A Single Pass and One Round Message Authentication-Encryption for Limited IoT Devices

Hassan N. Noura[1], Ola Salman[2], Raphaël Couturier[1], and Ali Chehab[2]

[1]Univ. Bourgogne Franche-Comté (UBFC), FEMTO-ST Institute, CNRS, Belfort, France
[2]American University of Beirut, Electrical and Computer Engineering Department, Lebanon

***Abstract*—In this work, we propose three efficient variants of a Message Authentication Encryption (MAE) algorithm, which is based on the dynamic key-dependent concept and dynamic operation mode to reach a high level of security. These variants consist of a single pass and a single round, in addition to using common operations for the encryption and authentication processes to reduce the required execution time and resources. Accordingly, the proposed scheme outperforms the existing solutions that are based on the static approach with multiple rounds. Furthermore, to reduce the overhead associated with the regeneration of the dynamic key and the corresponding cryptographic primitives, we propose a simple, yet effective update process. In such a scheme, even when the same plaintext is processed, it will be encrypted and authenticated using different cryptographic primitives (substitution and permutation tables in addition to round keys), which guards against the existing cryptanalysis techniques. The experimental results show that the proposed MAE variants are more efficient than the Counter with Cipher block Chaining Message authentication code (CCM), Galois Message Authentication Code (GMAC), Offset Codebook Mode (OCB), and the Chacha20-poly1305. The best performance is achieved with the third MAE variant that presents a high throughput with an enhancement of at least 373% compared to CCM, 90% compared to GCM, 23% compared to OCB, and 22% compared to Chacha20-poly1305.**

***Index Terms*—Lightweight message authentication encryption algorithm; Security analysis; Performance analysis**

I. Introduction

Emerging digital systems 1) incorporate a large number of connected devices, some of which are constrained in terms of energy and computational power, 2) generate tremendous amounts of data, and 3) some of them impose a real-time response. Such systems give rise to new security threats, which can cause drastic damage to the corresponding resources and data. Hence, the adoption of security services in such systems is becoming critical for their availability and the protection of their data.

Typically, the main security services such as data confidentiality, data integrity, and source authentication rely on a strong symmetric cipher algorithm. Data confidentiality is achieved through a symmetric encryption algorithm, either a block cipher or a stream cipher [1], [2]. Data integrity employs an unkeyed hash function, also called a Modification Detection Code (MDC) [3]. Source authentication is verified through the use of a Message Authentication Code (MAC) [4] that is based on a keyed hash function, or a block cipher in Cipher Block Chaining (CBC) mode, such as the case of CBC-MAC and Cipher-based MAC (CMAC) [5], [6]. In general, these three services are implemented separately, however, many applications that involve the transmission of sensitive data require that data confidentiality, data integrity, and source authentication be achieved simultaneously. To that end, a Message Authentication-Encryption (MAE) algorithm can be used, which typically achieves data confidentiality, data integrity, and source authentication by combining two separate processes, a conventional encryption algorithm, and a MAC based on the CBC operation mode [7].

An MAE algorithm (aka authenticated cipher) is typically used to encrypt and authenticate messages using a shared secret Session Key (SK) and a public Initial Vector (IV). Several message authentication schemes have been presented [8], and they can be classified into two categories. The first category uses a block cipher and a secure hash algorithm such as the Advanced Encryption Standard (AES) with CBC mode (CMAC), or the Hashed Message Authentication Code (HMAC) [9]. The other category is the Galois Message Authentication Code (GMAC) that

exhibits a better performance compared to CMAC and HMAC, but unfortunately, it suffers from security issues.

On the other hand, many advanced authentication encryption techniques are based only on a block cipher for both message encryption and authentication, to simplify the hardware implementation. The existing MAE schemes include Counter with CBC Message authentication code (CCM), Galois Message Authentication Code (GCM) [10], Offset Codebook Mode (OCB) [11], and Chacha20-poly1305 [12]. They can be employed currently in limited devices such as Internet of Things (IoT) [13], and they have been adopted in many standards such as Secure Shell (SSH), Secure Sockets Layer (SSL)/Transport Layer Security (TLS), and IP Security (IPsec) [14].

A. Problem

It is becoming essential to design a lightweight and robust MAE algorithm that accounts for the limitations of tiny devices and real-time applications. One possible solution is to adopt a single pass encryption-authentication algorithm using minimal resources and thus, exhibiting minimal delay. The existing lightweight approaches are not efficient enough since they process an input message at least twice, once for encryption, and a second time for authentication. Also, some of these approaches are not secure enough due to inappropriate combination of the encryption and authentication mechanisms [15]. Concerning the traditional symmetric cryptographic algorithms, they require multiple iterations of a round function, which includes several operations of substitution and diffusion.

