

# Hash algorithm comparison through a PIC32 microcontroller

Asmae Zniti, Nabih El Ouazzani

Faculty of sciences and Technology (FST), University Sidi Mohamed Ben Abdellah, Laboratory of Signals Systems and Components, (LSSC), Fez, Morocco

## Article Info

### Article history:

Received Oct 10, 2022

Revised Jan 5, 2023

Accepted Jan 29, 2023

### Keywords:

Cycles per byte

Hash functions

Memory size

PIC32

SHA-3

## ABSTRACT

This paper presents a comparative study involving SHA-3 final round candidates along with recent versions of hash algorithms. The proposed comparison between hash functions is performed with respect to cycles per byte and memory space. Tests are also carried out on a PIC32-based application taking into account several input cases, thus resulting in a set of ranked algorithms in terms of their specified metrics. The outcome of this work represents a considerable contribution in data protection and information security in relation to various communication and transmission systems, serving as a handy reference for developers to select an appropriate hash algorithm for their particular use condition.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Asmae Zniti

Signals Systems and Components Laboratory (LSSC)

Sidi Mohamed Ben Abdellah University

Faculty of sciences and Technology (FST), Route d'Imouzzer, Fez, Morocco

Email: znitiasmae@gmail.com

## 1. INTRODUCTION

Modern technology has made electronic devices smarter, more autonomous and better connected to the external world thereby making information security and data integrity questions of the utmost importance. In order to protect data from malicious attacks and to ensure information authenticity, various cryptographic hash functions have been developed. A hash function converts binary sequences of arbitrary length, called messages, into binary strings of a specific length [1], called message digests or hash values [2], [3]. Hash functions are mainly used for the confirmation of data integrity [4] as well as for message authentication codes (MAC) [5] or hash message authentication code (HMAC) [6]–[8].

Although, Message-Digest Algorithm 5 (MD5), Secure Hash Algorithm 1 (SHA-1) and Secure Hash Algorithm 2 (SHA-2) [9]–[12] have long been the three most popular cryptographic hash functions, significant recent advancements in cryptanalysis have triggered concerns over the level of security in the future. As a result, the National Institute of Standards and Technology (NIST) announced an open competition for the development of Secure Hash Algorithm 3 (SHA-3) [13], a new cryptographic hash function which could be an alternative in case the other functions are broken. In December 2010, the competition was narrowed down to five finalists: Blake, Grøstl, JH, Keccak, and Skein. In October 2012, the Keccak algorithm was selected as the SHA-3 winner [14], [15] while all five finalists continued to be used in several applications [16], [17] based on their individual strengths [18].

Several investigations have been performed to compare the performance of these algorithms and have provided insights. In [19], the author presents a performance evaluation of various hash functions on an ATMEL AVR ATtiny45 8-bit microcontroller. The results suggest that Blake offers the best performance among the SHA-3 finalists, followed by Grøstl, Keccak, Skein, and JH.

Rajeev Sobti *et al.* present in [20], a comparison of the performance of three of the five finalists for the SHA-3 hash function standard on the ARM Cortex M3 processor. In terms of the cycles per byte metric, Grøstl turns out to outperform Blake and JH. With increasing input size, JH shows a significant drop in cycles per byte value whereas Blake and Grøstl have only a slight change.

In [21], the performance of SHA-3 finalists are discussed in terms of execution time on a 64-bit Intel Core processor where it is found that Skein and Blake perform better than the other candidates insofar as the digest length and block size followed by Grøstl, Keccak, and JH in this order. Sobti and Geetha [22] discuss the results of a performance evaluation of the same algorithms on the ARM Cortex A8 architecture. The evaluation is achieved based on the cycles per byte metric, the results indicating that Blake, Keccak and Skein are the most efficient algorithms, closely followed by Grøstl and JH. For shorter messages, Blake's performance is better than Skein but as the message size increases, Skein starts improving to perform almost as well as Blake. For longer messages, Skein is the most efficient algorithm and narrowly overtakes Keccak and Blake.

