IV} \oplus H_{IV+\!\!+h0} \qquad (1)$$

$$C_2 = M_2 \oplus H_U \oplus H_{IV} \qquad (2)$$

$$C_{2k+1} = M_{2k+1} \oplus h_k \oplus h_{k-1} \qquad k \geq 1 \qquad (3)$$

$$C_{2k+2} = M_{2k} \oplus H_U \oplus h_k \qquad k \geq 1 \qquad (4)$$

$$h_1 = hash\,(H_{IV+\!\!+h0}; M_1 +\!\!+ M_2) \qquad (5)$$

$$h_2 = hash\,((h_1 \oplus H_{IV+\!\!+h0}); M_3 +\!\!+ M_4) \qquad (6)$$

$$h_k = hash\,((h_{k-1} \oplus h_{k-2}); M_{2k-1} +\!\!+ M_{2k}) \qquad k \geq 3 \qquad (7)$$

The equations that describe individual decryption steps take the following form:

$$M_1 = C_1 \oplus H_{IV} \oplus H_{IV+\!\!+h0}, \qquad (8)$$

$$M_2 = C_2 \oplus (H_U \oplus H_{IV}) \qquad (9)$$

$$M_{2k+1} = C_{2k+1} \oplus h_k \oplus h_{k-1} \qquad k \geq 1 \qquad (10)$$

$$M_{2k+2} = C_{2k} \oplus H_U \oplus h_k \qquad k \geq 1 \qquad (11)$$

The element $H_{IV+\!\!+h_0}$ in equations (1, 8) is just a hash from the string created as a concatenation of IV and h_0. The enhanced encryption/decryption methods are presented in Figures 1-2.

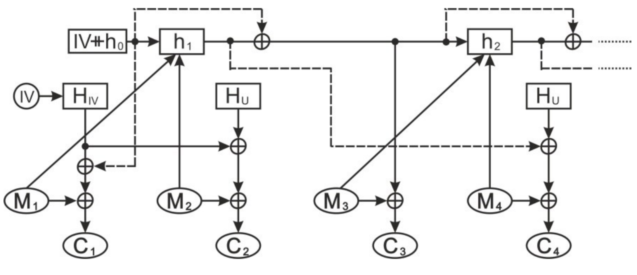

Fig. 1. Enhanced *SDEx* encryption method

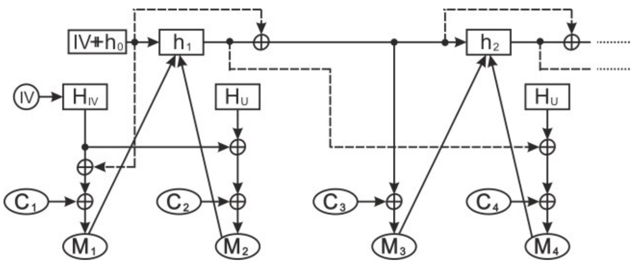

Fig. 2. Enhanced *SDEx* decryption method

In cryptography, the parameters that characterize the quality of a given encryption method are its speed and the level of security it can guarantee. Because the *SDEx* method is based on hash functions, the encryption speed is also associated with them [8]. The table below (Tab. I) contains an example speed comparison of SHA-256 and SHA-512 hash functions and AES algorithm for two different processors – Intel Core 2 1.83 GHz and AMD Opteron 2.2 GHz [9]. We can assume that the speed of *SDEx* encryption method is practically the same as the speed of the hash function used in it (omitting several XOR operations per cycle and the initial calculation of H_{IV} and H_U values). Taking into account the similar level of security and referring to the table below (Tab. I), it can be stated that the *SDEx* method using the SHA-512 hash function will be slightly faster than the AES algorithm with the 256-bit key.

