Noname manuscript No.
(will be inserted by the editor)

Novel One Round Message Authentication Scheme for Constrained IoT Devices

Hassan N. Noura, Ola Salman · Raphaël Couturier · Ali Chehab

Received: date / Accepted: date

Abstract Security and privacy concerns have emerged as critical challenges in the Internet-of-Things (IoT) era. These issues need to be carefully addressed due to the sensitive data within IoT systems. However, some IoT devices have various limitations in terms of energy, memory capacity, and computational resources, which makes them extremely vulnerable to security attacks. Data integrity with source authentication are essential security services for protecting IoT data value and utility. Existing Message Authentication Algorithms (MAAs), which are either based on block ciphers or keyed hash functions, require multiple rounds and complex operations, which leads to unacceptable overhead for resource-limited devices and delay-sensitive applications. Moreover, the high number of IoT connected devices generates a huge amount of data, which challenges even the capacity of powerful network devices to handle the security of such Big Data. As such, the protection of such amounts of generated data calls for lightweight security solutions. In this paper, we propose a lightweight MAA that provides data integrity and source authentication. The proposed solution is based on a dynamic key structure with a single round and simple operations. The used cryptographic primitives (substitution and permutation tables) are dynamic and get updated for each new input message by using specific update primitives. The dynamic structure of the proposed MAA allows for decreasing the required number of rounds to just one, while maintaining a high degree of security. The security tests results show that the proposed keyed hash functions 1) achieve the desired cryptographic properties, 2) are immune against existing attacks and 3) require low overhead in terms of computational and storage resources.

Keywords Lightweight Message Authentication Algorithm; Dynamic Key-Dependent Cryptography; Security and Performance Analysis.

Corresponding author, email: oms15@mail.aub.edu

Hassan N. Noura and Raphaël Couturier
Univ. Bourgogne Franche-Comté (UBFC), FEMTO-ST Institute, France

Ola Salman and Ali Chehab
American University of Beirut, Electrical and Computer Engineering Department, Beirut 1107 2020, Lebanon

1 Introduction

The last decade witnessed the emergence of new information and communication technologies such as the Internet of Things (IoT), in addition to new wireless technologies for public and private applications. This resulted in an explosive growth in the network traffic, which is expected to exceed 175 Zettabytes of generated data worldwide by 2025 (Patrizio, 2018). This will enable innovative applications (Arshad et al., 2018; Nour et al., 2019a), however, data security emerges as one of the most critical challenges facing the data proliferation in future networks (Nour et al., 2019b; Yaacoub et al., 2020a,b).

Data integrity is an essential security aspect that should be ensured given the pervasive access to data (Stallings, 2017). A simple modification of data may have drastic effects, including tangible and intangible losses. Data integrity and source authentication represent the first line of defense for any application. Even if confidentiality and availability measures are in place, integrity attacks can seriously impact the utility of data (Noura et al., 2019c). For example, ransomware attacks on medical systems can modify or corrupt medical records,

causing catastrophic damages.

The existing integrity solutions are based on multi-round and multi-operation functions, which exhibit a large overhead in terms of computations and storage (Noura et al., 2018a, 2019b). As such, and given the fact that many IoT device types are resource-constrained, there is a pressing need for the design of lightweight security solutions. In this paper, we propose a lightweight data integrity and source authentication solution for IoT devices. In contrast to existing solutions, the proposed one is dynamic and key-dependent, which allows it to be effective as it utilizes simple operations with just a single round. The performance and security analysis show that the proposed solution strikes a good balance between the high security level and the low computational overhead.

1.1 Related work

Preserving data security in IoT applications has attracted the attention of many researchers in the last few years (Nour et al., 2019b). Having ubiquitous access to published data calls for robust security measures to prevent the misuse or disclosure of such data. To prevent illegal access to data, different access control solutions have been proposed, most of them are based on the attribute-based access control scheme (Misra et al., 2017; Li et al., 2016; Xue et al., 2018; Fotiou and Polyzos, 2016).

On the other hand, several cryptographic solutions have been proposed to ensure data confidentiality. Bilal et al. in (Bilal and Pack, 2019) proposed a symmetric message authentication for distributed IoT data. Recently, a dynamic key-dependent block cipher was presented in (Noura et al., 2019d), to ensure data confidentiality. The proposed solution requires only one round and simple operations, which can be applied as a lightweight solution for ensuring data confidentiality.

However, in this paper, we aim at ensuring data integrity with source authentication for IoT devices, given the key importance of this aspect in preventing data utility loss. Data integrity can be provided using unkeyed hash functions such as the Secure Hash Function (SHA) with its variations SHA-2 (Handschuh), SHA-3 (Bertoni et al., 2011), BLAKE (Aumasson et al., 2010), Grostl (Gauravaram et al., 2009), Skein (Ferguson et al., 2009), and Keccak (SHA3) (Bertoni et al., 2011). On the other hand, different solutions have been presented for source authentication by using digital signature or symmetric MAA. The MAA can be keyed hash function-based such as Hash-based Message Authentication Code (HMAC) (Krawczyk et al., 1997), or block cipher-based such as the Cipher-based Message Authentication Code (CMAC) (Song et al.), and Galois Message Authentication Code (GMAC) (McGrew and Viega, 2006). In general, the symmetric MAA consists of several iterations of a compression function, which consists also of multiple operations to provide the desired confusion and diffusion properties. In this context, HMAC is based on the SHA variants, which require a large number of rounds (SHA-1, and SHA-512 with 80 rounds and SHA-3 with 24 rounds) (Stallings, 2017). CMAC uses the Advanced Encryption Standard (AES) (Miller et al., 2009) with Cipher Block Chaining (CBC). In this case, the used key for AES can be of size 128, 192 or 256 bits with 10, 12, or 14 rounds, respectively. Thus, a lightweight MAA is required to ensure data integrity and source authentication of the huge amount of data in future networks with low delay and computational overhead (Beaulieu et al., 2015).

In this context, several solutions have been presented in the literature for reducing the number of rounds of hash functions. To do so, some solutions consider the "Chaos" theory relying on a non-linear system, which is highly sensitive to the initial conditions and parameters (Teh et al., 2015)-(Kanso and Ghebleh, 2015). However, these solutions are not practical because they exhibit a high computational complexity (Noura et al., 2019a). Then, integer-based representations were proposed to overcome these challenges (Amigó et al., 2007; Masuda et al., 2006; Noura, 2012). However, these solutions also suffer from high delay and computational complexity. Alternatively, new lightweight cipher solutions have been proposed to reduce the required number of operations and rounds. Cipher schemes based on two rounds were proposed in (Noura et al., 2017; Noura and Courousse, 2016; Noura et al., 2018b) and others based on one round were proposed in (Noura et al., 2019d, 2018a). In this context, relying on the dynamic key structure maintains the security of the lightweight cipher solutions (Noura et al., 2017)-(Noura et al., 2018b). However, none of the previous works considered a one round message authentication algorithm (Keyed hash function) based on a dynamic key.

1.2 Contributions

The contributions of this paper can be summarized as follows:

- Proposing a lightweight dynamic key-dependent MAA with two variants of the compression function. In

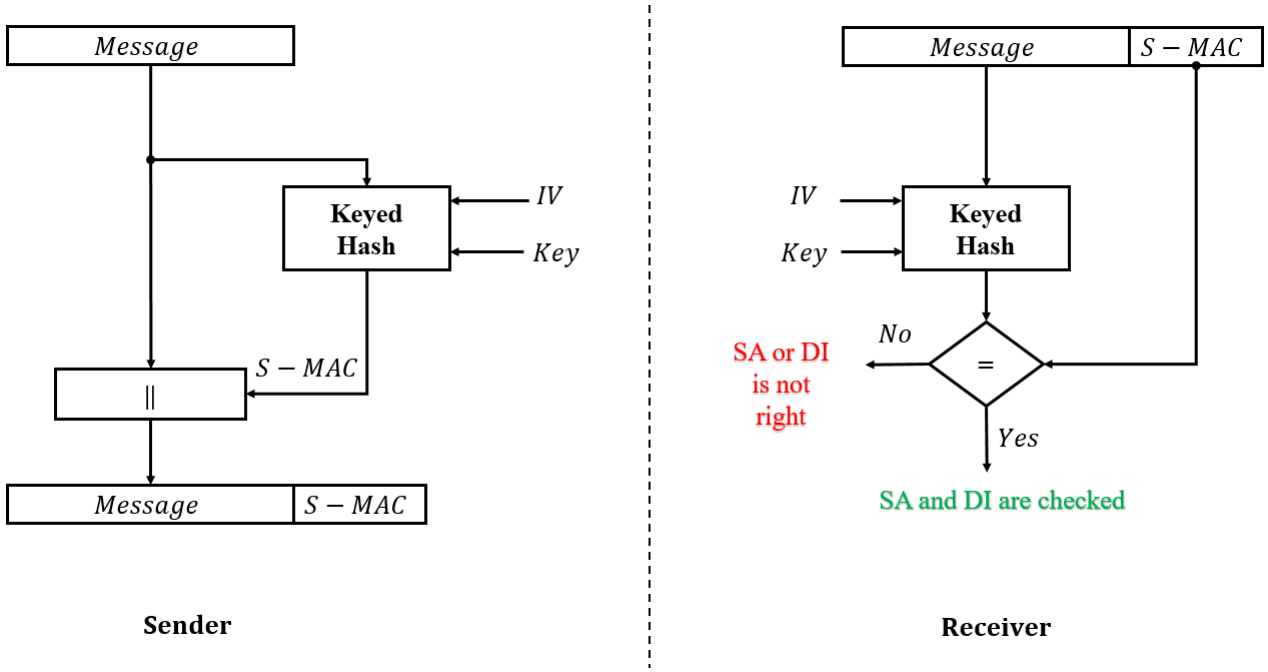


Fig. 1: Message authentication algorithm: signature and verification

the first variant, only one data block is processed at a time, while two data blocks are processed in the second variant.
– The proposed MAA requires a single round and uses simple operations such as substitution, permutation and "exclusive OR" operations, to minimize the computational complexity and to simplify the implementation.
– Achieving a high level of security including the essential cryptographic properties of randomness, uniformity, and sensitivity.