Similar comparisons are carried out on the ARM Cortex-M4 processor highlight that Blake is the best performer, followed by Keccak and Skein [23], Grøstl and JH occupying the lowest two positions. For long messages, the results are similar to those of short messages, with Blake again performing the best, followed by Keccak and Skein. A study of the previously mentioned algorithms based on ARM Cortex-A9 processor yields similar results [24]. Blake emerges as the clear victor, with a far superior rate of execution compared to its counterparts whereas Grøstl lags behind the other contenders. There seems to be plenty of conflicting information available as to which of the SHA-3 finalists is the overall best performing algorithm. Some studies suggest that Blake is the best, while others propose the Skein algorithm; however, it is important to keep in mind that the conclusions drawn in each of these studies are based on specific hardware and software implementations as well as message sizes.

This paper provides a comprehensive and up-to-date comparison of hash algorithms by examining the SHA-3 finalists as well as other newer versions such as Blake2 (2012), Shake (2015), Kangaroo Twelve (2016) and Blake3 (2020) on a PIC32 microcontroller platform. The main metrics considered include the number of cycles per byte required for a particular algorithm to fixed hash inputs, and the ROM capacity needed to store the program of the cryptographic algorithm. The choice of a PIC32 is strongly motivated by the fact that this microcontroller is integrated in several data transfer applications and sophisticated systems such as automotive embedded networks.

The rest of the paper is organized as follows. Section 2 gives a brief introduction about SHA-3 algorithm contenders. The tools adopted to carry out the evaluation of the algorithms are presented in section 3. Section 4 shows the experimental results and draws comparisons between different algorithms. Finally, the conclusions of the work are given in section 5.

## 2. SHA-3 CONTENDERS

### 2.1. SHA-3 finalists

The SHA-3 finalists are listed and defined as:

- a. Keccak: Keccak hash function is the winner of the SHA-3 competition. It is based on sponge construction and consists of seven permutation functions of different bit lengths, used in XOR and rotation operations.
- b. Blake: the Blake algorithm is an adaptation of the ChaCha stream that carries out transformations on 4 words involving XOR and a bit rotation with a fast implementation. A total of 10 to 14 rounds of ChaCha functions are used according to the required size of the message digests [25].
- c. Skein: Skein is based on the Threefish block cipher and is compressed by using mathematical operations such as addition, XOR and rotation to create a MIX function. The Skein algorithm requires 72 or 80 rounds depending on the block size needed to run the algorithm.
- d. Grøstl: the Grøstl algorithm was developed by a team of cryptographers from Technical University of Denmark (DTU) and TU Graz. It extracts elements from the advanced encryption standard (AES) cipher algorithm [26]. Since several optimizations on AES have been performed on software and hardware over the years, its throughput is at a high level.
- e. JH: the JH algorithm was created by Hongjun Wu. Inspired by the AES and Serpent cipher algorithms, it consists of 42 rounds of execution.

### 2.2. New algorithms versions

The new versions are as follows:

- a. Blake2: Blake2 is an improved version of Blake, created in fall 2012 after Keccak was declared as SHA-3 [27]. It was initially engineered to leverage Blake's high efficiency and security and then adapted to modern applications prioritizing simplicity and usability. Blake2 has two main flavors, Blake2b designed for 64-bit platforms and Blake2s for smaller architectures [28].

- b. Shake: in August 2015, Shake was announced by NIST as part of the SHA-3 family. It has two extendable Output Functions (XOFs), Shake-128 and Shake-256 [29].
- c. Kangaroo Twelve: Kangaroo Twelve is a fast and secure arbitrary output-length hash function combining many features in common with Shake-128 such as the sponge construction, the cryptographic primitive, the eXtendable Output Function (XOF) and the 128-bit security strength. Kangaroo Twelve is based on a reduced round version (12 rounds) of SHA-3 permutation function (Keccak [1600]), proposed by Bertoni *et al.* [30] with the purpose of being faster than SHA-3 and Shake.
- d. Blake3: released in 2020, this successor of Blake 2 was designed to run even faster [31]. Blake3 compression function is very close to that of Blake2s, its main difference being the number of rounds reduced from 10 to 7.

### 3. IMPLEMENTATION DETAILS

#### 3.1. Hardware platform

PIC32 is one of the most widely-used processors in embedded systems such as the automotive, and is able to deliver high-performance computing and power efficiency at a reduced cost. The under-test algorithms are run on the chipKIT Max32, a microcontroller board based on the Microchip PIC32MX795F512 which functions at up to 80 MHz with a Flash memory of 512KB and a RAM of 128 KB. It can be programmed by means of the multi-platform integrated development environment (MPIDE).