TABLE I
SPEED COMPARISON OF ALGORITHMS

Intel Core 2 1.83 GHz		**AMD Opteron 8354 2.2 GHz**	
Algorithm	**Speed (MB/s)**	**Algorithm**	**Speed (MB/s)**
SHA-256	116,38	SHA-256	145,75
SHA-512	103,81	SHA-512	161,48
AES (128-bit key)	145,75	AES (128-bit key)	207,62
AES (256-bit key)	100,66	AES (256-bit key)	146,80

The most general estimation of hash function security level is half of its hash length. When the SHA-512 function is used, the *SDEx* method's security level can be estimated at the level of 256 bits. Taking this into account, it can be said that the *SDEx* method based on SHA-512 hash function provides a similar or higher level of security then AES with 256-bit key, which nowadays is encryption standard.

III. THE SHA-3 FINALISTS

A. BLAKE

BLAKE [6, 10], like SHA-2, is a family of four hash functions in two variants. First one, used for computing hashes up to 256 bits long, uses 32-bit words (BLAKE-224 and BLAKE-256) and second, used for computing hashes up to 512 bits long, which uses 64-bit words (BLAKE-384 and BLAKE-512). Its structure is based on HAIFA construction and ChaCha stream cipher. BLAKE compression function is based on three basic bit operations (+, <<<, $\oplus$) called ARX construct (from the words Addition, Rotation, XOR). Its internal state is initialized using initial value, salt and a counter (number of bits hashed so far) as shown in Figure 3. The input data block for a single iteration of the compression function is twice as large as the partial hash generated by the function.

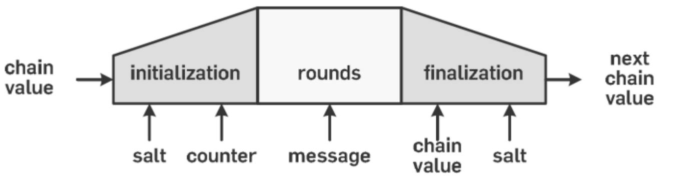

Fig. 3. BLAKE compression function

B. *Grøstl*

Grøstl [6, 11] is a family of hash functions, which can return message digests from 8 to 512 bits. The most popular variants are these returning hashes equal to 256 and 512 called Grøstl-256 and Grøstl-512, respectively. Grøstl divides the padded input message M into l-bit message blocks m_1, m_2 ,…, m_t and iteratively computes $h_i = f(h_{i-1}, m_i)$, where h_{i-1} is l-bit chaining input with an initial value of $h_o = IV$ (see Figure 4). The compression function f is based on a pair of l-bit permutation functions P and Q which are heavily based on the Rijndael (AES) block cipher. The result of each iteration of the compression function is a partial hash that is at least twice the size of the final size of the hash. In the last step of hash computing the output transformation Ω, after the processing of the last message block, truncates the output to the desired size n (usually 256 or 512 bits).

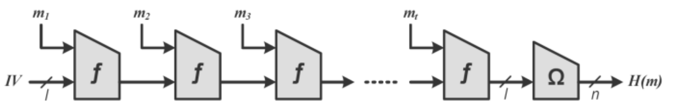

Fig. 4. Grøstl compression function

C. *JH*

JH family [6, 12] includes four hash functions: JH-224, JH-256, JH-384 and JH-512. The F_8 compression function structure is based on large block cipher, in which AES design methodology is applied, with constant key. It sequentially processes the split message blocks m_1, m_2 ,…, m_t, starting with an initial vector $IV = h_o$ (see Figure 5). Each iteration of the compression function takes on input the m-bit message block m_i and the $2m$-bit hash value $h_{(i-1)}$ and generates the $2m$-bit $h_{(i)}$. Therefore, the partial hash generated by the compression function is twice as large as the message block on its input. Final message digests are obtained by truncating the final output to the desired hash sizes.