The existing MAE algorithms are not appropriate for delay-sensitive applications nor for limited devices commonly available in IoT systems [16]; they cannot make use of the optimized AES instructions. Some lightweight techniques are based on chaotic cryptographic algorithms, however, these are not practical due to the need for floating-point computations, conversion operations, and complex hardware implementation.

To address these issues, recent research has focused on the development of a new class of lightweight cryptographic algorithms [16], [17], such as the techniques presented in [18]-[23]; These are mainly based on the concept of dynamic keys, and they require one or two rounds. As such, they exhibit a low latency and require few resources, while maintaining a high level of security.

B. Motivation

In this work, we present a new lightweight MAE solution that consists of a single pass and a single iteration of a round function. The scheme is designed to use common operations for encryption and authentication. Such efficiency is achieved due to 1) the use of variable cryptographic primitives, which eliminates the need for a large number of rounds, and 2) the elimination of a second pass for authentication. This makes the algorithm appropriate for limited devices and the required Quality of Service (QoS) of real-time applications [24], [25], [26].

C. Contributions

The proposed solution offers simultaneously the essential security services, data confidentiality, data integrity, and source authentication, while maintaining a low overhead in terms of computational complexity and latency. We present three variations of a new lightweight and secure MAE algorithm, based on the dynamic key-dependent cryptographic concept. The cryptographic structure of the schemes consists of a single pass and a single round with few operations. As compared to our previous works [21], [27], [28], the proposed structure makes use of common steps between encryption and authentication, to reduce the required number of operations for higher efficiency. Another enhancement is the use of a simple update process for the cryptographic primitives instead of regenerating them; the process is simply based on a permutation operation. Also, the update process is no longer required for each new message, instead, it can be configured to a specific set of messages. In the following, we present the advantages of the proposed solution in terms of system performance and security level.

System Performance:

- **Efficiency**: The proposed schemes exhibit low computational complexity by encrypting and authenticating one block at a time, for the first variant, and two blocks for the second variant. The operations involved are simple since they do not include any diffusion operation, in contrast to existing cryptographic algorithms such as AES. Moreover, the proposed encryption scheme does not require any chaining mode of operation, and hence, encryption and decryption can be realized in parallel.
- **Flexibility**: The different variants are flexible since the size of the input block (N bytes) is configurable, and it is set according to the requirements of the devices and the application.

- **Simple software & hardware implementations:** The proposed structure lends itself to simple hardware and software implementations since it makes use of simple operations, such as the "exclusive or (xor)" logical operation, look-up for substitution, and a simple permutation.
- **Error tolerance**: The proposed variants are designed to resist channel errors; a bit-error in an encrypted byte affects only the corresponding byte. The errors do not affect the entire block since the avalanche effect is achieved differently than traditional schemes. As such, the proposed structure is very practical for error detection and correction techniques.

Security and Robustness:

- **Dynamic key-dependent approach:** The session key SK is used to generate one or a set of dynamic vectors for sub-session configuration; the vectors are then used to generate a set of cryptographic primitives, and their corresponding update primitives that are used to modify the cryptographic primitives (round keys, substitution, and permutation tables), for each new input message, in a pseudo-random manner. This makes analytic attacks and implementation attacks very challenging [18], [29], [30], [31]. To further enhance the security of the proposed MAE variants, two distinct substitution tables are used instead of one.
- **Dynamic mode of operation**: The existing block ciphers adopt the static approach with fixed cryptographic primitives, and the blocks are encrypted and authenticated sequentially. The proposed MAE variants rely on a dynamic permutation table, which performs pseudo-random selection of a block. Thus, the relationship between the plain blocks becomes more complex and random, and the encryption-authentication sequence changes for each input message. This also enhances the resistance against analytic and implementation attacks.

Finally, the dynamic key approach exhibits typically a disadvantage related to the initialization phase, during which the dynamic key is generated and the cryptographic primitives are constructed. To reduce this overhead, we propose a simple update process for the generation of the cryptographic primitives.

D. Organization

The rest of this paper is structured as follows. Existing message authentication and/or encryption mechanisms such as CBC, CMAC, and CCM, are presented in Section II, in addition to a description of Internet of Things networks. The proposed key derivation function and the construction techniques for the cryptographic primitives and their updates, are presented in Section IV. The proposed MAE variants and the dynamic mode of operation are described in Section V. Section VI analyzes the cryptographic properties such as randomness, uniformity and sensitivity. Then, in Section VII, the robustness of the proposed variants against confidentiality and authentication attacks are discussed. The performance of the proposed MAE variants is analyzed in Section VIII to assess their effectiveness. Finally, the paper is concluded in section IX.

II. Background

This section provides an overview of the various message authentication and encryption techniques, as well as the IoT networking paradigm.