#### 3.2. Procedure and metrics

The entire implementation process and evaluation follow the requirements listed as:

- Each candidate algorithm has at least four cryptographic hash functions, called “algorithm\_Name”-X, where the suffix X stands for the corresponding length of the output which can be 224,384,256 or 512 bits.
- The basic metrics considered for evaluation are the code size and the number of cycles per byte. The latter refers to the necessary number of cycles required by a hash function divided by the number of input bytes. This metric is chosen over execution speed as it does not change regardless of device frequency. The cycle consumption is measured multiple times and then the average is calculated to record the readings.
- Performances are measured according to 5 different message lengths, ranging from a very small (smaller than one block) to a large sequence, which are 30, 48, 78, 150 and 400 bytes.

### 4. RESULTS AND DISCUSSION

Along the same line as in section 3, algorithm performances are determined based on two primary metrics: memory space and processing time for a variety of message sizes. The graph in Figure 1, illustrates the code size expressed in bytes for candidates with a 256-bit hash size. Through this evaluation, it is obvious that Blake2s needs the least memory space, which is 16756 bytes, whereas Keccak uses the largest size, reaching 56476 bytes. However, as the total capacity available within the platform substantially outsizes those needed by various codes, this problem is less likely to be influential in this case.

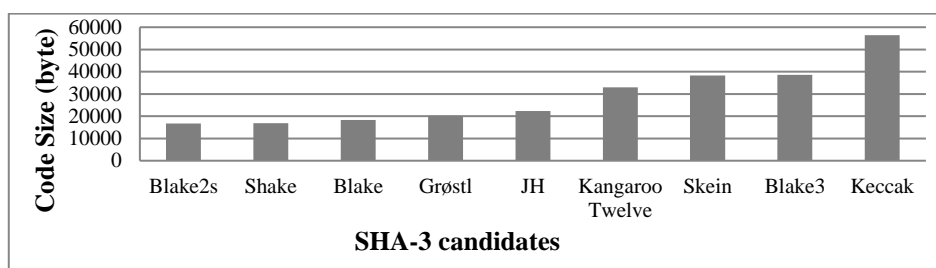


Figure 1. Code size in bytes

Conversely, Figures 2-3 illustrate the cycles per byte for four distinct hash sizes (224, 265, 384 and 512-bit) with varying input lengths. Figure 2 reveals the cycles per byte for three shorter inputs (30, 46, and 78 bytes) while Figure 3 examines the cycles per byte for two extended ones (150 and 400 bytes). These figures shed light on the performance of the hash functions for different input lengths, thereby providing a comprehensive overview of the efficiency of the hash algorithms.

From Figures 2(a) and 2(b) where results of 224 bits and 256 bits, respectively, are shown, it should be noted that Blake3 is the fastest for input sizes of 30- and 46-byte followed by Blake2s, Blake, Kangaroo

Twelve, Skein512, JH, Keccak Shake-128 and Shake-256, then Groestl. However, for the 78-byte input size the order remains the same except that Blake and Kangaroo Twelve are reversed. With regard to the 384-bit and 512-bit hash sizes, hardware simulation results are presented in Figures 2(c) and 2(d), respectively. The same order of algorithms is observed for input sizes 30, 46 bytes, with Blake3 at the top followed by Kangaroo Twelve, Skein512, Blake2s, Blake JH Keccak Shake-128, Shake-256 then Groestl. The 78-byte input size follows a similar trend except in Figure 2(c) where the order of Skein512 and Blake2s is reversed whilst in Figure 2(d) both Skein512 and Blake2s are reversed while Keccak moved down two places.

