A comparative study of hash algorithms with the prospect of developing a CAN bus authentication technique

Original Scientific Paper

Asmae Zniti
Sidi Mohamed Ben Abdellah University,
Faculty of Sciences and Technologies (FST), Laboratory of Signals, Systems and Components (LSSC)
Route d'Imouzzer, Fez, Morocco
znitiasmae@gmail.com

Nabih El Ouazzani
Sidi Mohamed Ben Abdellah University,
Faculty of Sciences and Technologies (FST), Laboratory of Signals, Systems and Components (LSSC)
Route d'Imouzzer, Fez, Morocco
nabih.elouazzani@usmba.ac.ma

Abstract *– In this paper, the performances of SHA-3 final round candidates along with new versions of other hash algorithms are analyzed and compared. An ARM-Cortex A9 microcontroller and a Spartan -3 FPGA circuit are involved in the study, with emphasis placed on the number of cycles and the authentication speed. These hash functions are implemented and tested resulting in a set of ranked algorithms in terms of the specified metrics. Taking into account the performances of the most efficient algorithms and the proposed hardware platform components, an authentication technique can be developed as a possible solution to the limitations and weaknesses of automotive CAN (Controlled Area Network) bus – based embedded systems in terms of security, privacy and integrity. From there, the main elements of such a potential structure are set forth.*

Keywords*: Hash algorithms, SHA-3, ARM-Cortex A9, FPGA, Number of Cycles, Authentication, CAN Bus*

1. INTRODUCTION

In modern technology, embedded devices are smarter, more autonomous and better connected. Therefore, questions of information security are increasingly sensitive and have become of an utmost importance.

Hash functions are used for data integrity confirmation and as message authentication codes (MAC) or hash message authentication codes (HMAC). There exist several families of Security Hash Functions such as SHA-0, SHA-1, SHA-2, and SHA-3. In 1997, the National Security Agency NSA detected a major flaw in SHA-0 and so a new, improved algorithm –SHA-1 – was developed. This one, however, also suffered from severe cryptographic weaknesses and was later replaced by SHA-2 in 2002. Although as of yet, no significant cryptographic issue has been found in SHA-2, it was considered to be algorithmically too closely related to SHA-1. Since SHA-0, SHA-1 and SHA-2 suffer from these same limitations as well, we dismiss them, choosing instead to address the SHA-3 finalists [1] (Blake, Skein, JH, Grøstl, Keccak).

A number of studies have been conducted to compare the effectiveness of various hash algorithms. In 2013, R.K. Dahal et al. published a paper examining the performances of the SHA-3 finalists in addition to the widely used SHA-2 [2]. Their findings indicated that, among the SHA-3 finalists, Skein and Blake were the most effective while, according to digest length and block size, Grøstl, Keccak and JH followed.

In 2015, R. Sobti and Ganesan G. Geetha provided a performance evaluation of the SHA-3 finalists on ARM Cortex A8-based devices [3]. The results showed that Grøstl and JH were not efficient while Skein was a better option for long messages. They also found that Blake was a good option, as it outperformed Keccak for both 224 / 256- and 512-bit hash.

In 2018, a similar comparison was carried out on the ARM Cortex-M4 platform [4] with different results. Blake is the best choice as it performs better than all the algorithms for all message digests, regardless of the input size. Skein is also a good second option if, for higher security margins, we need a 512-bit hash instead of a 256-bit hash.

Numerous factors affect the performances of a hash algorithm. For example, Skein may be a better choice for long messages, while Blake may be better for short messages. It is important to consider the specific application when choosing the appropriate hash algorithm since no single algorithm is necessarily the best in all cases.

The proposed study provides a more comprehensive and up-to-date comparison of hash algorithms by implementing the SHA-3 finalists, as well as other newer common hashing functions, such as Blake2, Shake, Kangaroo Twelve and Blake3 on ARM cortex-A9 also taking their speed performance on an FPGA platform into consideration [5, 6]. As a result, the best algorithm is determined and the validity of the speed is verified.