1.3 Organization

The rest of this paper is organized as follows. In Section 2, a background is given about MAA and IoT. In Section 3, we present the system and threat models. Section 4 discusses the proposed message authentication scheme, including the cryptographic primitives derivation and the proposed hash function variants. Section 5 analyzes the strength of the proposed variants with respect to several cryptographic properties. In Section 6, we include a crytanalysis discussion of the proposed solution. Results on computational complexity and performance, of the proposed hash function, are presented in Section 7. Finally, Section 8 concludes this work.

2 Background

In this section, we provide a background of the different message authentication schemes including symmetric and asymmetric ones, in addition to the IoT networking paradigm.

2.1 Message Authentication Algorithm

MAA is an important security service that allows the verification of data integrity and source authenticity. To authenticate the origin of data and verify its integrity, symmetric message authentication codes (MACs) or asymmetric digital signatures can be used. Both types of solutions may use hash functions, which could be either keyed or un-keyed. The un-keyed hash functions are called Modification Detection Codes (MDCs), and they target only data integrity. The values returned by an un-keyed hash function are commonly referred to as hash code, and they play a fundamental role in integrity protection and digital signatures.

Mathematically, a cryptographic hash function (Menezes et al., 1996) is a non-linear and non-invertible compression function h: $\{0,1\}^* \rightarrow \{0,1\}^{Tb}$, where Tb is a positive integer representing the size in bits of an input block. Technically, hash functions divide the input data, D, having an arbitrary length $|D|$, into nb blocks,

each with a size of Tb bits. An MAA should satisfy the following properties to ensure its safe implementation:

- Preventing exhaustive search attacks, where the size of the secret key (K) has to be sufficiently large ($\geq$ 128 bits).
- Preventing birthday attacks, where the size of the MAC has to be sufficiently large ($\geq$ 128 bits).
- The MAC values should exhibit a high level of randomness and a uniform distribution to prevent statistical attacks.
- It should satisfy the avalanche effect, for both the message and the key, to prevent key-related attacks and chosen/known plain-text attacks.

Generally, a symmetric MAA takes as input a secret key (K), an Initial Vector (IV) (e.g. counter, identity, etc.), and the message itself (see Fig. 1). The output is a Tb-bit MAC that will be appended to the corresponding message. For message authentication verification, the receiver computes the expected message authentication code ($R-MAC$) of the received message. If there is a match with the received message authentication code ($S-MAC$), the message is authenticated.

The existing message authentication algorithms are based on the structure of Merkle-Dangard (Damgård, 1990; Merkle, 1989), which is illustrated in Fig. 2. A special case of MAA is when it is un-keyed and all cryptographic primitives of the compression function f are also independent of the key. In this case, only data integrity can be verified.

2.2 Internet of Things Network

The IoT architecture consists mainly of four layers: the device layer, the gateway layer, the network layer, and the application layer. The device layer consists of the set of IoT devices organized into sub-networks. Each sub-network is connected to a gateway. The gateway layer consists of the ensemble of IoT gateways interconnected through the core network. The core network has the role to manage the inter-networking and communication between the different IoT sub-networks. Finally, the application layer has the role of managing the data pertaining to different IoT applications. It should be mentioned here that each layer has a security management role. Thus, a fifth vertical layer can be added to manage security and privacy at the different IoT layers.

3 Problem Statement

In this section, the system and threat models are presented.

3.1 System Model

The network model consists of n IoT nodes, each having a certain energy and computational capacity. These nodes are connected to the IoT network through a gateway (see Fig. 3). The proposed MAA is applied at the IoT devices level, and the integrity and authentication check is performed at the application server level.

The main steps of the proposed solution, at the IoT device side, are:

1. For each data message, the IoT device generates a dynamic key, DK, based on a master key MK and a $Nonce$, which is a unique dynamic key for each data message.
2. To ensure the data authentication and integrity, the IoT device uses the dynamic key to apply the proposed lightweight MAA to produce the corresponding MAC, which is appended at the end of the message.

The main steps of the proposed solution, at the application server side, are:

1. When the application server receives data, it checks its integrity and authenticates its source, before storing it.
2. To do so, it acquires the corresponding dynamic key, DK.
3. The key exchange can be based on symmetric (key distribution center) or asymmetric schemes.
   In the symmetric case, a pre-shared master key (PSK) is shared between the communicating parties and trust server through a secure channel. This key is used to exchange the dynamic key. In the asymmetric case, a Public Key Infrastructure (PKI) is needed to exchange DK.

3.2 Threat Model

IoT networks face many security challenges and they are prone to different security attacks (AbdAllah et al., 2015). In this work, we consider three types of data integrity attacks or errors:

- A message could be modified due to a channel error. The proposed scheme prevents erroneous messages from being stored; the node makes another request of the original data.

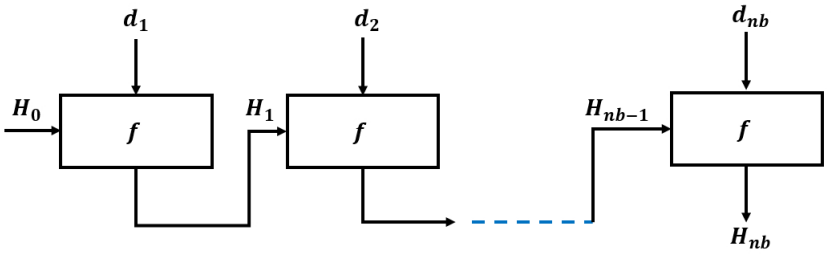


Fig. 2: Structure of the Merkle-Damgard algorithm used traditionally to construct a message authentication algorithm, where H_0 can represent a secret key and the message is divided into nb blocks $\{d_1, d_2, \ldots, d_{nb}\}$ that have a fixed size in addition to f, which represents the compression function

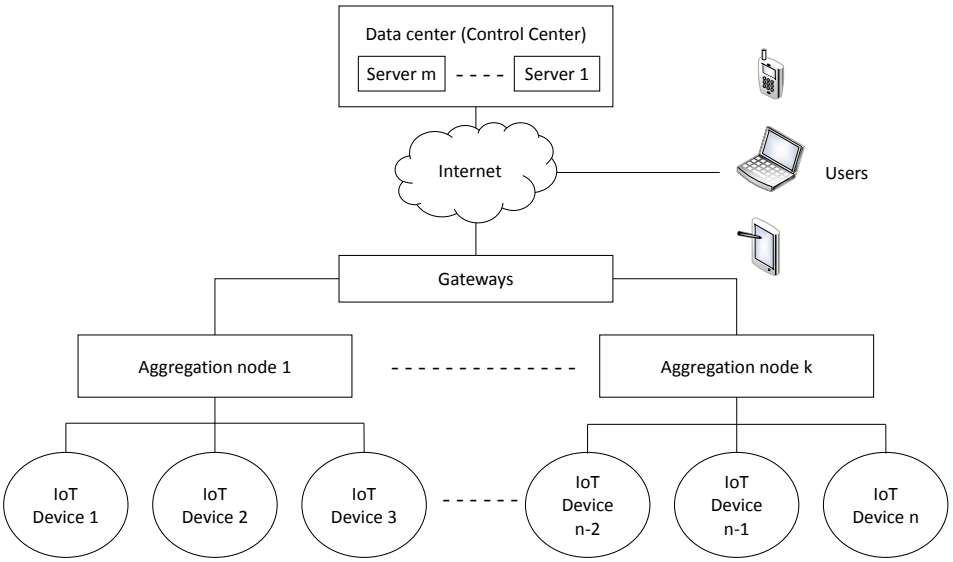


Fig. 3: An example of IoT network model: a set of n IoT devices, the corresponding gateways, and the application servers (or data center)

– A malicious attacker could modify the encrypted data without being able to generate the correct MAC value of the modified message. Such an attack is easily detectable, however, it could be used to launch a DDoS attack, compromising the availability of data. In this case, the proposed scheme can only detect the malicious data manipulation.
– A more dangerous attack is when a malicious attacker tries to infer, by statistical analysis, the secret key to modify the data and re-generate the new MAC value. In this case, and since the proposed scheme is based on the dynamic key approach, it guards against key-related statistical attacks and well-known cryptanalysis techniques.