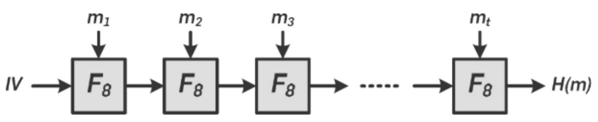

Fig. 5. JH compression function

D. *Keccak*

Keccak [6, 13] is a family of hash functions that can have many variants. It is based on sponge construction (see Figure 6). The main function of Keccak is permutation f. Permutation size and state size in the sponge construction can take values such as 25, 50, 100, 200, 400, 800, 1600. In the competition for SHA-3, the Keccak algorithm was submitted with the largest permutation size (1600 bits). The message block size is referred to as r (for rate). The difference between the permutation size and the message block size r is referred to as c (for capacity). Therefore, the message block size m_i varies according to the output size: Keccak-512 has a block size of 576 bits, Keccak-384 has 832 bits, Keccak-256 has 1088 bits, and Keccak-224 has 1152 bits. In conclusion, the bigger output of the compression function the smaller the message block that is processed by each compression-function call. As a result the bigger the hash of the message, the slower the Keccak algorithm runs.

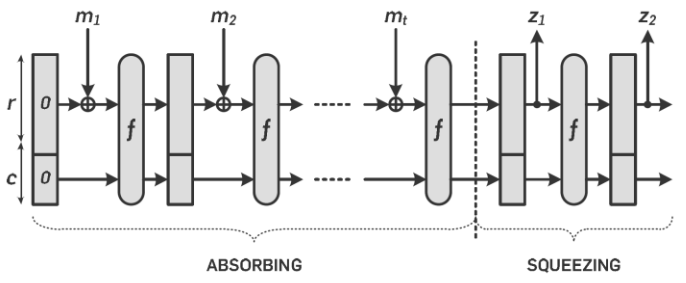

Fig. 6. Sponge construction of Keccak

E. *Skein*

Skein [6, 14] is an iterative hash function that is built on a tweakable block cipher Threefish, which is defined for 256, 512 and 1024-bit block sizes. Threefish uses only three mathematical operations: Addition, Rotation and XOR (ARX construct). The key is the same size as the block, and the tweak value is 128 bits for all block sizes. Its compression function is based on Threefish and a modified Matyas-Meyer-Oseas. This structure is often referred to as Unique Block Iteration (UBI).

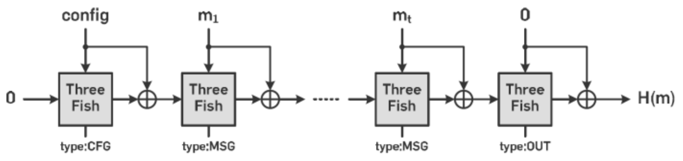

Fig. 7. Skein normal hashing scheme

IV. ANALYSIS OF THE POSSIBILITY OF USE OF SHA-3 FINALISTS IN THE *SDEx* METHOD

A. *Structure and design*

To be able to use the selected hash function in the *SDEx* method, certain dependencies must be met. The most important and the main one is the dependence related to the size of the message data block, on which the hash function operates, and the hash generated by it, i.e. the input message block size should be twice as large as the hash size generated by this function. Based on the analysis of the finalists on SHA-3, this criterion is fulfilled by three hash functions, i.e. Blake, Grøstl and Skein (see Tab. II).

The second important dependence is that for individual iterations of the hash function, the generated partial hash, called the internal state of the hash function or "internal hash sum" after each compression, should also be half shorter than the size of the input message block. In this case, out of the three above-mentioned functions, the condition is fulfilled only by BLAKE algorithm (Tab. II). For the Grøstl and Skein, each iteration of the hash function produces a partial hash of the same length as the input message block. The final hash it generates is half the length of the partial hash due to cutting off half its length in the last iteration of the hash count. Therefore, the only finalist of the SHA-3 competition that could be used in the *SDEx* method without modifying it is BLAKE.

Due to the fact that the BLAKE algorithm has its successors, i.e. BLAKE2b (successor of BLAKE-512), BLAKE2s (successor of BLAKE-256) [15] and the latest version of the BLAKE3 algorithm from 2020 [16], they will also be considered in further analysis. They also meet the main conditions for using them in the *SDEx* method and, what is important, they have much better properties than the basic version of BLAKE.