As a potential application, an enhanced automotive CAN bus network [7] can be implemented based on a new structure relying on an authentication technique [8, 9, 10]. Considering the issue that a CAN bus lacks the security features such as message authentication and is therefore vulnerable to spoofing attacks [11, 12, 13], an effective solution may consist in implementing a hash process. Each message on the bus must be hashed with a key using a hash algorithm to form a message authentication code (MAC), thus allowing each node to check the authenticity of a received message. The process is mainly intended to generate two frames. The first deals with the transmission of data, while the second manages the authentication and filtering of unauthorized frames.

The structure of the paper is organized as follows. In Section 2, we briefly describe the SHA-3 Hash Function contenders. In Section 3, we present an overview of the CAN bus along with the principle of an authentication solution. Section 4 gives an outline of the methodology and the tools used for evaluation. Section 5 is dedicated to the hardware simulation results and performance analysis. The hash algorithm applied to produce digests is selected and the authentication time is calculated in section 6. Finally, Section 7 provides the conclusion.

2. AVAILABLE HASH ALGORITHMS

2.1. SECURE HASH ALGORITHM

A secure hash algorithm (SHA) is a standard invented by the National Institute of Standards and Technology (NIST) [14], based on the Message Digest (MD5) algorithm [15]. As the SHA-1 algorithm has already been cracked and SHA-2 proved to suffer from the same weaknesses, the NIST launched a public competition in November 2007 to make out a new cryptographic hash algorithm called SHA-3. In October 2012, NIST declared the Keccak algorithm as the winner of the SHA-3 competition.

2.2. THE SHA-3 FINALISTS

Keccak [16] was chosen from a range of five very strong candidates (Skein [17], Blake [18], Grøstl [19] and, JH [20]). NIST stated in its final report [21], that all five finalists had acceptable performances, and that any of the finalists would have represented an effective option for SHA3. In terms of performances, the report noted that some of the five algorithms operate well in software, while others appear more efficient in hardware.

2.3. THE BLAKE ALGORITHM FAMILY

Blake2 [22] is an improved version of Blake, provided in 2012 after Keccak was selected as SHA3. Blake2 was engineered to take full advantage of Blake's strengths and optimize it for modern applications. Blake2 comes in two main types: Blake2b which is optimized for 64-bit platforms and Blake2s for smaller architectures.

A successor of Blake 2, Blake 3 [23], created in 2020, was developed to be as fast as possible. The compression function of Blake3 is closely based on that of Blake2s.

2.4. THE KECCAK ALGORITHM FAMILY

Shake was declared by NIST in August 2015 as a part of the SHA-3 family. It combines two eXtendable Output Functions (XOFs), Shake128 and Shake256.

Kangaroo Twelve [24] is another XOF based on a reduced number of rounds (12 rounds) of the SHA-3 permutation function (Keccak [1600]). It is designed to be faster than SHA-3 and Shake while maintaining its flexibility and security.

3. CAN BUS OVERVIEW

3.1. CAN BUS PROTOCOL

The Controller Area Network (CAN) [7] is a serial communication bus that operates according to a specific standard for efficient and reliable information transmission between sensors, actuators, controllers, and other nodes in real-time applications.

On a CAN bus, the communication between different Electronic Control Units (ECUs) is achieved through four frames: data frame, remote frame, error frame, and overload frame.

As an example, Fig. 1 shows all the fields that make up the whole data frame. The data field length can reach up to 8 bytes, depending on the Data Length Code (DLC) word. A unique identifier is also assigned in order to manage both data transmission priorities between different nodes and filtering these upon reception. The size of the Identifier field is 11 bits for CAN version 2.0A and 29 bits for version 2.0B.