4 Proposed Message Authentication Solution

In this section, the proposed message authentication scheme is detailed, including the derivation of the cryptographic primitives and the two proposed variants of the compression function. Note that all the used notations are listed in Table 1.

4.1 Derivation of Cryptographic Primitives

As shown in Fig. 4, the dynamic key generation is linked to a master key, MK, and the message timestamp, TS. The master key can be renewed at a fixed interval, which depends on the sensitivity level of the data.

Table 1: Summary of notations

Notation	Definition
MK	Master key
DK	Dynamic Key
MD	Meta Data
D	Data input
M	Data input length in bits
TS	Message timestamp
$Nonce$	Random number
Tb	Number of bits in a data block
TB	Number of bytes in a data block, with $TB = Tb/8$
nb	Number of data blocks in D, where $nb = M/Tb$
n	Number of IoT nodes
Sk_1 and Sk_2	Substitution sub-keys
ST_1 and ST_2	Substitution tables
PK	Permutation Key
π	Permutation table
RKG	Round key generation sub-key
$RK1$ & $RK2$	Two sets of round keys
m	Number of round keys
RKS	Round key selection
RST	Round key selection table
π	Round key selection permutation table
KPS	Block selection sub-key
τ	Threshold sum of sizes of δ small-sizes messages
δ	Number of small-sized messages
l	Maximum size of small-sized messages

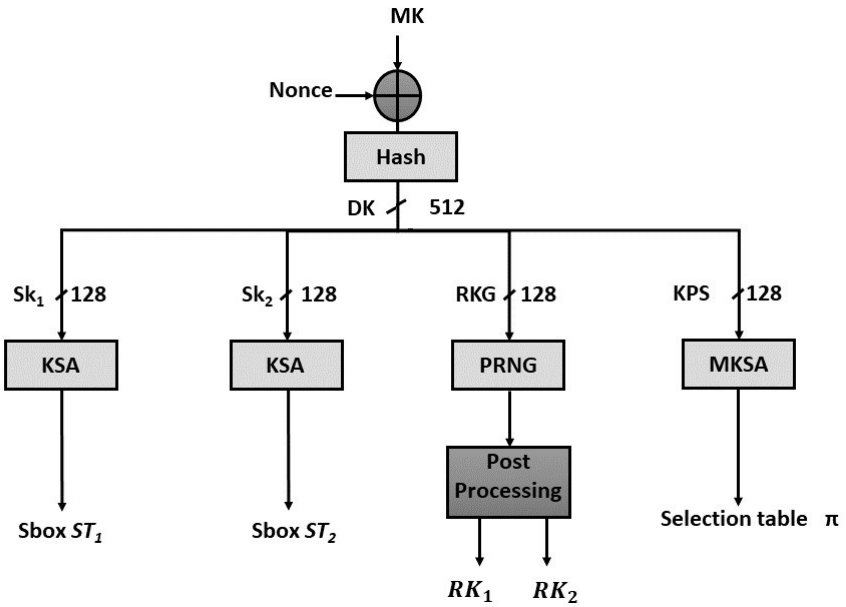


Fig. 4: Proposed dynamic key generation (DK) and construction of cryptographic primitives (substitution and selection tables in addition to round keys)

For each new data, the provider generates a $Nonce$ by first performing an xor operation on the metadata MD and the timestamp TS, and then, hashing the result. Next, a new dynamic key, of size 64 bytes, is generated by performing the xor operation between MK and the $Nonce$. Next, the dynamic key is divided into five sub-keys: $DK = \{SK_1,\ SK_2,\ PK,\ RKG,\ RKS\}$; Sk_1 and Sk_2 are of size 64 bits (8 bytes) each, and the

remaining keys have each a size of 128 bits (16 bytes). In the following, we describe the role of each sub-key.

- **Substitution Sub-Keys Sk_1 and Sk_2**: Sk_1 consists of the most significant 8 bytes, and Sk_2 consists of the next most significant 8 bytes of DK. These sub-keys are used for the construction of two substitution tables, ST_1 and ST_2, based on the RC4 Key Setup Algorithm (KSA) (Noura et al., 2018a, 2019d). The elements of ST_1 and ST_2 have values in the range [0-255], given that the substitution is performed at the byte level.
- **Permutation Sub-Key** PK: it consists of the next most significant 8 bytes of DK, and it is used to construct a permutation table π of length TB. The modified RC4 key setup algorithm described in (Noura et al., 2018a) is used to produce the permutation table. The elements of π are in the range [1 - TB], where $TB = \frac{Tb}{8}$ denotes the number of bytes in a block. The permutation process is also applied at the byte level.
- **Round Key Generation Sub-Key** RKG: it consists of the next most significant 16 bytes of DK, and it is used as a seed for any Pseudo Random Number Generation (PRNG) function to produce one pseudo random block (RK_1) for the first version of the compression function, and two pseudo-random blocks (RK_1 and RK_2) for the second version, for each processed block.
- **Block Selection Sub-key** KPS: finally, this sub-key consists of the least significant 16 bytes of DK. The modified RC4 (Noura et al., 2018a) is applied with KPS to generate a permutation table π with length nb elements. This acts as a selection table to randomize the processing order of the input blocks, in contrast to the existing techniques, which adopt sequential processing.

This derivation scheme of the cryptographic primitives guarantees that a one-bit difference in MK will result into a completely different dynamic key DK, which in turn leads to different cryptographic primitives.

4.1.1 Adaptation of the Key Derivation Function for Low Data Rate Applications

Although the proposed scheme is designed primarily for high a volume of data, it can also be adapted for small messages (low data rate applications), as the scheme is flexible and easily configurable. For small-message use, the update of the dynamic key and cryptographic primitives is not performed for each input message and can be realized after a specified data threshold length, which is called δ messages (It is the size of a set of small messages). The dynamic key is updated for each δ, and similarly for the cryptographic primitives. In fact, a low value of δ means a high level of security but this requires more overhead in terms of delay and resources. The technique to update the substitution cryptographic primitives for each message is based on updating ST_1 in function of ST_2, and ST_2 in function of ST_1. The update process of the substitution primitives, after each message encryption, is as follows:

$$\begin{aligned} Temp &= ST_1 \\ ST_1 &= ST_1(S_2 >> mod(it,\ 256)) \\ ST_2 &= ST_2(Temp) \end{aligned} \tag{1}$$

where it is incremented by 1 at each update, representing the update counter, and $ST_2 >> v$ consists of a circular shift of the substitution table S_2, by v elements.

4.2 Proposed One Round Message Authentication

The MAA algorithm consists of two variations of a compression function, and it requires a single round. The first variant consists of authenticating only a single block per iteration, while in the second one, two blocks are processed simultaneously. In both variants, the input blocks are selected according to the selection permutation table π to overcome the issues associated with sequential block processing.

The proposed MAA relies on the Merkle Damgrad (MD) structure, shown in Fig. 2, with the difference of dynamic block selection. After generating the cryptographic primitives, the input data D is padded, if necessary, to have a length divisible by the size of the data block, Tb; the block size could be set to 64, 128, 256, 512, or 1024 bits. Then, D is divided into nb blocks ($d_1,\ d_2,\ \ldots,\ d_{nb}$), where $nb = \frac{|D|}{Tb}$. The choice of Tb is related to the required security level and the available device resources. Afterwards, each data block $d_i,\ 1 \leq i \leq nb$, goes through the compression function, f, after being mixed with the previous block output H_{i-1}. Finally, the MAC value consists of the last compressed block (H_{nb}), and it can be calculated according to the following equation:

$$H_i = f(H_{i-1} \oplus d_i,) \qquad i = 1, 2, 3, \ldots, nb \tag{2}$$

where $H_0 = IV1$ for the first compression function, and $H_0 = IV1||IV2$ for the second one.

The proposed solution is applied at the byte level and it can be adapted to the word level. Also, it inherently satisfies the message and key avalanche properties

since for each new input data, a new dynamic key and new cryptographic primitives are generated and used. The same is true for any change in the dynamic key. Moreover, the proposed algorithm can be applied to any data type such as images, video, text, etc. In the following, we describe the proposed MAA blocks, namely BlocksSelection, and RoundFunction(RF).