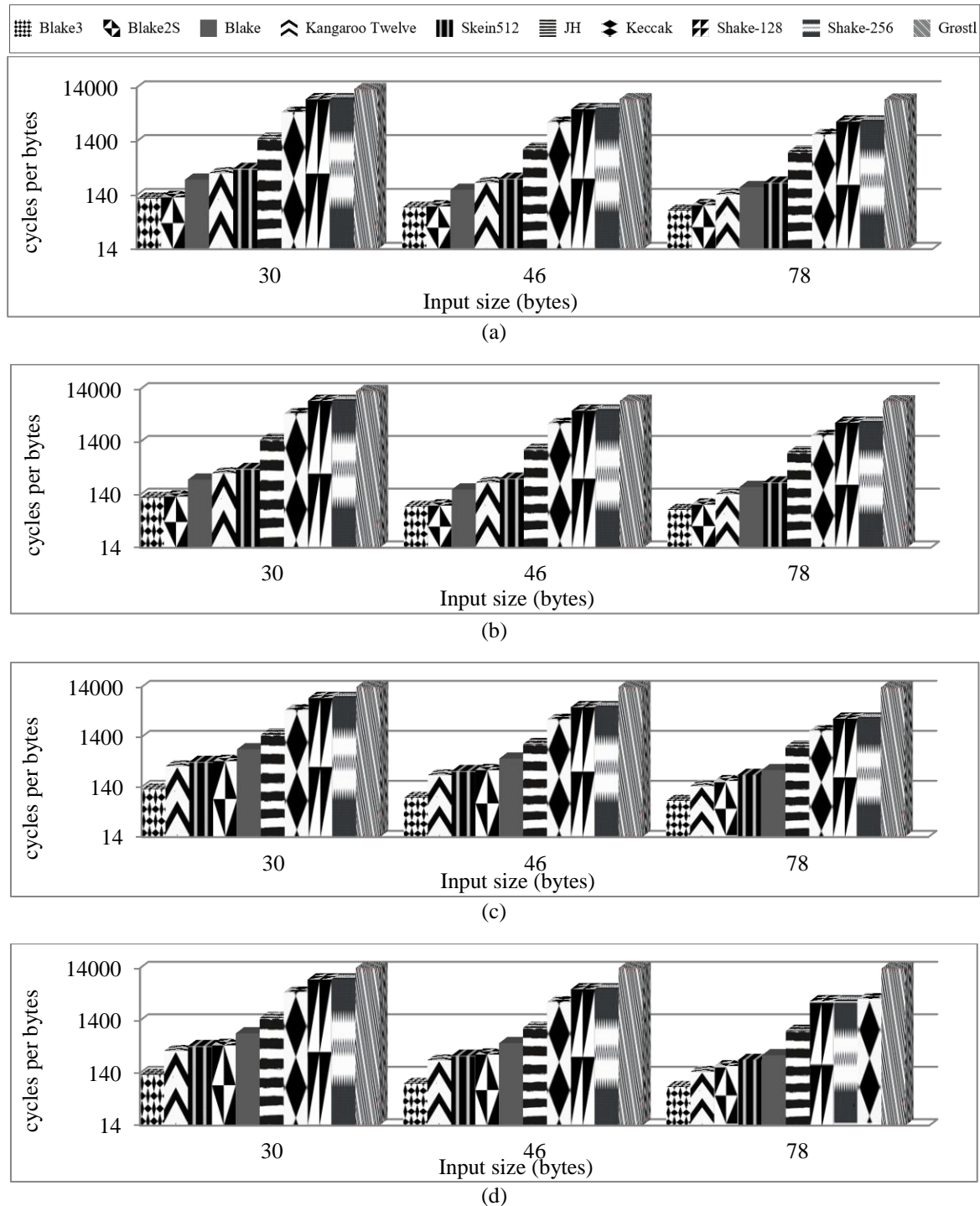


Figure 2. Cycles per byte for input sizes of 30, 46 and 78 bytes (a) 224-bit hash size, (b) 256-bit hash size, (c) 384-bit hash size, and (d) 512-bit hash size

In addition, from the 224-bit and 256-bit hash size results illustrated in Figure 3(a) and Figure 3(b), respectively, we notice that Blake3 is the fastest for input size of 150-byte, followed by Blake2s, Kangaroo Twelve, Blake, Skein512, JH, Shake-128, Keccak, Shake-256 and then Groestl. A somewhat similar pattern can be seen at the 400-byte input size except that the order of Blake and Skein512 is reversed in comparison to the former case.