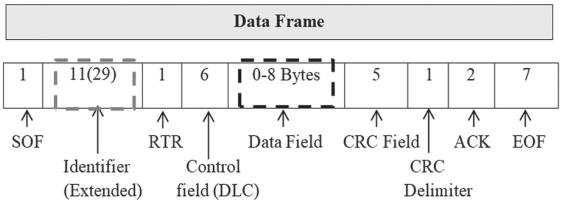

Fig. 1. CAN data frame structure

3.2. CAN BUS LIMITATIONS

As previously mentioned, the CAN bus is vulnerable to attacks perpetrated by malicious codes, leading to software damage and physical harm [25]. Amongst the weaknesses, we find:

- Non-confidentiality and transmission of unencrypted messages, which animate replay attacks and vehicle espionage.
- Absence of the authenticity and non-repudiation, which allows attackers to send arbitrary frames on the network or even transmit valid messages to trigger certain actions.
- Integrity: An attacker is able to add, delete, or modify any type of data carried by the relayed message.
- Availability: By sending high-priority messages, nodes are prevented from responding, which causes a denial of service (DoS), and consequently affects the system availability.

3.3. PRINCIPLE OF AN AUTHENTICATION TECHNIQUE

The fundamental idea depends on a system that includes a monitoring node made from an FPGA, equipped with a particular CAN controller, responsible for authenticating each message by verifying the MAC, generated by (1):

$$MAC=\text{hash function}(IDi,Di,FCi,Keyi) \quad (1)$$

where IDi is the CAN-ID (11 bits), Di indicates the data of the message i (64 bits), FCi represents a complete monotones counter for the message i of 32 bits, and KEYi denotes the encryption key for the node i encoded on 128, 256 or 512 bits.

Message data and the MAC are transmitted on two separate frames.

The planned protocol, illustrated in Fig. 2, will consist in computing the hash value by means of the CPU-based nodes participating in the communication, and performing the authentication of each CAN packet thanks to an FPGA-based monitoring node in charge of checking the hash value already calculated with its source.

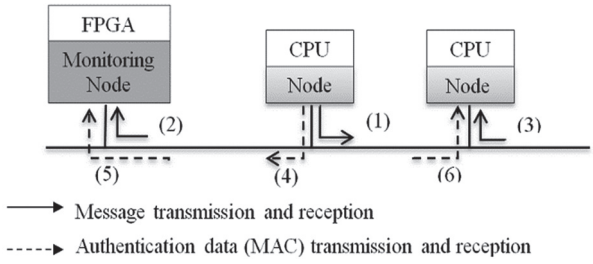

Fig. 2. Under-consideration communication protocol

4. ALGORITHM IMPLEMENTATION

4.1. PROCEDURE DETAILS

The flowchart in Fig. 3 shows the entire process of analyzing and comparing the performances in terms of the number of cycles and the execution time.

The algorithms of interest are run on an ARM Cortex A9-based platform, ranked according to their performances and compared to those obtained by means of an FPGA circuit as described in [5, 6].