4.2.1 Blocks Selection

The round function f is applied on data blocks that are selected based on the generated π. This selection results into a pseudo-random sequence of the blocks, avoiding the sequential relationship among the authenticated blocks and thus, complicating the cryptanalysis. For the first variant, f considers one input block ($d_{\pi(i)}$) at each iteration, where $i = 1, 2, \ldots, nb$. However, for the second variant, two input blocks ($d_{\pi(i+\frac{nb}{2})}$ and $d_{\pi(i)}$) are processed per iteration, where $i = 1, 2, \ldots, \frac{nb}{2}$ and $d_{\pi(i)}$ represents the $\pi(i)^{th}$ input block.

4.2.2 Round Function Variants

This phase represents a substitution operation, where a round key is dynamically chosen based on the round key selection table from a set of generated round keys. In the following, we present the two variants of the compression function.

4.2.2.1 First Round Function Variant

It employs the permutation table (π) in addition to the first substitution table (ST_1) and a round key ($RK1$), which is produced for each processed block. $RK1_i$ is used for i^{th} input block.

At each iteration of this round function (see Fig. 5), a data input block ($d_{\pi(i)}$) is chosen to be compressed according to the following equation (Eq. 3):

$$IV = ST_2((ST_1(d_{\pi(i)} \oplus RK1_i) \oplus IV) \tag{3}$$

As shown in Eq. 3, for each data block, IV is updated by: first, xoring the i^{th} data input block $d_{\pi(i)}$ chosen based on the permutation table π with the i^{th} round key of $RK1$. Then, the result is substituted using ST_1. Next, the substituted block is xored with the current IV. Finally, the output is substituted again by using the second substitution table ST_2 to form the updated IV. The last updated IV output of the last iterated block nb represents the MAC value. The pseudo code of the compression function, *compression_Function*, is presented in Algorithm 1.

4.2.2.2 Second Round Function variant

The second compression round function variant uses both substitution tables ST_1 and ST_2, and two sets of round keys $RK1$ and $RK2$, as described in Algorithm 2 and shown in Fig. 6. At each iteration, two data blocks ($d_{\pi(i+\frac{nb}{2})}$ and $d_{\pi(i)}$) are processed in parallel to update two IVs, $IV1$ and $IV2$, as indicated in Eq. 4.

$$\begin{aligned} IV2 &= ST_2(d_{\pi(i)} \oplus IV1 \oplus ST_1(d_{\pi(i+\frac{nb}{2})} \oplus RK1_{(i)})) \\ IV1 &= ST_1(d_{\pi(i+\frac{nb}{2})} \oplus IV2 \oplus ST_2(d_{\pi(i)} \oplus RK2_{(i)}))) \end{aligned} \tag{4}$$

To update $IV2$, first, $d_{\pi(i+\frac{nb}{2})}$ is xored with the i^{th} round key of $RK1$. The result is substituted using ST_1. Then, the obtained result is xored with $\pi(i)^{th}$ message plain block $d_{\pi(i)}$ and current $IV1$. Finally, the output is substituted using ST_1, which represents the updated $IV2$.

$IV1$ is updated by mixing (xor) $d_{\pi(i)}$ with the $(i)^{th}$ round key of $RK1$ ($RK1_{(i)}$). The result is substituted by using the substitution table ST_2. Then the substituted output is xored with the $\pi((i + \frac{nb}{2})^{th}$ message plain block $d_{\pi(i+\frac{nb}{2})}$ and current $IV2$. The obtained output is substituted by using the substitution table ST_1, which represents the updated $IV1$.

The last updated $IV1$ and $IV2$ outputs of the last iteration are mixed ($IV1 \oplus IV2$) to produce the MAC value.

4.2.2.3 Comparison Between Compression Function Variants

Both compression functions are dynamic and key-based, employing chaining, pseudo-random selection, and mixing operations, as indicated in Table 2. Even though one round operation is performed, the dynamic substitution and permutation of the different blocks ensures the randomness, uniformity and independence of the obtained MAC for the same data, which results in a high security level. However, the two variants differ in the number of operations, having two additional exclusive-or operations for the second one, which increases slightly the required execution time (5% to 10%). However, this slight cost is compensated by enhanced randomness and robustness.

5 Security Analysis

To analyze the security of the proposed message authentication variants, several security tests were per-

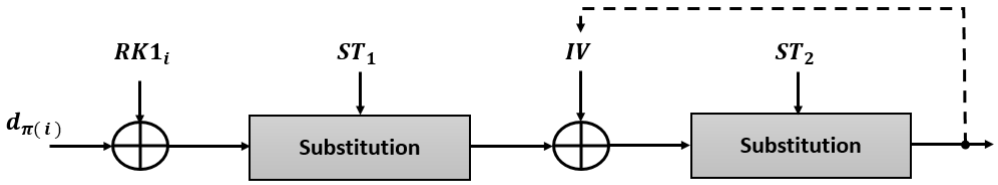


Fig. 5: Proposed first compression function variant (one block is processed in each iteration)

Algorithm 1 First Compression Function Variant

procedure FIRST_COMPRESSION_FUNCTION (D, N, RK_1, ST_1, IV)
  $D \leftarrow padding(D, N)$
  $d_1, d_2, \ldots, d_{nb} \leftarrow DividetoBlocks(D, N)$
  for $i = 1 \rightarrow nb$ **do**
    $IV = ST_2(IV \oplus ST_1(d[\pi[i]] \oplus RK1[i]))$
  end for
  $MAC \leftarrow IV$
  return MAC
end procedure

Algorithm 2 Second Compression Round Function Variant

procedure SECOND_COMPRESSION_FUNCTION (D, N, RK_1, RK_2 ST_1, ST_2, $IV1$, $IV2$)
  $D \leftarrow padding(D, N)$
  $d_1, d_2, \ldots, d_{nb} \leftarrow DividetoBlocks(D, N)$
  for $i = 1$ to $\frac{nb}{2}$ **do**
    $IV2 = ST_2(ST_1(d[\pi[i]] \oplus RK1[i]) \oplus IV1 \oplus d[\pi[i + \frac{nb}{2}]])$
    $IV1 = ST_1(ST_2(d[\pi[i + \frac{nb}{2}]] \oplus RK2[i]) \oplus IV2 \oplus d[\pi[i]])$
  end for
  $MAC \leftarrow IV1 \oplus IV2$
  return MAC
end procedure

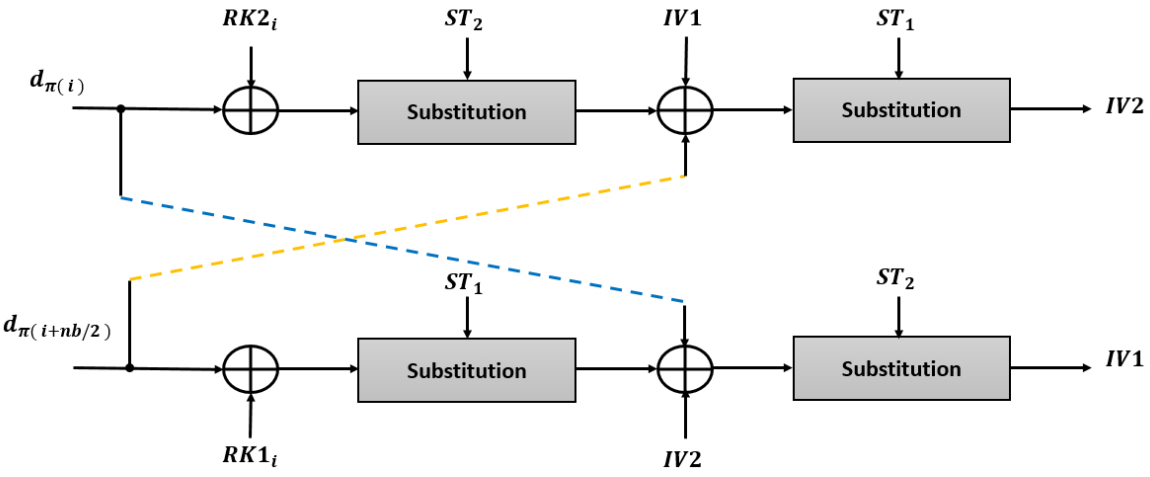


Fig. 6: Proposed second compression function variant (two blocks are processed in each iteration)

formed including randomness, uniformity and sensitivity. Also, a comparison is conducted between the proposed MAA and standard ones like CMAC and HMAC.

Table 2: Comparison between the proposed compression function variants

Metric	**First Variant**	**Second Variant**
Delay	Lower one	More delay
Randomness	+	++
Robustness (makes the relation among input blocks more complicated)	+	++
Parallel processing	Yes	Yes

5.1 Randomness and Uniformity Tests

The randomnesses and uniformity of the obtained MAC values are measured in relation to the message and the used cryptographic primitives. In this test, the MAC values are evaluated by simulating the proposed variants on 1,000 input random messages.