TABLE II
ANALYSIS OF THE STRUCTURE AND DESIGN OF FINALIST SHA-3 CANDIDATES

Hash Algorithm	Hash size (bits)	Message Block size (bits)	Internal state size of hash function (bits)	Rounds of Compression
BLAKE-256	256	512	256	14
BLAKE-512	512	1024	512	16
Grøstl-256	256	512	512	9
Grøstl-512	512	1024	1024	10
JH-256	256	128	256	42
JH-512	512	256	512	42
Keccak-224	224	1152	1600	24
Keccak-256	256	1088	1600	24
Keccak-384	384	832	1600	24
Keccak-512	512	576	1600	24
Skein-512	256	512	512	72
Skein-1024	512	1024	1024	72

B. *Security analysis*

Taking into account the security analysis of the five candidates of the SHA-3 competition [6], it can be concluded that after several years of intensive analysis, no attack came close to threatening the basic security properties of any of them. Additionally, neither of them was identified as a clear winner or loser in this category. Therefore, it is currently considered that the minimum security level of each of the SHA-3 and SHA-2 finalists is estimated at half the length of the hash that the function generates.

Instead, there are differences in the size of their security margins [6,17], where the security margin is defined as the fraction of the hash or compression function that has not been successfully attacked. For example, an attack on six rounds of a ten-round hash function would give a 40% security margin. Based on the table below (Tab. III), it can be concluded that Keccak has the greatest security margin with 79% of the hash function still uninterrupted. BLAKE was not much worse (71%), while JH turned out to be the weakest (38%).

The depth of the hash function analysis performed [17], which is a rather subjective measure, may also take an important role (see Tab. III). It takes into account not only the number of published works on the cryptanalysis of individual hash functions, but also the depth and maturity of the analysis and the tools used for the cryptanalysis of the algorithm. On this basis, it can be concluded that the hash function most thoroughly examined during the SHA-3 competition was Grøstl, followed by BLAKE and Skein, and only then Keccak - the winner of the competition.

TABLE III
SECURITY MARGINS FOR THE FIVE FINALISTS BASED ON COLLISION-TYPE ATTACKS

Algorithm	Best Attack on Algorithm	Security Margin	Depth of Analysis
BLAKE	Semi-free-start near collision	71%	High
Grøstl	Semi-free-start collision	40%	Very high
JH	Semi-free-start near collision	38%	Low
Keccak	Near collision	79%	Medium
Skein	Semi-free-start near collision	56%	High
SHA-2	Collision	62%	Medium

C. *Performance comparison*

The speed of algorithms can be considered in two categories - depending on whether they are implemented in software or in hardware. Certain algorithms may turn out to be faster in the development implementation but have lower performance in hardware, or vice versa. In the article, the author will focus mainly on the first category - software implementation of algorithms - because it is the most widely researched and most often used in practice, and it is also related to his scientific work to date.

In the SHA-3 competition, the performance in the case of software implementation was measured for various classes of computers: various architectures and processors - both for 32 and 64 bit available on the market, such as AMD64, X86, ARM-NEON, 32 bit RISC or embedded microcontrollers. The vast majority of the available data on finalists' speed scores on SHA-3 was provided by two projects: eBASH (ECRYPT Benchmarking of All Submitted Hashes) [18] and XBX (eXternal Benchmarking eXtension) [19]. Tests were carried out for both shorter and longer messages. Overall, the highest performance achieved BLAKE and Skein algorithms. What is interesting, the Keccak algorithm was weaker. It was included in the group of algorithms of average performance.

In 2013, a new version of the BLAKE algorithm was presented, called BLAKE2 [15], which comes in two varieties:

- BLAKE2b, which is optimized for 64 bit platforms and can generate hashes from 1 to 64 bytes,
- BLAKE2s, which is optimized for 8 to 32 bit platforms and can generate hashes from 1 to 32 bytes.

BLAKE2b does 12 rounds, and BLAKE2s does 10 rounds, against 16 and 14 respectively for BLAKE. By reducing the number of rounds, while maintaining practically the same level of security, these algorithms were approximately 25% and 29% faster, respectively, than the original version of BLAKE.