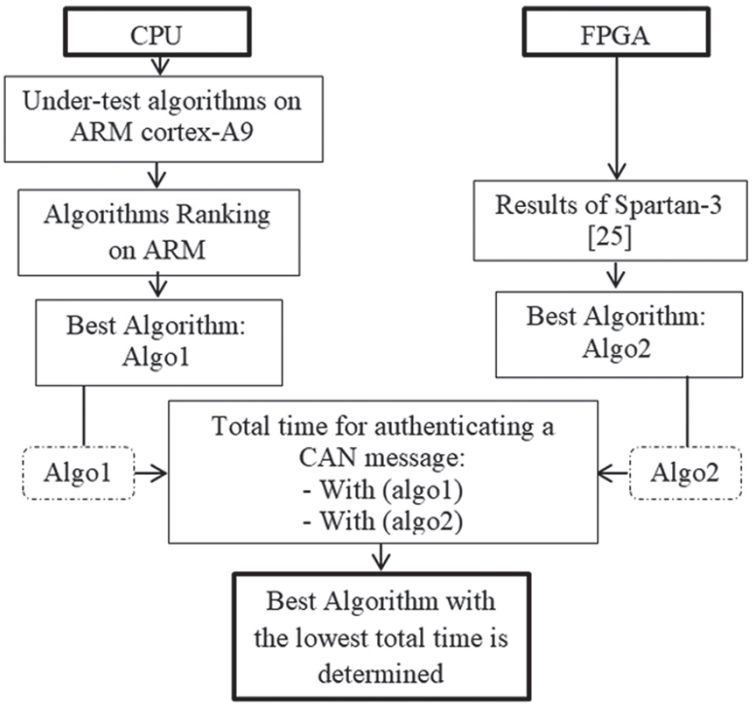

Fig. 3. Evaluation process of performance

4.2. ARM IMPLEMENTATION DATA

Knowing that 32-bit microcontrollers are widely used in embedded systems, particularly in the automotive industry, and in order to carry out simulations as closely as possible to real cases, the ARM Cortex A9 has been chosen as a testing tool. Three input sizes are considered depending on key bit lengths according to (1). The results are shown in Fig. 4.

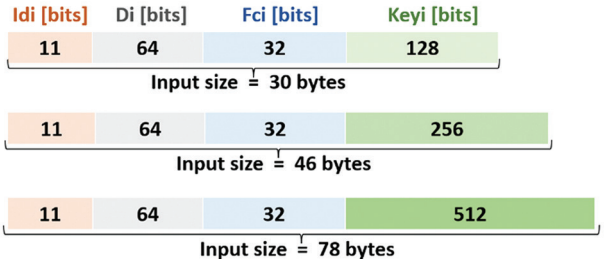

Fig. 4. Considered hash input sizes

In addition, running a hash algorithm on a Cortex A9-based platform requires a specific number of cycles regardless of the CPU frequency. As a result, the number of cycles is chosen as a metric for assessing performance.

5. RESULTS AND ANALYSIS

5.1. ARM CORTEX A9-BASED SIMULATIONS

Table 1 presents the cycles needed for 256 hash size of different algorithms taking three input lengths into account.

Table 1. Number of cycles for different input sizes

Algorithm	Number of Cycles [cycles]		
	Input size = 30 Bytes	Input size = 46 Bytes	Input size = 78 Bytes
Blake2s	13338	13180	21972
Blake	18866	18926	55376
Blake3	37002	69334	82278
Kangaroo Twelve	48320	83310	169190
Skein 256-256	64080	79646	79646
Skein 512-256	64152	64588	80190
JH	261690	261472	385978
Keccak	398660	398430	398976
Shake	558136	546286	547622
Grøstl	1682462	1690474	2789116

In this particular case of short inputs (less than 78 bytes), it is obvious that Blake2s and Blake outperform all the other contenders, providing the best choice.

Indeed, the results show that Blake2s beats all the other algorithms and presents the best speed of execution while Grostl comes in a distant last place in comparison to other candidates.

Therefore, in real-case applications such as automotive networks, blake2s can be the core software code of any improved data transmission protocol.

5.2. FPGA-BASED RESULTS

As described in [5, 6], the FPGA Spartan-3 is a hardware platform that allows the considered hash algorithms to be run, highlighting major performance parameters from which a substantial advantage for the Keccak algorithm can be seen.

Table 2 mainly shows the results obtained through the FPGA of the two chosen algorithms according to the CPU and FPGA-based analysis.

Table 2. Performances of Keccak and Blake on Spartan-3 FPGA

Algorithm	Tclk [ns]	Bloc size	Rounds	Throughput [Mbps] = Blocsize / (Rounds*Tclk)
Keccak	9.75	1088	24	4650
Blake2s	43.4	512	10	1179.72

With regard to the FPGA implementation, Keccak has an advantage over Blake2s by providing the most suitable parameter values. Thus, Keccak is still considered the most appropriate solution.