We consider two original messages, one consisting of random characters expressed by their ASCII codes, and a second one consisting of all zeros. The distribution of the messages is shown in Fig. 7-a and Fig. 8-a, respectively. We compute the MAC values, each of size 16 bytes, of these messages using the two proposed variants. Then, for each MAC value, we consider its hexadecimal representation, which yields 32 hex digits. The distribution of their values are illustrated in Fig. 7-b and Fig. 8-b for the first variant, and Fig. 7-c and Fig. 8-c for the second variant. As it can be seen, the hex values of (0 to 15) are uniformly spread.

Note that these tests are repeated with 1,000 different dynamic keys. On the other hand, we conducted a test to compute the number of unique bytes within each MAC value. We generated 1,000 values while considering different dynamic keys and data contents. The MAA variants are flexible in terms of the Tb size, and for the simulations, we show the results of the randomness and uniformity test with two values of Tb, 128 and 256. Clearly, increasing Tb enhances the randomness and uniformity test results.

Table 3 and 4 show the results corresponding to the percentages of unique byte elements, within each MAC value, of the two variants and for $Tb = 128$ and 256, respectively. For $Tb = 128$, the results show that about $\approx 62.22\%$ of the MAC values have all of the 16 bytes unique, and about $\approx 30.75\%$ of the MAC values have 15 unique bytes out of the 16 bytes. For $Tb = 256$ (32 bytes), the results show that about $\approx 13.43\%$ of the MAC values have all of the 32 bytes unique, about $\approx 28.22\%$ of the MAC values have 31 unique bytes out of the 32 bytes, and about $\approx 30.20\%$ of the MAC values have 30 unique bytes. Similar results were obtained for the second variant. The above results confirm that the generated MAC values have uniform distributions with a high level of randomness.

5.2 Sensitivity Tests

Message and key sensitivity tests are performed to validate that different MAC values are obtained for a slight modification in the message or in the secret key.

5.2.1 Message Sensitivity (Avalanche Effect)

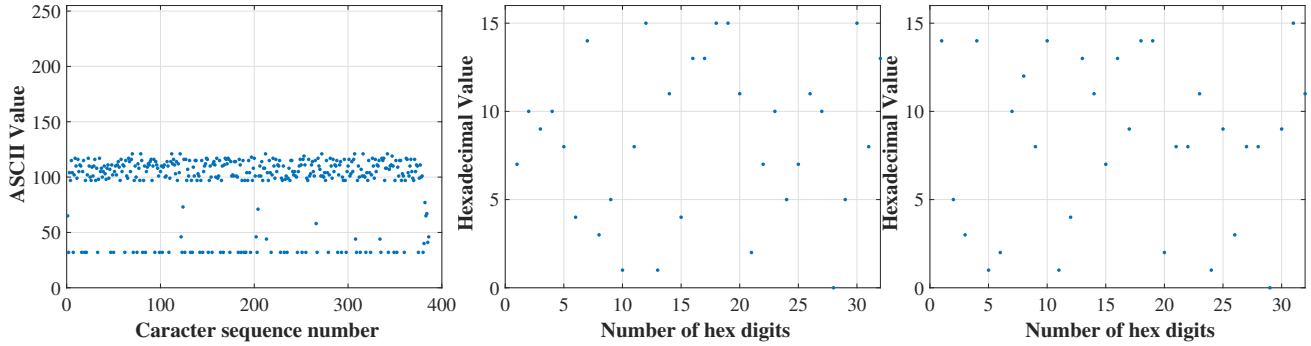
In this test, we consider two messages, which are identical except for a single bit difference, to be authenticated by the same IoT node. Given that the proposed algorithm generates different cryptographic primitives for each new message, the sensitivity of the MAC value is guaranteed based on the key avalanche effect. To confirm it, we run several tests under the following conditions:

- C1: A random original message;
- C2: Flip the Most Significant Bit (MSB) of first character (8 bits) of first block;
- C3: Flip the Least Significant Bit (LSB) of last character (8 bits) of last block;
- C4: Replace the full stop from the original paragraph with comma;
- C5: Add a blank space to the original paragraph.

The corresponding percentages of changed bits under these conditions are presented in Table 5 for the first variant. It is clear from the results that any minor change in the original data results in a very different MAC value with a difference around 50%. Statistical results were presented in Table 6-b) for the first variant. Similar results were obtained for the second variant.

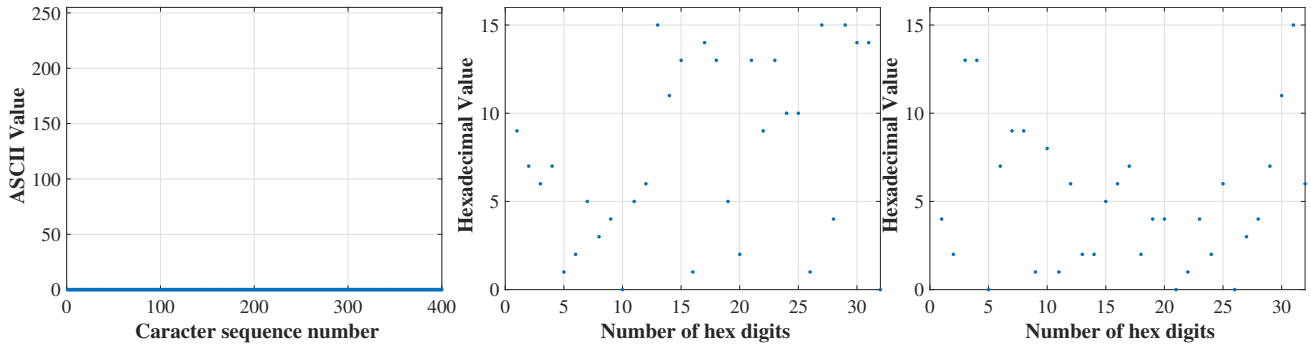
5.2.2 Key Sensitivity

This test aims at evaluating the change in the MAC values for a small change in the master key MK or the



(a) (b) (c)

Fig. 7: Spread of message and MAC value: a-distribution of the message in ASCII code; b-distribution of the MAC value in hexadecimal format (first variant); **and c-for the second variant**



(a) (b) (c)

Fig. 8: Spread of all Zeros message and MAC value: a-distribution of all Zeros message; b-distribution of the MAC value in hexadecimal format (first variant); **and c-for the second variant**

Table 3: Percent of the different number of ASCII characters (bytes) for n_{MAC}=10000 with $Tb = 128$

Percentage /Diff ASCII	16	15	14	13	12
First variant	62.22	30.75	6.42	0.59	0.02
Second variant	62.22	30.75	6.42	0.59	0.02

Table 4: Percent of the different number of ASCII characters for n_{MAC}=10000 with $Tb = 256$

Percentage/ Diff ASCII	32	31	30	29	28	27	26	25	24
First variant	13.43	28.22	30.2	17.85	7.38	2.28	0.56	0.05	0.03
Second variant	13.43	28.22	30.2	17.85	7.38	2.28	0.56	0.05	0.03

Table 5: Distribution of changed bit number under different conditions

Case	C1	C2	C3	C4	C5
C1	-	47.6563	49.2188	53.1250	48.4375
C2	47.6563	-	45.3125	57.0313	46.0938
C3	49.2188	45.3125	-	47.6563	52.3438
C4	53.1250	57.0313	47.6563	-	50.0000
C5	48.4375	46.0938	52.3438	50.0000	-

$Nonce$. In the proposed scheme, all of the cryptographic primitives, in addition to the dynamic key, are generated from MK and the $Nonce$. The desired behavior is that a one bit change in MK or the $Nonce$ should result in different set of primitives, and thus, different MAC values. To confirm it, we conducted sensitivity test of MK, for 1,000 random keys (MK), and we computed the Hamming distance between every pair of the generated MAC values, based on the following equation:

$$MKS_w = \frac{\sum_{k=1}^{T} MAA_{MK_w}(M) \oplus MAA_{MK'_w}(M)}{Tb} \times 100\% \quad (5)$$

where MKS_w is the obtained percentage of Hamming distance between the MAC value obtained by using the master keys MK_w and MK'_w; Tb is the number of bits in the MAC value, and $MAC_w = MAA_{MK_w}(M)$ and $MAC'_w = MAA_{MK'_w}(M)$ are the obtained MAC values using the master keys MK_w and MK'_w, respectively. We considered a difference between MK_w and MK'_w in the Least Significant Bit (LSB) of a randomly chosen byte. Similarly, the sensitivity of $Nonce$ is evaluated, given that the dynamic key depends on MK and the $Nonce$.

The key sensitivity test results are included in Fig. 9 for both variants and in Table 6-a) for the first variant. The obtained results are similar to the obtained ones with CMAC and HMAC. The plotted results of KS with respect to 1,000 random keys for both variants show that in the majority of cases, KS is approximately 50% and that KS follows a normal distribution. Similar results are obtained for the $Nonce$ sensitivity test for the second variant.