A. Message Authentication-Encryption

Symmetric MAC approaches are based on the Merkle-Dangard structure [32], [33]. The block cipher message authentication approach requires less computational complexity, which makes it preferable for limited devices or real-time applications. The existing message encryption algorithms rely on a block cipher with the counter mode, to achieve a good performance and a high level of robustness. Note that a block cipher with the counter mode is actually a stream cipher.

From a mathematical perspective, a symmetric encryption algorithm [34] is an invertible non-linear function $E()$: $\{0,1\}^{Tb} \rightarrow \{0,1\}^{Tb}$, where Tb represents the size of an input block in bits, an integer with a value of 128 or 256. On the other hand, a cryptographic hash function [3] is a non-invertible non-linear compression function h: $\{0,1\}^{*} \rightarrow \{0,1\}^{Tb}$. Typically, an MAE algorithm divides the input data, M, of size $|M|$, into nb blocks, each of Tb bits.

In general, a symmetric MAE algorithm takes as input a plaintext message, a secret key (K), and a public initial value IV (e.g. counter, identity, etc.) for the encryption process, and another one for the authentication process. The output is a ciphertext that has the same length of the plaintext and Tb-bit MAC that is appended to the message. The receiver computes the MAC of the received message ($CMAC$) and compares it to the received message authentication code ($RMAC$) for message authentication. If they are equal, the message is authenticated.

There are several existing MAE schemes in the literature; CCM and GCM were approved by the National Institute of Standards and Technology (NIST) [10], [6], [13]. CCM uses a robust block cipher such as AES with 1) the CounTeR (CTR) mode for encryption, and 2) the CBC mode for message authentication. The authentication process can be done in parallel with the encryption process, and the obtained MAC is encrypted. The main issue with CCM is that it needs to apply AES twice for each input block, and each AES iteration requires r rounds of substitution and diffusion operations. On the other hand, GCM reduces the computational overhead of message authentication by using a Galois multiplication operation instead of a block cipher. This makes GCM more efficient than CCM, however, it suffers from several security issues.

Another scheme, the Offset Codebook Mode (OCB) [11] requires only one pass instead of two, compared to CCM and GCM, which makes it more efficient. The encryption process, in OCB mode, is similar to the ECB mode, and it overcomes the limitations of the latter (same plaintext block leads to same ciphertext block). A recent MAE solution, ChaCha20-Poly1305 [35], has been proposed to address the scenario when AES hardware acceleration is not feasible. It combines the ChaCha20 stream cipher and the Poly1305 authentication. The authors showed that the algorithm exhibits a similar performance to the OCB mode in terms of latency.

When compared to the above solutions, the proposed MAE structure is designed to be lightweight since 1) it consists of a single pass and just one iteration of a round function, and 2) it uses common steps for the two processes of encryption and authentication. Also, it exhibits a high level of security due to the use of the dynamic key approach and the varying cryptographic primitives.

B. Internet of Things Network

The IoT architecture is divided into fours layers, devices, gateway, network, and application. The IoT devices can be grouped into sub-networks, with a gateway connected to each sub-network. These gateways are linked together via the core network, whose function is to coordinate the inter-networking and communication among the IoT sub-networks. Finally, the application layer is responsible for managing the data generated by the various IoT applications. It is important to note that each layer has its own security management role. Due to the limitations of some IoT devices, the need arises for lightweight security solutions.

III. Problem Statement

The system and threat models are described in detail in this section to highlight the significance of the proposed MAE variants.

A. System Model

The network model is made up of n IoT nodes, each with a specific amount of energy and computing power. The IoT network is connected to these nodes through a gateway. One of the proposed MAE variants can be applied at the devices' level and at the application server level (or cloud). Note that at the legal destination (device or application server), the data integrity and authentication are performed first, followed by the data decryption process.

We assume that a Pre-Shared master Key (PSK) is shared between the communicating entities and a trusted server, which is used to generate and exchange a symmetric key (SK) between the communicating entities. Then, the following two steps are performed at the source entities (IoT devices).

1) For each new session, a new SK is generated and shared between each pair of IoT devices and application server(s). This key is used to produce a dynamic vector DV that should be unique for each new session. Different key expansion techniques could be used for this purpose such as the one used in AES.
2) DV is used to construct the cryptographic primitives needed for the MAE algorithm to encrypt and authenticate the communicated packets. The packets will end up containing a ciphertext and a MAC appended at the end of the ciphertext payload.

At the destination (application server-side or IoT devices), the application server acquires the corresponding dynamic vector, DV, and produces the cryptographic primitives to decrypt and check the source authentication and data integrity.