Figure 3(c) reveals that Blake3 is the most effective algorithm for both 150-byte and 400-byte inputs, while when it comes to the input size of 150-byte, Kangaroo Twelve and Skein512 are close second and third respectively. These are followed by Blake2s, Blake, JH Shake-128 and Keccak Shake-256, leaving Groestl trailing behind at the bottom of the list. A similar hierarchy is observed for 400-byte inputs, with the exception of Keccak that moves up one position towards the end of the order. This pattern remains the same for 512-bit hash sizes as demonstrated in Figure 3(d) through both 150 and 400 byte inputs.

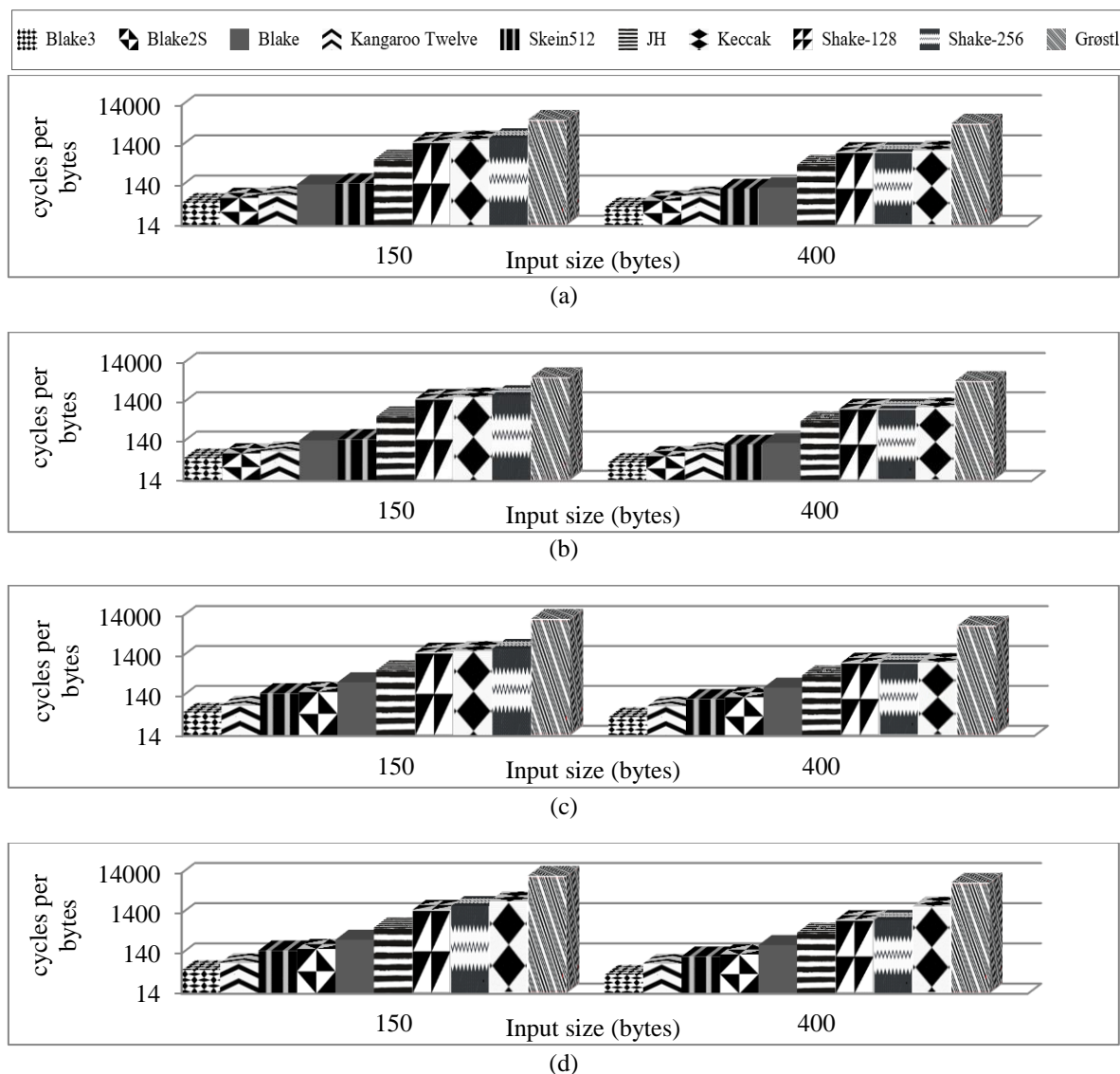


Figure 3. Cycles per byte for input sizes of 150 and 400 bytes (a) 224-bit hash size, (b) 256-bit hash size, (c) 384-bit hash size, and (d) 512-bit hash size