5.3 Collision Resistance

Collision resistance is another security requirement for any MAA. It aims at assessing the probability of having two distinct input data with same MAC value. Actually, hash functions are designed to resist collision, and to avoid having two distinct inputs with same hash result. This test was applied for 1,000 dynamic keys, and each time, a randomly-selected bit of a paragraph (in ASCII format) is modified. Then, we apply the proposed MAA variants and compare the MAC values of the initial and modified data: we compute the number of identical ASCII characters at the same positions, as shown in the following equation:

$$Diff = \sum_{i=1}^{TB} D\{MAC(i), MAC'(i)\}, \quad (6)$$

where $D(x, y) = 1$ if $x = y$, else $= 0$.

MAC and MAC' represents the MAC values of original and modified message, respectively; MAC(i) represents the i^{th} ASCII character of the MAC.

The results, shown in Table 7, show that only 3 characters are equal for $Tb = 128$ and 256, and 4 characters are equal for $Tb = 512$ and 1024. This indicates that there is a very small percentage of similar characters when considering different input data with the same key. As such, the proposed scheme is immune against collision and it can resist statistical attacks such as birthday attack, meet-in-the-middle and differential attacks (Wang and Yu, 2005).

6 Cryptanalysis

In this section, we present cryptanalysis against the proposed MAA by considering key- and MAC-related attacks.

6.1 Key Space Analysis

The secret-key space of the proposed authentication algorithm is a very large one, 2^{Tb}, with Tb =128, 160, 196, 256, 512, or 1024 bits. To resist brute force attacks, according to Schneier (Schneier, 2007), the key space should be larger than 2^{128}, which is the case of the proposed scheme.

6.2 Pseudo-collision Resistance

In a pseudo-collision attack, the attacker tries to modify the message and the associated MAC value, by relying on weaknesses in the compression function (Akhavan et al., 2013; Yang et al., 2009). However, the proposed MAA consists of consecutive substitutions of message blocks, organized in a pseudo random sequence, and based on dynamic key-related cryptographic primitives. Moreover, the compression function is non-linear and not commutative, which ensures the pseudo-collision resistance.

6.3 Resistance to Birthday Attack

Birthday attacks are traditional attacks that can target the MAA in the aim at finding two messages with same MAC values in less than $2^{\frac{N}{2}}$ trials (Tb is the size of MAC value) (Menezes et al., 1996). In the proposed method, the smallest MAC size is 128, which

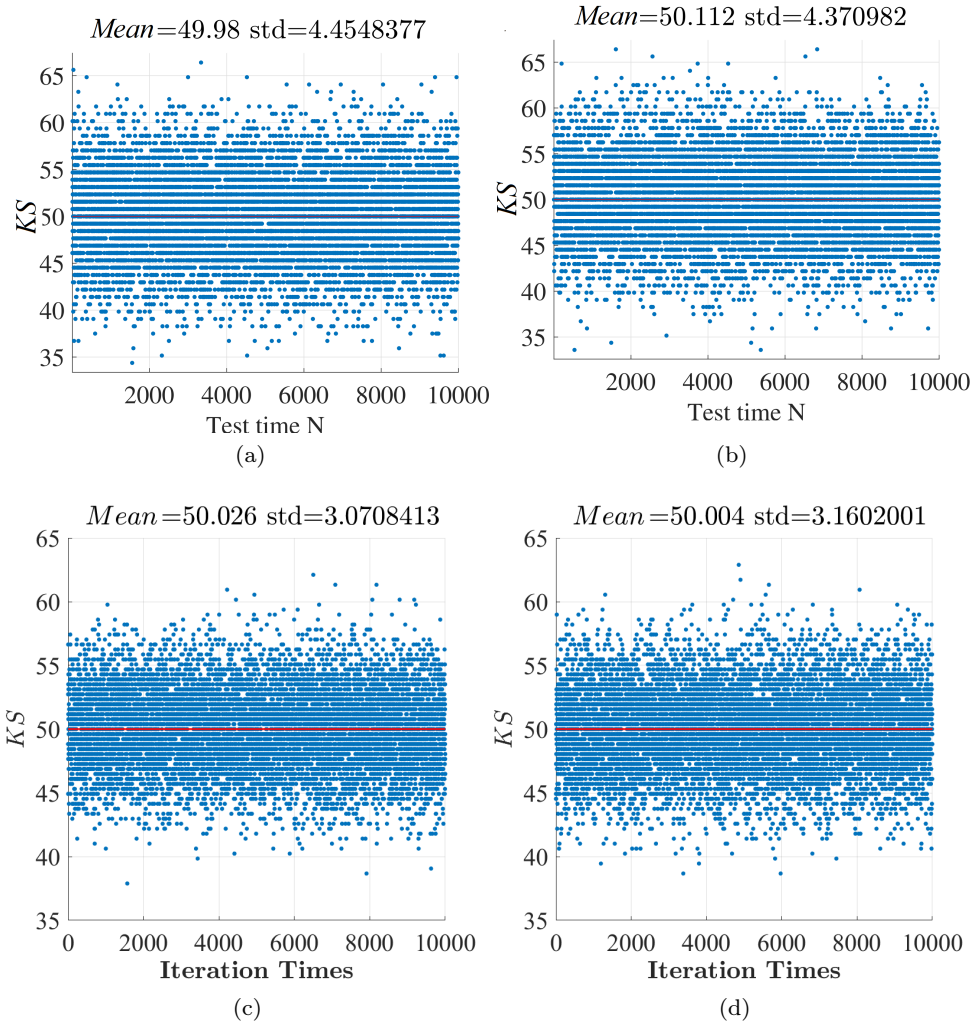


Fig. 9: Variation of the key sensitivity (KS) versus 1,000 random dynamic secret keys (changed random bit of the secret key) for the first (a) and second (b) variant, (c) CMAC, and (d) HMAC, respectively.

Table 6: Statistical Results of PS and KS

(a)

Statistical Results of PS

Tb	128	256	512	1024
min	31.25	37.11	42.38	43.16
max	69.53	62.5	59.57	56.83
Avg	49.99	49.97	49.98	49.98
STD	4.41	3.15	2.2164	1.6123

(b)

Statistical Results of KS

Tb	128	256	512	1024
min	33.59	38.67	41.99	42.28
max	66.4	61.72	57.42	56.15
Avg	50.11	49.97	49.98	49.98
STD	4.37	3.12	2.22	1.553

imposes 2^{64} trials for a brute force attack. As stated in (Damgård, 1990), "*if an appropriate padding scheme is used and the compression function is collision-resistant, then the hash function will also be collision resistant*". Indeed, the proposed method has sufficient size of the MAC value, and consequently the proposed variants are collision resistant, and thus, they are immune against birthday attacks.

6.4 Resistance to Meet-in-the-Middle Attack

Having a data content with nb blocks, $D = (D_1, D_2, \ldots, D_{nb-1}, D_{nb})$, the meet-in-the-middle attack aims at

Table 7: Percent distribution of the number of ASCII characters with the same value at the same location in the MAC value for random LSB bit of secret key (a) and message M (b) versus block size Tb.

(a) Secret key

Number of Hits					
hits / Tb	0	1	2	3	4
128	94.17	5.72	0.11	0	0
256	88.91	10.42	0.64	0.03	0
512	77.42	19.72	2.63	0.2	0.03
1024	60.54	30.26	7.81	1.11	0.28

(b) Plain message

Number of Hits					
hits / Tb	0	1	2	3	4
128	94.11	5.69	0.18	0.02	0
256	88.16	11.26	0.56	0.02	0
512	97.27	2.48	0.24	0.01	0
1024	62.05	28.86	7.67	1.1	0.32

finding a block D_i that can replace one of the nb blocks without changing the final MAC value (Amin et al., 2009; Kanso and Ghebleh, 2015). Since the proposed MAA solution processes the data blocks in a pseudo-random manner, and uses variable cryptograhic primitives for each input data, this renders such an attack impossible. To validate it, we replaced a randomly chosen data block by a random D_i and we computed the MAC values. This experiment was repeated $N = 1,000$ times. The results, in Fig. 10, show that the difference between the obtained MAC values is at least $Tb/2$ bits for $Tb = 128$, and similarly for Tb =256.

7 Performance Analysis

In this section, the efficiency of the proposed MAA solution is analysed with respect to its computational complexity and execution time.

7.1 Computational Delay

In the following, we detail the computational delay components of the proposed cryptographic primitives derivation scheme:

1. T_{xor} represents the time required to xor two blocks of TB bytes.
2. T_H represents the time required to hash one block of TB bytes.
3. T_{KSA} represents the time required to execute the RC4-KSA.
4. $T_{MKSA}(x)$ represents the time required to run the modified KSA of RC4 for an x-element table.
5. T_{PRNG} represents the time needed to execute the Pseudo-Random Number Generation function.

Thus, the total time required to derive the cryptographic primitives is expressed by CD_{KDF} as in the following equation:

$$CD_{KDF} = T_{xor} + T_H + 2 \times T_{KSA} + T_{MKSA}(nb) + T_{MKSA}(N) + T_{PRNG} \quad (7)$$

In fact, KSA and MKSA, used initially to generate the permutation and substitution tables, exhibit a low computational delay. Moreover, we employ a lightweight PRNG function to construct the round keys ($RK1$ and $RK2$).