B. Threats/Attacks Model

IoT networks are facing many security challenges and are subject to various attacks [36], [37] that could compromise their data, users and systems. Different types of security attacks are considered in this work, with a description of the mechanisms used by attackers:

- **Eavesdropping Attacks**: The attacker intercepts a communicated message between IoT devices and

application server(s)/cloud. Since the messages are encrypted, and with different dynamic keys, this prevents eavesdroppers from extracting any useful information.

- **Analytic Cryptanalysis**: A more serious attack occurs when an attacker attempts to deduce the secret key using any of existing cryptanalysis techniques (linear, differential, key-related, or brute force attacks). The proposed MAE variants are based on the dynamic key approach and thus, they are immune against such cryptanalysis techniques, which were designed to target static keys and cryptographic primitives. On the other hand, the size of the session and dynamic keys is sufficiently large to prevent brute force attacks.
- **Man-in-the Middle Attacks**: An adversary is able to communicate with IoT devices impersonating an application server, and with the application server impersonating an IoT device. Such an attacker has the ability to intercept all communicated messages, decrypt them, modify them, and finally, re-authenticate them. Also, the attacker could delete messages or generate new ones. This attack is mainly due to a weakness in the authentication protocol, and not to the underlying cryptographic algorithm. Therefore, to prevent such attacks, there is a need to either adopt strong multi-factor authentication protocols [38], [39], [40], or to rely on physical layer characteristics of IoT devices such as Radio-Frequency Fingerprinting (RFF), with One-Time Password (OTP), for device authentication [41], [42]. Moreover, IoT device authentication schemes could be based on machine learning solutions such as SVM [43] for higher accuracy and precision.
- **Message Tampering-Alteration Attacks**: The attacker aims to manipulate the exchanged messages and thus, compromise the data integrity. This can be done by collecting ciphertext via a compromised gateway or an intermediate node. However, any modification in even a bit of the ciphertext will lead to a different MAC. A secure communication, based on any of the proposed MAE solutions, enables the application server(s) to detect such type of data manipulation.
- **Malicious Data Injection Attacks**: This kind of attacks is based on compromising an IoT device. The attacker manipulates the device, generate fake messages, and transmits them to the application server(s). The weakness is not related to the cryptographic algorithm, and it can be mitigated using a lightweight intrusion detection system with device system integrity.
- **Masquerading Attacks**: This occurs when either an IoT device, gateway or application server (or faked one) is exploited by an attacker for malicious purposes. The attacker is able to send false messages or modify received ones. Again, this type of attacks cannot be prevented via cryptographic algorithms, but instead through a set of security mechanisms such as an intrusion detection system and multi-factor authentication [44], [45].
- **Replay Attacks:** An attacker transmits again a previously communicated message (encrypted data with correct MAC). This could be used to perform a DDoS attack against application server(s) or IoT device(s) and to hinder the availability of communication with the destination device. Such an attack is easily detectable in the proposed solution since the cryptographic primitives are updated for each new input message.
- **Implementation Attacks**: An attacker, with physical access to IoT devices, performs side channel attacks or fault attacks. The importance of the proposed solution is that it ensures random physical properties such as energy consumption due to the use of dynamic cryptographic primitives, which gives immunity at the algorithmic level. This could also be complemented with countermeasures at the hardware or compiler level.
- A message may also be altered as a result of a channel error (interference, fading). The storage and decryption of incorrectly received messages are avoided in the proposed solution.

IV. Proposed Cryptographic Primitives Construction

The construction of the dynamic cryptographic primitives is illustrated in Figure 1, and all used notations are listed in Table I. Each new session starts with a new shared secret SK between an authorized IoT device and an application server or cloud. The focus of this work is not on key management between legal entities. Readers can refer to [34] for further information on possible key management schemes.

The dynamic vector DV, with a size of 64 bytes, could be produced by hashing the SK, or by using any secure key derivation function. Then, DV is split into two main parts; the first one is used to construct the cryptographic primitives, and the second one to construct the corresponding update primitives. DV could be produced synchronously by IoT devices and the application server(s).