6. COMPARISON OF BLAKE2S AND KECCAK

As a last step in the analysis and comparison procedure and from knowing the respective strength of Keccak and Blake2s, the focus in this section falls mainly on the entire processing time combining ARM and FPGA devices.

In the prospect of using an FPGA platform and CPU-based circuits in a protected CAN bus system, the processing time required by both components is computed by means of (2) and (3) respectively, with a CPU frequency of 667 MHz.

$$\text{Processing time (CPU)=Number of Cycle/Frequency} \quad (2)$$

$$\text{Processing time (FPGA)= Input Size/Throughput} \quad (3)$$

Table 3 shows the total time needed to generate an authentication message in this case.

Table 3. Keccak and Blake2S performances

Algorithm	Processing time [µs]					
	Input size = 30 Bytes		Input size = 46 Bytes		Input size = 78 Bytes	
	ARM	Sparatan3	ARM	Sparatan3	ARM	Sparatan3
Keccak	597.69	0.05	597.35	0.08	598.16	0.14
	Total = 597.74		**Total = 597.42**		**Total = 598.30**	
Blake2s	20.00	0.20	19.76	0.31	32.94	0.53
	Total = 20.20		**Total = 20.07**		**Total = 33.47**	

We notice, from Table 3, that the CPU processing contribution accounts for a large part of the total time for both Keccak and Blake2s. In addition, Blake2s offers a reduced message generating time ranging from 20 to 34 microseconds, which is 30 times faster than Keccak.

As a result, the use of the blake2s algorithm requires a maximum total hash time of 33.47 microseconds. Although the latency of a CAN message in automotive systems is typically around a few milliseconds, a few tens of microseconds is readily acceptable, as it would not have a significant effect on the CAN communication latency.

7. CONCLUSION

In this paper, an evaluation has been applied to the SHA-3 contenders' algorithms through an ARM Cortex A9 processor. A relevant comparison has also been performed involving an FPGA implementation to determine the fastest platform. From the standpoint of an ARM evaluation, the comparison shows a substantial win for Blake2s algorithm whereas Keccak offers excellent performances on FPGA circuits.

However, it is demonstrated that the CPU-based platform's impact on processing time is far more important than that of FPGA-based circuits. The comparison between Keccak and Blake2s shows that the latter is more likely to suit perfectly the targeted performances of the planned security system.

The use of cryptographic algorithms in automotive CAN bus systems can introduce limitations of computing speed related especially to the time response when dealing with real-time demands. However, considerable room for improvement with respect to security and data protection can be achieved thanks to the possibility of including a monitoring node along with the application of the justifiably chosen Blake2s that can contribute to a greater effectiveness.

This process is meant to be part of an authentication system within the CAN bus aiming to overcome the vulnerability problems of the network. As a matter of fact, implementing a prototype and running hardware simulations according to the guidelines of the enhanced network will be the follow-up task with respect to the development of the ongoing process.

8. REFERENCES:

[1] NIST, SHA-3 Competition (2007-2012), https://csrc.nist.rip/groups/ST/hash/sha-3/ (accessed: 2017)

[2] R. Dahal, J. Bhatta, T. Dhamala, "Performance Analysis of Sha-2 and Sha-3 Finalists", International Journal on Cryptography and Information Security, Vol. 3, No. 3, 2013, pp. 1-10.

[3] R. Sobti, 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, 2015, pp. 367-384.

[4] R. Sobti, 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, 2018, pp. 63-73.

[5] J. Sugier, "Improving FPGA implementations of BLAKE and BLAKE2 algorithms with memory resources", Proceedings of the 12th International Conference on Dependability and Complex Systems, Brunów, Poland, 2-6 July 2017, pp. 394-406.

[6] J. Sugier, "Low cost FPGA devices in high speed implementations of KECCAK-f hash algorithm", Proceedings of the 9th International on Dependability and Complex Systems, Brunów, Poland, 30 June - 4 July 2014, pp. 433-441.