Concerning the delay associated with the proposed MAA authentication process, it is assessed as follows:

1. T_S represents the time required for the substitution process of a block of TB bytes.
2. T_{xor} represents the time required for xoring two blocks of TB bytes.
3. T_{Sl} represents the time required for selecting a couple of input blocks.

Thus, the Computational Delay (CD) needed to execute the compression function of the authentication algorithm, on one data block, is represented by:

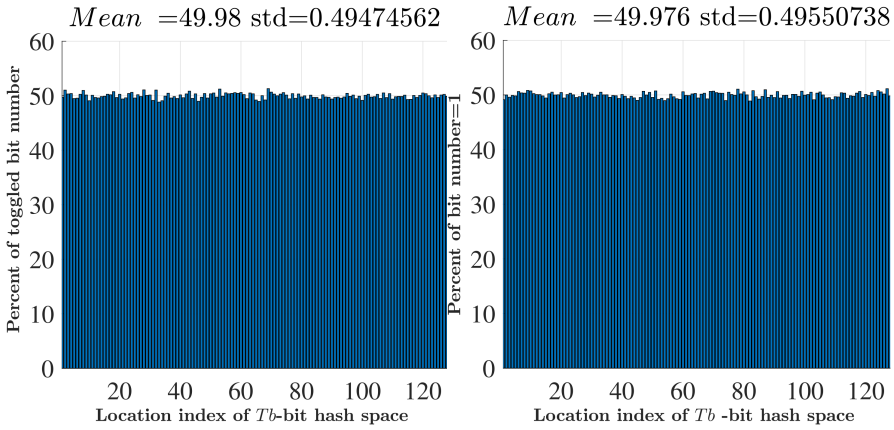


Fig. 10: Percent of the distribution of changed bit number between $H_n^{''}$ and H versus N tests (a) and its corresponding distribution (b).

$$CD_{V1} = 2 \times T_S + 2 \times T_{xor} \tag{8}$$

$$CD_{V2} = 2 \times T_S + 3 \times T_{xor} + \frac{T_{Sl}}{2} \tag{9}$$

To compare against the existing MAA solutions (CMAC and CCM operation mode), which are based on the AES encryption algorithm, we compute the total computational delay of the standard AES to encrypt one block (Daemen and Rijmen, 2013):

$$CD_{AES} = r \times T_S + (r+1) \times T_{xor} + (r-1) \times T_D + r \times T_{SR} \tag{10}$$

1. T_D represents the required time for the AES Mix-column.
2. T_{SR} represents the required time for the AES "shift-rows".
3. r represents the number of rounds.

Considering the minimum number of rounds (r=10) and minimum length of secret key (128 bits), the total time required for the AES operations can be computed as follows:

$$CD_{AES(r=10)} = 10 \times T_S + 11 \times T_{xor} + 9 \times T_D + 10 \times T_{SR} \tag{11}$$

Comparing the delays, it can be noticed that AES exhibits a larger delay compared to the proposed MAA solution, mainly due to the lack of diffusion operations.

Moreover, other MAA solutions like HMAC and CMAC require a higher execution time compared to the proposed MAA as shown in Table 8.

7.2 Execution Time

The proposed MAA variants are implemented in C and they are tested on 2 versions of Raspberry Pi (RPI-0 and RPI-3). In addition, the proposed variants are compared to the existing MAA, optimized OpenSSL HMAC and CMAC-AES. Fig. 11-a) shows the throughput (bytes/second) of the proposed MAA variants when using the listed Raspberry Pi devices for a message length $l = 786432$ bytes as input data (standard Lenna image) versus different sizes of TB. The obtained results show that the required execution time decreases when TB increases. In addition, the first variant was the fastest one, considering both Raspberry devices, and minimum latency is obtained with RPI-3.

On the other hand, Table 8 shows the throughput ratio between the proposed MAA variants and CMAC (AES-128) and HMAC (SHA-512) for different classes of Raspberry Pi devices. This reflects the benefit of using the proposed MAA variants instead of using the optimized CMAC and HMAC. These results show clearly that the proposed MAA variants require half and one quarter of the execution time of HMAC and CMAC, respectively.

The proposed MAA exhibits a lower execution time compared to the optimized CMAC-AES and HMAC. This indicates that the proposed MAA outperforms CMAC (AES) and HMAC implementations in OpenSSL. Moreover, we note that OpenSSL uses optimized assembly instructions to decrease the required execution time.

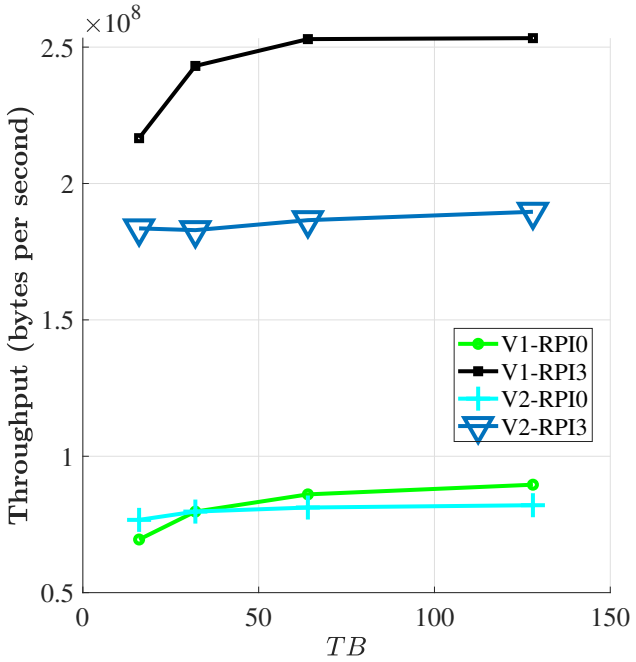


Fig. 11: Average throughput in bytes per second for 10,000 times, of the proposed variants for the input color Lenna image ($512 \times 512 \times 3$) versus TB ranging from 16 to 128 on (a) RPi-0 and RPi-3.

Thus, devices such as Raspberry (Pi0 and Pi3) or Arduino are not suitable for standard MAA (CMAC and HMAC) due to their limited processing capabilities. In this context, generally, the majority of the IoT tiny devices require an efficient and robust MAA solution. This paper addresses this problem and proposes an efficient solution for message authentication.

Also, let us point out that the second MAA variant decreases the throughput with a factor between 8% and 20% compared to the first one with RPI-0. Furthermore, the throughput of the second MAA variant is reduced also with a factor between 15% and 27% compared to the first one with RPI-3. Regardless of the overhead delay of the second variant, both variants achieve an enhanced performance compared to CMAC and HMAC as indicated in Table 8.
In conclusion, the proposed MAA variants achieve the desired cryptographic properties and require less latency compared to the current MAA cryptographic standards. This makes the proposed variants good candidate MAA solutions for restricted devices and real-time applications.

7.3 Flexibility

The proposed MAA deals with flexible block length, TB, which can be adjusted depending on the device's constraints and application requirements. Increasing Tb will make it more resistant against attacks, but this requires more powerful devices.

8 Conclusion

Message authentication is an essential security service to validate data correctness and source authenticity. With data proliferation, the shift towards a new networking scheme is key for handling the unexpected amounts of generated data. In this context, IoT emerges as a viable solution to enable several IoT innovative applications. Ensuring data integrity with source authentication in IoT is critical, given the ubiquitous access to published data. Dealing with big data and IoT constrained devices calls for lightweight security schemes. In this paper, a new MAA is proposed and it consists of two main phases: pseudo-random blocks selection and round compression function. Two variants of the round compression function are presented, each requiring a single round. The first variant processes one data block at a time, while the second variant processes two. Both variants use simple cryptographic operations ("exclusive-or" and substitution). The originality of this work is the design of an efficient MAA that strikes a good balance between the security level and system performance compared to existing MAA solutions. The later solutions require multi-rounds and multi-operations per round to achieve the key and data avalanche effects. In contrast, the proposed one is based on the dynamic key approach to meet these requirements by relying only on the key avalanche property. Security analysis tests were performed to prove the randomness, uniformity, high level of sensitivity and resistance against collision. Moreover, cryptanalysis discussion validates that the proposed solution is immune against well-known message authentication attacks. Finally, the computational complexity and throughput tests show that the proposed MAA exhibits a lower overhead in terms of processing and execution time compared to well-established existing solutions, like CMAC and HMAC.

Compliance with Ethical Standards

– **Funding:** This research was partially supported by funds from the Maroun Semaan Faculty of Engineering and Architecture at the American University of

Table 8: Throughput ratio between the proposed MAA variants and existing standard MAA (CMAC and HMAC) for $TB = 32$.