Table I: Summary of notations

Notation	Definition
SK	A shared secret session key
DV	A dynamic vector that uses to construct the required cryptographic primitives
M	The original message input (plaintext)
m_i	The i^{th} plaintext block
C	The encrypted message (ciphertext)
c_i	The i^{th} ciphertext block
$\|M\|$	Input message length in bits
N	Number of bytes in one block message
Tb	Number of bits in a data block and it is equal to $8 \times N$
nb	The number of blocks in a single input message is equal to $\lceil \frac{\|M\|}{Tb} \rceil$
n	Number of IoT nodes
KS_i	i^{th} Substitution sub-key
S_i	The i^{th} dynamic substitution table produced by using kS_i sub-key
$S_i(x)$	The bytes of x block is substituted by using the i^{th} substitution table
$S_i^{-1}(x)$	The bytes of x block is substituted by using the i^{th} inverse substitution table S_i^{-1}
$x \oplus y$	"Exclusive or" two blocks x and y
K_{US}	The update substitution sub-Key
US	The update substitution table produced by using the K_{US} sub-key. It is used to update the substitution table π
K_π	Permutation sub-Key
π	Permutation table produced by using K_π sub-key. It is used as selection table to decide the processing order of input block from M or C during message encryption/decryption authentication process
K_{UP}	The update permutation sub-Key
$U\pi$	The update permutation table produced by using the K_{UP} sub-key. It is used to update the permutation table π
K_{RK}	Round key generation sub-key and it is used to produce two set of round keys $RK1$ and $RK2$
$RK1$	m dynamic round keys and represents the first set of round keys
$RK2$	The second set of round keys and it consists also of m dynamic round keys
m	Number of round keys
K_{SRK}	The round selection sub-key
SRK	A selection round table and is used to determine which round keys of $RK1$ and $RK2$ are utilized for each input block during encryption/decryption-authentication processes.
K_{USRK}	The update selection sub-Key
$USRK$	The update selection table produced by using the K_{USRK} sub-key. It is used to update the selection table SRK
$Y[j]$	It represents the j^{th} element of the table Y

The cryptographic primitives are updated for each input message using the update primitives and a packet counter, and thus, each input message is processed using different cryptographic primitives. The dynamic vector DV is divided into two parts: cryptographic keys ($\{KS_1,\ KS_{S2},\ K_P,\ K_{RK},\ K_{SRK}\}$) and update primitives $\{KS_{US},\ KS_{UP},\ KS_{USRK}\}$. Each of the 8 sub-keys has a length of 64 bits (8 bytes), starting from the most significant bytes assigned to KS_1, and so on. Each dynamic sub-key is utilized for a specific purpose, as explained below:

- **Substitution Sub-key** KS_1: It is used to build the first substitution table S_1 using the technique presented in [21], which uses the Key Setup Algorithm (KSA) of Rivest Cipher 4 (RC4). The substitution process is performed at the byte level, and the elements in the substitution table S have values ranging from 0 to 255.
- **Substitution Sub-key** kS_2: In a similar manner to KS_1, it is used to build the second substitution table S_2.
- **Permutation Sub-key** k_P: It is employed to build a flexible permutation table π, having nb elements, using the modified KSA of RC4, which was proposed in [21]. The elements of the permutation table π have values varying from 1 to nb.
- **Round Key Generation Sub-key** k_{RK}: It is used as a seed for an efficient stream cipher, which will be iterated to produce $(2 \times m \times N)$ bytes, where m represents the number of rounds keys, and N the number of bytes for each round key. Then, the produced key-stream RK is divided into two equal parts ($RK1$ and $RK2$), where each one represents a set of round keys with a size of $(m \times N)$, resulting into two different sets of m round keys $RK1 = RK1_1,\ RK1_2,\ \ldots,\ RK1_m$ and $RK2 = RK2_1,\ RK2_2,\ \ldots,\ RK2_m$. Each $RK1_w$ or $RK2_w$ for $w = 1,\ 2,\ \ldots,\ m$ has N bytes. Any

stream cipher or key expansion algorithm could be used to generate these round keys. Moreover, m is a configurable value that is set based on memory limitations.

- **Dynamic Round Key Selection** k_{SRK}: This is the Selection Round Key (SRK) table, with nb elements, each having a value between 1 and m. This vector can also be generated using a stream cipher.
- **Update Substitution Sub-key** KS_{US}: It is used to build a substitution table SUS for updating the substitution table.
- **Update Permutation Sub-key** KS_{UP}: It is used to build a permutation table π_{UP}, which is used to update the selection block table.
- **Update Selection Round-key Sub-key** KS_{USRK}: It is used to build a permutation table $USRK$, which is used to update the round key selection table π_{SRK}.

It is important to note that any bit change in the secret key generates a different DV vector, and different cryptographic and update primitives. This confirms the high key sensitivity of the proposed MAE approach.

At the legal destination, the inverse substitution tables (S_1^{-1} and S_2^{-1}) are needed, and can be obtained by applying the following equation:

$$S_j^{-1}[S_j[w]] = w \text{ for } w = 0,\ 1,\ \ldots,\ 255 \text{ and } j = 1 \text{ or } 2 \quad (1)$$

V. Proposed Message Authentication-Encryption Algorithm

Any type of data communication such as multimedia or text files, may be handled by the proposed MAE algorithm. Also, the proposed encryption/decryption algorithm is performed in parallel, while the message authentication is performed with a chaining process. The cipher scheme can be viewed as a dynamic CTR mode (D-CTR). The MAE technique selects the blocks to be encrypted and authenticated based on a dynamic pseudo-random sequence rather than the traditional approach that uses a sequential order. A dynamic permutation table is used to select the order of input blocks, and it is updated for each input message. Two blocks are mixed pseudo-randomly, in the second version, to further complicate the cryptanalysis process.