In terms of cycles per byte, it is obvious from Figures 2-3, that the Blake3 candidate has the highest computing speed, beating the other algorithms in all situations. In contrast, Grøstl ranks the lowest in all cases. In order to better illustrate the results, an alternative presentation can be adopted. First, a score is assigned to every position obtained in the ranking, starting with 10 and ending with 1, as shown in Table 1. Second, the overall rankings are summarized in Table 2 indicating that Blake3 is the winner, followed by Kangaroo Twelve and Blake2s, while keccak, shake and Grøstl come in order with the lowest ratings.

Table 1. Ranking scores

Ranking	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>	4 <sup>th</sup>	5 <sup>th</sup>	6 <sup>th</sup>	7 <sup>th</sup>	8 <sup>th</sup>	9 <sup>th</sup>	10 <sup>th</sup>
Score	10	9	8	7	6	5	4	3	2	1

Table 2. SHA-3 candidate ranking

Candidate	Ranked										Score
	1	2	3	4	5	6	7	8	9	10	
Blake3	20	0	0	0	0	0	0	0	0	0	200
Blake2S	0	10	2	8	0	0	0	0	0	0	162
Blake	0	0	4	4	12	0	0	0	0	0	132
Kangaroo twelve	0	10	6	4	0	0	0	0	0	0	166
Skein512	0	0	8	4	8	0	0	0	0	0	140
JH	0	0	0	0	0	20	0	0	0	0	100
Keccak	0	0	0	0	0	0	11	3	6	0	65
Shake-128	0	0	0	0	0	0	9	11	0	0	69
Shake-256	0	0	0	0	0	0	0	6	14	0	46
Grøstl	0	0	0	0	0	0	0	0	0	20	20

## 5. CONCLUSION

Throughout this work, various SHA-3 algorithms have been subjected to a thorough comparative analysis related to important parameters such as memory size and number of cycles per byte. In order to carry out this investigation, a hardware implementation was set up on a PIC32, thereby running several different simulations. Several input sizes have been dealt with considering different output lengths in order to give a more complete picture of the performance of each algorithm. As a result and according to the specified metrics, Blake3 is at the top of the ranking. The second position based on the output size is occupied by Kangaroo Twelve in the case of long outputs and Blake2s for short outputs, while Grøstl comes in last.

The current investigation is set up to provide designers and information security protocol developers with an efficient algorithm which aims to be implemented in various platforms. Knowing that PIC32 microcontrollers are increasingly incorporated into such systems, the present contribution can be useful, especially in the automotive industry.