A comparison of the speed of the most popular hash functions from the documentation of the BLAKE2 algorithm can be found in the Figure 8.

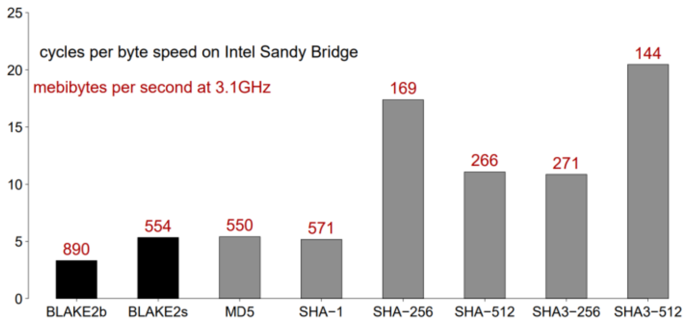

Fig. 8. Speed comparison of various popular hash functions [15]

In 2015, NIST published a new standard based on the Keccak algorithm [20]. It turned out to be the fastest of all SHA-3 finalists at that time, even from BLAKE2. The situation changed again in 2020 with the appearance of BLAKE3 - the evolution of BLAKE2 [16].

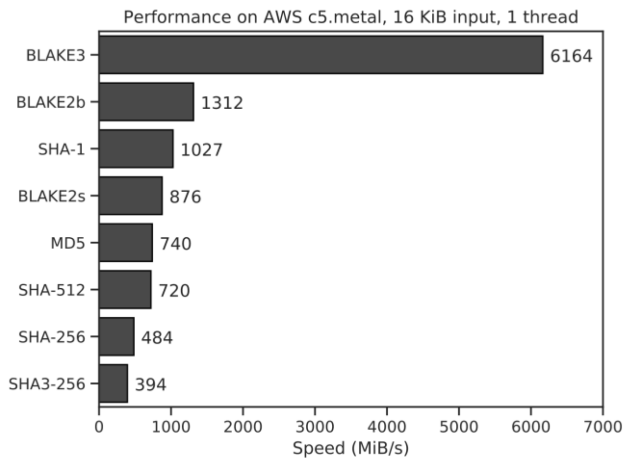

Fig. 9. Speed comparison of BLAKE3 to other popular hash functions [16]

BLAKE3 is a single algorithm with no variants. The compression function works on 32-bit words, the input message block size is 512 bits, and the generated hash and internal state size of hash function (input chaining value) is 256 bits. Based on the length of the generated hash, it can be assumed that the minimum security level it guarantees is 128 bits. Referring to the information by the authors of the BLAKE3 algorithm on its speed, it is about 4 times faster than BLAKE2b, 8 times than SHA-512 and 12 times than SHA-256 - a study conducted on Intel Cascade Lake-SP (see Figure 9).

CONCLUSIONS

The aim of the research was to conduct a literature analysis and a preliminary cryptographic analysis of the possibility of using selected (leading) hash functions submitted for the competition for the SHA-3 standard in the *SDEx* method in order to develop a secure and very fast encryption algorithm. Such an algorithm would be more competitive to the current AES encryption standard both in security and speed - mainly in the software implementation, which is the most widely used.

The conducted analysis takes into account such elements of the hash function as structure and design, security and performance. Based on the structure of selected hash functions, it can be concluded that only BLAKE function and its successors (BLAKE2 and BLAKE3) allow its use in the *SDEx* method. Like the functions of the SHA-2 family (SHA-256 and SHA-512), it meets the main requirement that the input message block size for each iteration of the hash function should be twice as large as the final and partial hash. Taking into account the security aspect, the BLAKE function, along with the winner - the Keccak algorithm, fared best compared to the other finalists of the SHA-3 competition. A similar situation is in comparing the speed of hash functions in software implementation - BLAKE again proved to be one of the best.

Summarizing the use of the BLAKE hash function, and in particular its successors BLAKE2 and BLAKE3, in the *SDEx* method will allow for quite significant improvement of it in terms of speed, maintaining an appropriate level of security. This method can be several (using BLAKE2) or a dozen or so (using BLAKE3) times faster than its original version using the SHA-2 family functions. What is important, it can also be many times faster than the nowadays used AES algorithm, of course taking into account a similar level of security.