A dynamic vector is used for each session, and the following cryptographic primitives are generated:

1) $RK1$ and $RK2$, each consisting of m round keys;
2) A selection table SRK;
3) S_1 and S_2, the first and second substitution tables;
4) π, the permutation table used for the selection of input blocks.

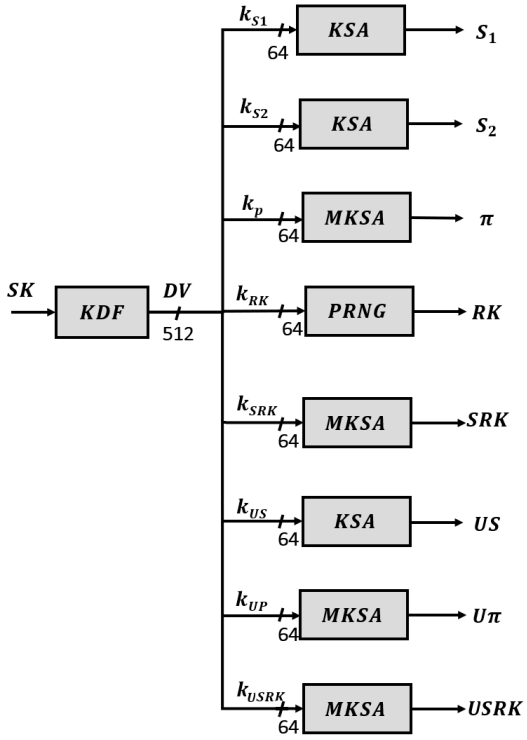


Figure 1: The proposed dynamic key derivation approach in addition to the proposed approach to build the required cryptographic and update cryptographic primitives.

In addition, three update primitives are also produced, US, U_π, and $USRK$. US is used to substitute and update the first and the second substitution tables (S_1 and S_2). U_π is used to permute and update the permutation table π. while $USRK$ is used also to permute and update SRK. The cryptographic and update primitives are required at the source and destination. However, at the destination, the substitution tables are replaced by their inverses (S_1^{-1} and S_2^{-1}).

The message is divided into nb blocks, and padding is used whenever necessary. For the second variant, nb should be even since two blocks are processed at a time. The input blocks, $M = m_1,\ m_2,\ \ldots,\ m_{nb}$, each has N bytes, where N is a configurable parameter that changes depending on the application or device limitations. In real-time applications, a smaller N value is preferred. We shall fix N to 32 in the rest of the paper, and each block will have a size of 256 bits.

A. Encryption/Decryption Algorithms

There are two sub-functions in the proposed MAE algorithm: BlocksSelection, and RoundFunction(RF), as described next.

*1) **Blocks Selection**:* In each iteration, the proposed MAE algorithm encrypts and authenticates one block ($m[\pi(i)]$) in the first variant for $i = \{1\ ,\ 2,\ \ldots,\ nb\}$, and two input blocks ($m[\pi(i)]$ and $m[\pi(i + \frac{nb}{2})]$) in the second variant, for $i = \{1\ ,\ 2,\ \ldots, \frac{nb}{2}\}$. The input blocks are selected using the dynamic permutation table π, and $m[\pi(i)]$ refers to the $\pi(i)^{th}$ input block.

*2) **First Variant of MAE Round Function** RF:* The round function employs substitution tables S_1 and S_2 in addition to the permutation table π for input block selection, and the selection round key table SRK, as well as a pair of m round keys ($RK1$ and $RK2$). The round keys $RK1[SRK(i)]$ and $RK2[SRK(i + \frac{nb}{2})]$ are utilized for the i^{th} input block, respectively.

The first variant of the message authentication-encryption algorithm, with one input block per iteration, is described in Algorithm 1.