[7] R. Bosch, "Can specification version 2.0", Postfach, Stuttgart, Germany, Technical Report Bosch, 1991.

[8] O. Avatefipour, A. Hafeez, M. Tayyab, H. Malik, "Linking received packet to the transmitter through physical-fingerprinting of controller area network", Proceedings of the IEEE Workshop on Information Forensics and Security, Rennes, France, 4-7 December 2017, pp. 1-6.

[9] P. Mundhenk, A. Paverd, A. Mrowca, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, S. Chakraborty, "Security in automotive networks: Lightweight authentication and authorization", ACM Transactions on Design Automation of Electronic Systems, Vol. 22, No. 2, 2017, pp. 1-27.

[10] J. Van Bulck, J. T. Mühlberg, F. Piessens, "VulCAN: Efficient component authentication and software isolation for automotive control networks", Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, USA, 4-8 December 2017, pp. 225–237.

[11] H. Zhang, X. Meng, X. Zhang, Z. Liu, "CANsec: A Practical in-Vehicle Controller Area Network Security Evaluation Tool", Sensors, Vol. 20, No. 17, 2020, p. 4900.

[12] A. Zniti, N. El Ouazzani, "Implementation of a bluetooth attack on controller area network", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 21, No. 1, 2021, pp. 321-327.

[13] Y. Yang, Z. Duan, M. Tehranipoor, "Identify a spoofing attack on an in-vehicle can bus based on the deep features of an ecu fingerprint signal", Smart Cities, Vol. 3, No. 1, 2020, pp. 17-30.

[14] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions", NIST, Gaithersburg, MD, USA, Technical Report NIST FIPS-202, 2015.

[15] R. Rivest, "The MD5 message-digest algorithm", IETF Network Working Group, MA, USA, Technical Report RFC 1321, 1992.

[16] G. Bertoni, J. Daemon, M. Peeters, G. V. Assche, "The Keeccak sha-3 submission", NIST SHA-3 Competition (Round 3), Gaithersburg, MD, USA, Technical Report 03, 2011.

[17] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker, "The skein hash function family", NIST SHA-3 Competition (Round 3), Gaithersburg, MD, USA, Technical Report 1.3, 2010.

[18] J. P. Aumasson, W. Meier, R. C. Phan, L. Henzen. "The Hash Function BLAKE", 1st Edition, Springer Berlin Heidelberg, 2014.

[19] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. chberger, M. Schlffer, S. S. Thomsen, "SHA-3 proposal grostel", NIST SHA-3 Competition (Round 3). Gaithersburg, MD, USA, Technical Report 2.0.1, 2008.

[20] H. J. Wu, "The hash function jh", NIST SHA-3 Competition (Round 3), Gaithersburg, MD, USA, Technical Report 42, 2011.

[21] NIST: Third-round report of the SHA-3 cryptographic hash algorithm competition, http://www.nist.gov/hash-competition (accessed: 2012)

[22] J. P. Aumasson, S. Neves, Z. W. O'Hearn, C. Winnerlein, "BLAKE2: Simpler, smaller, fast as MD5", Proceedings of the 11th International Conference on Applied Cryptography and Network Security, Banff, AB, Canada, 25-28 June 2013, pp. 119-135.

[23] S. Neves, J. O'Connor, J.P. Aumasson, Z. Wilcox-O'Hearn, BLAKE3: One function, fast everywhere, https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf (accessed: 2020)

[24] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, R.V. Keer, B. Viguier, "KangarooTwelve: Fast hashing based on Keccak-p", Proceedings of the 16th International Conference on Applied Cryptography and Network Security, Belgium, 2-4 July 2018, pp. 400-418.

[25] L. Pan, X. Zheng, H. Chen, T. Luan, H. Bootwala, L. Batten, "Cyber security attacks to modern vehicular systems", Journal of Information Security and Applications, Vol. 36, 2017, pp. 90-100.