## REFERENCES




- [1] J. Tchórzewski and A. Jakóbbik, "Theoretical and Experimental Analysis of Cryptographic Hash Functions," *Journal of Telecommunications and Information Technology*, vol. 1, no. 2019, pp. 125–133, Mar. 2019, doi: 10.26636/jtit.2019.128018.
- [2] A. Kuznetsov, I. Oleshko, V. Tymchenko, K. Lisitsky, M. Rodinko, and A. Kolhatin, "Performance Analysis of Cryptographic Hash Functions Suitable for Use in Blockchain," *International Journal of Computer Network and Information Security*, vol. 13, no. 2, pp. 1–15, Apr. 2021, doi: 10.5815/ijcnis.2021.02.01.
- [3] A. Mittelbach and M. Fischlin, "The Theory of Hash Functions and Random Oracles," in *The Theory of Hash Functions and Random Oracles: An Approach to Modern Cryptography*, A. Mittelbach and M. Fischlin, Eds. Cham: Springer International Publishing, 2021, pp. 1–788. doi: 10.1007/978-3-030-63287-8\_1.
- [4] M. A. AlAhmad, "Design of a New Cryptographic Hash Function – Titanium," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 10, no. 2, p. 827, May 2018, doi: 10.11591/ijeecs.v10.i2.pp827-832.
- [5] A. W. A. Qader, I. E. Salem, and H. R. Abdulshaheed, "A new algorithm for implementing message authentication and integrity in software implementations," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 5, p. 2543, Oct. 2020, doi: 10.12928/telkomnika.v18i5.15276.
- [6] M. Harran, W. Farrelly, and K. Curran, "A method for verifying integrity and authenticating digital media," *Applied Computing and Informatics*, vol. 14, no. 2, pp. 145–158, Jul. 2018, doi: 10.1016/j.aci.2017.05.006.
- [7] P. Zhang, X. Zhang, and J. Yu, "A Parallel Hash Function with Variable Initial Values," *Wireless Personal Communications*, vol. 96, no. 2, pp. 2289–2303, Jun. 2017, doi: 10.1007/s11277-017-4298-9.
- [8] V. Rao and P. K.V., "Light-weight hashing method for user authentication in Internet-of-Things," *Ad Hoc Networks*, vol. 89, pp. 97–106, Jun. 2019, doi: 10.1016/j.adhoc.2019.03.003.
- [9] D. Rachmawati, J. T. Tarigan, and A. B. C. Ginting, "A comparative study of Message Digest 5(MD5) and SHA256 algorithm," *Journal of Physics: Conference Series*, vol. 978, p. 012116, Mar. 2018, doi: 10.1088/1742-6596/978/1/012116.
- [10] R. L. Quilala and T. F. G. Quilala, "Improved MSHA-1 algorithm with mixing method," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2144–2151, Aug. 2021, doi: 10.11591/eei.v10i4.2366.
- [11] Rohit, S. Kamra, M. Sharma, and A. Leekha, "Secure Hashing Algorithms and Their Comparison," in *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, Mar. 2019, pp. 788–792.
- [12] B. U. I. Khan, R. F. Olanrewaju, M. A. Morshidi, R. N. Mir, M. L. B. M. Kiah, and A. M. Khan, "Evolution and analysis of secured hash algorithm (SHA) family," *Malaysian Journal of Computer Science*, vol. 35, no. 3, pp. 179–200, Jul. 2022, doi: 10.22452/mjcs.vol35no3.1.
- [13] S. Chang *et al.*, "Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition," National Institute of Standards and Technology, Nov. 2012, doi: 10.6028/nist.ir.7896.
- [14] S. Ghoshal, P. Bandyopadhyay, S. Roy, and M. Baneree, "A Journey from MD5 to SHA-3," in *Lecture Notes in Networks and Systems*, Springer Singapore, 2020, pp. 107–112, doi: 10.1007/978-981-15-1624-5\_11.