ACKNOWLEDGMENTS

This work was supported by the grant from the Polish National Science Centre (no. 2020/04/X/ST6/00407).

REFERENCES

[1] Podlaski K., Hłobaż A., Milczarski P. (2016) Secure Data Exchange Based on Social Networks Public Key Distribution. In: Mandler B. et al. (eds) Internet of Things. IoT Infrastructures. IoT360 2015. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 169. Springer, Cham. https://doi.org/10.1007/978-3-319-47063-4_5

[2] Hłobaż A., Podlaski K., Milczarski P. (2017) Enhancements of Encryption Method Used in SDEx. In: Gaj P., Kwiecień A., Sawicki M. (eds) Computer Networks. CN 2017. Communications in Computer and Information Science, vol 718. Springer, Cham. https://doi.org/10.1007/978-3-319-59767-6_11

[3] P. Milczarski, A. Hłobaż and K. Podlaski, "Analysis of enhanced SDEx method," *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Bucharest, 2017, pp. 1046-1050, https://doi.org/10.1109/IDAACS.2017.8095245

[4] A. Hłobaż, "Statistical Analysis of Enhanced SDEx Encryption Method Based on SHA-256 Hash Function," *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, Osnabrueck, Germany, 2019, pp. 238-241, https://doi.org/10.1109/LCN44214.2019.8990714

[5] A. Hłobaż, "Statistical Analysis of Enhanced SDEx Encryption Method Based on SHA-512 Hash Function," *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, Honolulu, HI, USA, 2020, pp. 1-6, https://doi.org/10.1109/ICCCN49398.2020.9209663

[6] Chang, S. H., Ray A. Perlner, W. Burr, M. Turan, J. Kelsey, S. Paul and Lawrence E. Bassham. "Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition." (2012), http://doi.org/10.6028/NIST.IR.7896

[7] Preneel B., Davies-Meyer hash function. In van Tilborg, H.C.A., ed.: Encyclopedia of Cryptography and Security, Boston, MA, pp. 136-146, Springer US, 2005.

[8] National Institute of Standards and Technology: Secure hash standard (shs). Technical report, NIST, 2008.

[9] https://www.cryptopp.com/benchmarks.html

[10] Aumasson JP., Meier W., Phan R.CW., Henzen L. (2014) Specification of BLAKE. In: The Hash Function BLAKE. Information Security and Cryptography. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44757-4_3

[11] Grøstl documentation: https://www.groestl.info/groestl-implementation-guide.pdf

[12] JH documentation: https://www3.ntu.edu.sg/home/wuhj/research/jh/jh_round3.pdf

[13] Keccak documentation: https://keccak.team/keccak_specs_summary.html

[14] Skein documentation: https://www.schneier.com/wp-content/uploads/2015/01/skein.pdf

[15] BLAKE2 documentation: https://www.blake2.net/blake2.pdf

[16] O'Connor J., Aumasson J.-P., Neves S., Wilcox-O'Hearn Z, BLAKE3 one function, fast everywhere: https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf

[17] I. F. Alshaikhli, M. A. Alahmad and K. Munthir, "Comparison and Analysis Study of SHA-3 Finalists," 2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Kuala Lumpur, Malaysia, 2012, pp. 366-371, https://doi.org/10.1109/ACSAT.2012.64

[18] D. Bernstein and T. Lange (editors), eBASH: ECRYPT Benchmarking of All Submitted Hashes, http://bench.cr.yp.to/ebash.html

[19] External Benchmarking Extension (XBX), http://xbx.das-labor.org/trac

[20] Announcing Approval of Federal Information Processing Standard (FIPS) 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, and Revision of the Applicability Clause of FIPS 180-4, Secure Hash Standard: https://www.federalregister.gov/documents/2015/08/05/2015-19181/announcing-approval-of-federal-information-processing-standard-fips-202-sha-3-standard