Algorithm 1 The proposed MAE algorithm's first variant, which only processes one input block per thread.

procedure MAE_1VARIANT($M,\ S_1,\ S_2,\ \pi,\ RK_1,\ RK_2,\ SRK,\ IV,\ nb$)
  for $i = 1$ to nb **do**
    $temp = S_1(RK_1[SRK[i]] \oplus M[\pi[i]])$
    $C[i] = S_2(RK_2[SRK[i]] \oplus temp)$
    $IV = S_2(temp \oplus IV)$
  end for
  $MAC = S_1(IV))$
end procedure

Algorithm 2 The corresponding Message Authentication-Decryption (MAD) algorithm of the proposed first variant

procedure MAD_1VARIANT($C,\ S_1,\ S_2,\ S_1^{-1},\ S_2^{-1},\ \pi,\ RK_1,\ RK_2,\ SRK,\ IV,\ nb$)
  for $i = 1$ to nb **do**
    $temp = S_2^{-1}(C[i]) \oplus RK2[SRK[i]]$
    $IV = S_2(IV \oplus temp)$
    $M[\pi[i]] = S_1^{-1}(temp) \oplus RK1[SRK[i]])$
  end for
  $MAC = S_1(IV))$
end procedure

In each iteration of the first variant, an input block ($m[\pi(i)]$) is selected and encrypted-authenticated to produce the encrypted block $c[(i)]$ and to update IV as illustrated in Algorithm 1.

$m[\pi(i)]$ represents the $\pi(i)^{th}$ input block and it is encrypted and stored at the i^{th} encrypted block index by doing the following operations:

1) The $SRK(i)^{th}$ round key of $RK1$, ($RK1[SRK(i)]$) is mixed ("xor") with the $\pi(i)^{th}$ plain block ($m[\pi(i)]$).
2) Then, the first substitution table S_1 is used to substitute the corresponding output of the previous step to produce $temp$.
3) Next, the substituted output $temp$ is mixed with the $SRK(i)^{th}$ round key of $RK2$, ($RK2[SRK(i)]$).
4) Finally, the output is subjected to another substitution operation by using the second substitution table S_2.

In addition, IV is updated for each iteration by mixing the current IV with $temp$, and the output of this operation is substituted by using the second substitution table S_2. After iterating all input blocks, the final IV is substituted by using the first substitution table to produce the MAC.

The decryption algorithm follows the same steps, but 1) it uses the inverse round function RF^{-1}, which operates in a reverse order compared to the round function RF, and 2) instead of S_1 and S_2, RF^{-1} employs the inverse substitution tables S_1^{-1} and S_2^{-1}.

Algorithm 2 describes the inverse function RF^{-1}, which deals with one input block at a time. Another possible mode of operation for the first variant is to process two input blocks $m[\pi(i + \frac{nb}{2})]$ and $m[\pi(i)]$) in parallel, instead of one block. This variant is flexible since it can encrypt and authenticate α blocks at each iteration, but this requires nb to be divisible by α. Also, α IVs will be produced and substituted, and then, mixed to produce the final MAC. For $\alpha = 2$, the final MAC value is obtained by performing "xor" of the substituted values of $IV1$ and $IV2$. Note that the blocks use the same cryptographic primitives.

*3) **Second Variant of Round Function** RF:* In the second variant, two input blocks ($m_{\pi(i+\frac{nb}{2})}$ and $m_{\pi(i)}$) are chosen and mixed in each iteration. As shown in Algorithm 3, two encrypted blocks ($C_(i)$ and $C_{\pi(i+\frac{nb}{2})}$) are produced in addition to updating $IV1$ and $IV2$. The message authentication-encryption algorithm of the second variant with two input blocks is presented in Algorithm 3.

The first encrypted block $C_{(i)}$ is obtained by doing the following operations:

1) The $SRK(i)^{th}$ round key of $RK1$, which is $RK1[SRK(i)]$ is mixed ("xor") with the $\pi(i)^{th}$ input plain block ($m[\pi(i)]$) and the $\pi(i+\frac{nb}{2})^{th}$ message plain block $m[\pi(i+\frac{nb}{2})]$.
2) Then, the first substitution table S_1 is used to substitute the corresponding output of the previous step to produce $temp1$.
3) Next, the substituted output $temp1$ is mixed with the $SRK(i)^{th}$ round key of $RK2$, which is $RK2[SRK(i)]$.
4) Finally, the output is subjected to another substitution operation by using the second substitution table S_2.

The second encrypted block $C[(i+\frac{nb}{2})]$ is obtained by applying the following operations:

1) The $SRK(i+\frac{nb}{2})^{th}$ round key of $RK2$, which is $RK2[SRK(i+\frac{nb}{2})]$ is mixed ("xor") with the $\pi(i+\frac{nb}{2})^{th}$ input plain block $m[\pi(i+\frac{nb}{2})]$.
2) Then, the second substitution table S_2 is used to substitute the corresponding output of the previous step to produce $temp2$.
3) Next, the substituted output $temp2$ is mixed with the $SRK(i+\frac{nb}{2})^{th}$ round key of $RK1$, which is $RK1[SRK(i+\frac{nb}{2})]$.
4) Finally, the output is subjected to another substitution operation by using the first substitution table S_1.