- [15] P. P. Pittalia, "A Comparative Study of Hash Algorithms in Cryptography," *International Journal of Computer Science and Mobile Computing*, vol. 8, no. 6, pp. 147–152, 2019.
- [16] A. Zniti and N. E. Ouazzani, "Improvement of the Authentication on In-Vehicle Controller Area Networks," in *Embedded Systems and Artificial Intelligence*, Springer Singapore, 2020, pp. 23–32, doi: 10.1007/978-981-15-0947-6\_3.
- [17] M. Harikrishnan and K. V. Lakshmy, "Secure Digital Service Payments using Zero Knowledge Proof in Distributed Network," in *2019 5th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Mar. 2019, doi: 10.1109/icaccs.2019.8728462.
- [18] A. J. Hlobaž, "Analysis of the possibility of using selected hash functions submitted for the SHA-3 competition in the SDEx encryption method," *International Journal of Electronics and Telecommunications*, vol. 68, no. 1, Feb. 2022, doi: 10.24425-ijet.2022.139848.
- [19] J. Balasch *et al.*, "Compact Implementation and Performance Evaluation of Hash Functions in ATtiny Devices," in *Smart Card Research and Advanced Applications*, Springer Berlin Heidelberg, 2013, pp. 158–172, doi: 10.1007/978-3-642-37288-9\_11.
- [20] R. Sobti, G. Geetha, and S. Anand, "Performance Comparison of Grøstl, JH and BLAKE - SHA-3 Final Round Candidate Algorithms on ARM Cortex M3 Processor," in *2012 International Conference on Computing Sciences*, Phagwara, India, Sep. 2012, pp. 220–224, doi: 10.1109/iccs.2012.57.
- [21] R. K. Dahal, J. Bhatta, and T. N. Dhamala, "Performance Analysis of Sha-2 and Sha-3 Finalists," *International Journal on Cryptography and Information Security*, vol. 3, no. 3, pp. 1–10, Sep. 2013, doi: 10.5121/ijcis.2013.3301.
- [22] R. Sobti and G. Geetha, "Performance Comparison of Keccak, Skein, Grøstl, Blake and JH: SHA-3 Final Round Candidate Algorithms on ARM Cortex A8 Processor," *International Journal of Security and Its Applications*, vol. 9, no. 12, pp. 367–384, Dec. 2015, doi: 10.14257/ijisa.2015.9.12.34.
- [23] R. Sobti and G. Ganesan, "Performance Evaluation of SHA-3 Final Round Candidate Algorithms on ARM Cortex-M4 Processor," *International Journal of Information Security and Privacy*, vol. 12, no. 1, pp. 63–73, Jan. 2018, doi: 10.4018/ijisp.2018010106.
- [24] A. Zniti and N. E. Ouazzani, "A comparative study of hash algorithms with the prospect of developing a CAN bus authentication technique," *International journal of electrical and computer engineering systems*, vol. 13, no. 9, pp. 741–746, Dec. 2022, doi: 10.32985/ijeces.13.9.2.
- [25] S. Barbero, E. Bellini, and R. H. Makarim, "Rotational analysis of ChaCha permutation," *Advances in Mathematics of Communications*, vol. 0, no. 0, p. 0, 2021, doi: 10.3934/amc.2021057.
- [26] A. Menezes and D. Stebila, "The Advanced Encryption Standard: 20 Years Later," *IEEE Security and Privacy*, vol. 19, no. 6, pp. 98–102, Nov. 2021, doi: 10.1109/msec.2021.3107078.
- [27] H. E. Makhtout and Y. Bentaleb, "Comparative Study of Keccak and Blake2 Hash Functions," in *Networking, Intelligent Systems and Security*, Springer Singapore, 2021, pp. 343–350, doi: 10.1007/978-981-16-3637-0\_24.
- [28] M.-J. O. Saarinen and J.-P. Aumasson, "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)," Internet Engineering Task Force, Request for Comments RFC 7693, Nov. 2015, doi: 10.17487/RFC7693.
- [29] M. J. Dworkin, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," National Institute of Standards and Technology, Jul. 2015, doi: 10.6028/nist.fips.202.
- [30] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, R. V. Keer, and B. Viguier, "KangarooTwelve: fast hashing based on Keccak-p," in *Applied Cryptography and Network Security*, Springer International Publishing, 2018, pp. 400–418, doi: 10.1007/978-3-319-93387-0\_21.
- [31] J. O'Connor, J.-P. Aumasson, S. Neves, and Z. W. O'Hearn, "BLAKE3: one function, fast everywhere," BLAKE3 team, Feb. 22, 2023.

## BIOGRAPHIES OF AUTHORS



**Asmae Zniti**    received a Bachelor's degree in Industrial Engineering, in 2015, and a Master's degree from Sidi Mohamed Ben Abdellah University, Faculty of Sciences and Technology, Fez, Morocco (FST-Fez), in 2017. She is currently pursuing her PhD degree in Automotive Electronics at the Laboratory of Signals, Systems and Components (LSSC), FST-Fez. Her research interests include controller area network security, cryptographics and authentication. She can be contacted at email: znitiasmae@gmail.com.



**Nabih El Ouazzani**    received his Ph.D degree in microwave circuits, especially microwave filters, from the University of Limoges-France at the Xlim institute in 1995. Since then he has been a professor at the Faculty of Sciences and Technology-Fez, Morocco (FST-Fez) and a member of the Laboratory of Signals, Systems and Components (LSSC). He carries out several activities with respect to research and education. The disciplines relevant to his expertise are high- frequency technology and telecommunication devices. He is also involved in the research area of embedded systems, particularly in the automotive field with regard to circuit protection and network security. He has co-organized many international conferences related to the subjects of ICT and telecommunication in Morocco and has participated in numerous scientific committees. He can be contacted at email: nabih.elouazzani@usmba.ac.ma.