Standard MAA	**MAA Variant**	**Raspberry Pi0**	**Raspberry Pi3**
CMAC	First	4.29	4.38
HMAC		2.75	2.81
CMAC	Second	5.65	4.75
HMAC		2.57	2.16

Beirut and by the EIPHI Graduate School (contract "ANR-17-EURE-0002").

– **Conflict of interest:** The authors declare that they have no conflict of interest.
– **Ethical approval:** This article does not contain any studies with human participants or animals performed by any of the authors.

References

Eslam G AbdAllah, Hossam S Hassanein, and Mohammad Zulkernine. A survey of security attacks in information-centric networking. *IEEE Communications Surveys & Tutorials*, 17(3):1441–1454, 2015.

Amir Akhavan, Azman Samsudin, and Afshin Akhshani. A novel parallel hash function based on 3d chaotic map. *EURASIP Journal on Advances in Signal Processing*, 2013(1):1–12, 2013.

JM Amigó, Ljupco Kocarev, and Janus Szczepanski. Theory and practice of chaotic cryptography. *Physics Letters A*, 366(3):211–216, 2007.

Mohamed Amin, Osama S Faragallah, and Ahmed A Abd El-Latif. Chaos-based hash function (cbhf) for cryptographic applications. *Chaos, Solitons & Fractals*, 42(2):767–772, 2009.

Sobia Arshad, Muhammad Awais Azam, Mubashir Husain Rehmani, and Jonathan Loo. Recent advances in information-centric networking-based internet of things (ICN-IoT). *IEEE Internet of Things Journal*, 6(2):2128–2158, 2018.

Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. Sha-3 proposal blake. Submission to NIST (Round 3), 2010. URL `http://131002.net/blake/blake.pdf`.

Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. Simon and speck: Block ciphers for the internet of things. *IACR Cryptology ePrint Archive*, 2015:585, 2015.

G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The keccak reference. Submission to NIST (Round 3), 2011. URL `http://keccak.noekeon.org/Keccak-reference-3.0.pdf`.

Muhammad Bilal and Sangheon Pack. Secure distribution of protected content in information-centric networking. *IEEE Systems Journal*, 2019.

Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

Ivan Damgård. A design principle for hash functions. In *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '89, pages 416–427, London, UK, UK, 1990. Springer-Verlag. ISBN 3-540-97317-6.

Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The skein hask function family, 2009.

Nikos Fotiou and George C Polyzos. Securing content sharing over icn. In *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, pages 176–185. ACM, 2016.

Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Thomsen Søren S. Grøstl - a sha-3 candidate. In Helena Handschuh, Stefan Lucks, Bart Preneel, and Phillip Rogaway, editors, *Symmetric Cryptography*, number 09031 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. URL `http://drops.dagstuhl.de/opus/volltexte/2009/1955`.

R Guesmi, MAB Farah, A Kachouri, and M Samet. A novel chaos-based image encryption using dna sequence operation and secure hash algorithm sha-2. *Nonlinear Dynamics*, 83(3):1123–1136, 2016.

Helena Handschuh. Springer. ISBN 978-1-4419-5905-8.

A Kanso and M Ghebleh. A structure-based chaotic hashing scheme. *Nonlinear Dynamics*, 81(1-2):27–40, 2015.

H Krawczyk, M Bellare, and R Canetti. Hmac: Keyed-hashing for message authentication. 1997.

Bing Li, Dijiang Huang, Zhijie Wang, and Yan Zhu. Attribute-based access control for ICN naming scheme. *IEEE Transactions on Dependable and Secure Computing*, 15(2):194–206, 2016.

Naoki Masuda, Goce Jakimoski, Kazuyuki Aihara, and Ljupco Kocarev. Chaotic block ciphers: from theory to practical algorithms. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 53(6):1341–1352, 2006.

D. McGrew and J. Viega. The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH. RFC 4543 (Proposed Standard), may 2006.

Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996. ISBN 0849385237.

Ralph C. Merkle. A certified digital signature. In *Proceedings on Advances in cryptology*, CRYPTO '89, pages 218–238, New York, NY, USA, 1989. Springer-Verlag New York, Inc. ISBN 0-387-97317-6. URL `http://dl.acm.org/citation.cfm?id=118209.118230`.

Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Advanced Encryption Standard*. Alpha Press, 2009. ISBN 6130268297, 9786130268299.

Satyajayant Misra, Reza Tourani, Frank Natividad, Travis Mick, Nahid Ebrahimi Majd, and Hong Huang. Accconf: An access control framework for leveraging in-network cached data in the icn-enabled wireless edge. *IEEE Transactions on Dependable and Secure Computing*, 16(1):5–17, 2017.

Boubakr Nour, Fan Li, Hakima Khelifi, Hassine Moungla, and Adlen Ksentini. Coexistence of ICN and IP networks: An NFV as a service approach. 2019a.

Boubakr Nour, Kashif Sharif, Fan Li, and Yu Wang. Security and privacy challenges in information centric wireless IoT networks. 2019b.

Hassan Noura. *Design and simulation of efficient chaos based generators, crypto-systems and hash functions*. Theses, UNIVERSITE DE NANTES, August 2012.

Hassan Noura, Ali Chehab, Lama Sleem, Mohamad Noura, Raphaël Couturier, and Mohammad M Mansour. One round cipher algorithm for multimedia IoT devices. *Multimedia tools and applications*, 77(14): 18383–18413, 2018a.

Hassan Noura, Ali Chehab, Mohamad Noura, Raphaël Couturier, and Mohammad M Mansour. Lightweight, dynamic and efficient image encryption scheme. *Multimedia Tools and Applications*, 78(12):16527–16561, 2019a.

Hassan Noura, Raphaël Couturier, Congduc Pham, and Ali Chehab. Lightweight stream cipher scheme for resource-constrained iot devices. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8. IEEE, 2019b.

Hassan Noura, Ola Salman, Ali Chehab, and Raphaël Couturier. Preserving data security in distributed fog computing. *Ad Hoc Networks*, 94:101937, 2019c. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2019.101937. URL `http://www.sciencedirect.com/science/article/pii/S1570870519303634`.

Hassan N Noura and Damien Courousse. Method of encryption with dynamic diffusion and confusion layers, June 9 2016. URL `https://www.google.com/patents/WO2016087520A1?cl=en`. WO Patent App. PCT/EP2015/078,372.

Hassan N Noura, Lama Sleem, Mohamad Noura, Mohammad M. Mansour, Ali Chehab, and Raphaël Couturier. A new efficient lightweight and secure image cipher scheme. *Multimedia Tools and Applications*, Sep 2017. ISSN 1573-7721.

Hassan N Noura, Mohamad Noura, Ali Chehab, Mohammad M Mansour, and Raphaël Couturier. Efficient and secure cipher scheme for multimedia contents. *Multimedia Tools and Applications*, pages 1–30, 2018b.

Hassan N. Noura, Ali Chehab, and Raphaël Couturier. Efficient & secure cipher scheme with dynamic key-dependent mode of operation. *Signal Processing: Image Communication*, 78:448 – 464, 2019d. ISSN 0923-5965.

Andy Patrizio. Idc: Expect 175 zettabytes of data worldwide by 2025 | network world. `https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html`, DECEMBER 2018.

Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons, 2007.

JH. Song, R. Poovendran, J. Lee, and T. Iwata. RFC 4493 (Informational), June .

William Stallings. *Cryptography and network security: principles and practice*. Pearson Upper Saddle River, 2017.

Je Sen Teh, Azman Samsudin, and Amir Akhavan. Parallel chaotic hash function based on the shuffle-exchange network. *Nonlinear Dynamics*, 81(3):1067–1079, 2015.

Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *In EUROCRYPT*. Springer-Verlag, 2005.

Qi Wu. A chaos-based hash function. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2015 International Conference on*, pages 1–4. IEEE, 2015.

Fei Xiang, Changwei Zhao, Jian Wang, and Zhiyong Zhang. One-way hash function based on cascade

chaos. *The Open Cybernetics & Systemics Journal*, 9(1), 2015.

Kaiping Xue, Xiang Zhang, Qiudong Xia, David SL Wei, Hao Yue, and Feng Wu. SEAF: A secure, efficient and accountable access control framework for information centric networking. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2213–2221. IEEE, 2018.

Jean-Paul Yaacoub, Ola Salman, Hassan N Noura, and Ali Chehab. Security analysis of drones systems: Attacks, limitations, and recommendations. *Internet of Things*, page 100218, 2020a.

Jean-Paul A Yaacoub, Mohamad Noura, Hassan N Noura, Ola Salman, Elias Yaacoub, Raphaël Couturier, and Ali Chehab. Securing internet of medical things systems: limitations, issues and recommendations. *Future Generation Computer Systems*, 105: 581–606, 2020b.

B. Yang, Z. Li, S. Zheng, and Y. Yang. Hash function construction based on coupled map lattice for communication security. In *Global Mobile Congress 2009*, pages 1–7, Oct 2009.