Similarly to the first variant, with the two input blocks version, $IV1$ and $IV2$ are updated using the same operations. The final MAC is obtained by xor-ing the substituted values of $IV1$ and $IV2$.

Algorithm 3 The proposed second variant of the one round one single message authentication encryption algorithm with two input blocks

procedure MAE_2VARIANT_2BLOCKS(M, S_1, S_2, π, RK_1, RK_2, SRK, $IV1$, $IV2$, nb)
  for $i = 1$ to $\frac{nb}{2}$ **do**
    $temp1 = S_1(M[\pi[i]] \oplus RK1[SRK[i]]) \oplus M[\pi[(i+\frac{nb}{2})]]$
    $C[i] = S_2(temp1 \oplus RK2[SRK[i]])$
    $IV1 = S_2(IV2 \oplus temp1)$
    $temp2 = S_2(M[\pi[(i+\frac{nb}{2})]] \oplus RK2[SRK[(i+\frac{nb}{2})]])$
    $C[(i+\frac{nb}{2})] = S_1(temp2 \oplus RK1[SRK[(i+\frac{nb}{2})]])$
    $IV2 = S_1(IV1 \oplus temp2)$
  end for
  $MAC = S_2(IV1) \oplus S_1(IV2))$
end procedure

Algorithm 4 The corresponding authentication-decryption algorithm of the second variant that requires at least two input blocks

procedure MAD_2VARIANT_2BLOCKS(C, S_1, S_2, S_1^{-1}, S_2^{-1}, π, RK_1, RK_2, SRK, $IV1$, $IV2$, nb)
  for $i = 1$ to $\frac{nb}{2}$ **do**
    $temp2 = S_1^{-1}(C[(i+\frac{nb}{2})]) \oplus RK1[SRK[(i+\frac{nb}{2})]]$
    $M[\pi[(i+\frac{nb}{2})]] = S_2^{-1}(temp2) \oplus RK2[SRK[(i+\frac{nb}{2})]]$
    $temp1 = S_2^{-1}(C[i]) \oplus RK2[SRK[i]]$
    $M[\pi[i]] = S_1^{-1}(temp1 \oplus M[\pi[(i+\frac{nb}{2})]]) \oplus RK1[SRK[i]]$
    $IV1 = S_2(IV2 \oplus temp1)$
    $IV2 = S_1(IV1 \oplus temp2)$
  end for
  $MAC = S_2(IV1) \oplus S_1(IV2))$
end procedure

As previously stated, the order of encrypted blocks, in addition to the mixing of the authenticated blocks, is determined by the permutation table π, which is different for each input message.

Note that when a block is to be substituted with an odd index, this is done via the first substitution table S_1, after mixing with $RK1$, and then substituted with S_2, after mixing with $RK2$. While for an even index, this is done via the second substitution table S_2, after mixing with $RK2$, and then substituted via S_1, after mixing with $RK1$.

The decryption algorithm differs only by using the inverse round function RF^{-1} with a reverse order of the round function. Moreover, RF^{-1} uses the inverse substitution tables S_1^{-1} and S_2^{-1}. In Algorithm 4, RF^{-1} is described. The encryption and decryption processes can run in parallel, while the message authentication procedure is done in a chaining mode of operation.

For the authentication and encryption processes, the proposed MAE variants involve the dynamic CBC and the dynamic CTR modes, respectively. The dynamic permutation table is used to select the order of encryption blocks and the order of mixed authenticated blocks. Similarly, a selection table guides the choice of rounds keys. This proposed approach randomizes the link between the encrypted-authenticated blocks, leading to a higher level of randomness, uniformity, and independence, thus, a higher resistance to existing attacks.

*4) **Third Variant of Round Function** RF:* The second variant requires more execution time compared to the first one since it involves more operations to encrypt two input blocks. We present a third variant as the most lightweight MAE variant for two input blocks. It is designed to optimize performance and security level compared to the previous ones.

Algorithm 5 describes the third MAE variant at the source side, and Algorithm 6 shows the corresponding authentication-decryption algorithm at the destination side. The same two input blocks and the same cryptographic primitives described in the second variant are used. Note that only one set of round keys is used in this variant, which is $RK1$ instead of two as in the second variant. Also, only one IV is computed instead of two, and it is based on $temp1$ and $temp2$.

Algorithm 5 The proposed third MAE variant with two input blocks

procedure MAE_3VARIANT_2BLOCKS(M, S_1, S_2, π, RK_1, RK_2, SRK, IV, nb)
  for $i = 1$ to $\frac{nb}{2}$ **do**
    $temp1 = S_1(M[\pi[i]] \oplus RK1[SRK[i]])$
    $C[i] = S_2(temp1)$
    $temp2 = S_2(M[\pi[(i + \frac{nb}{2})]] \oplus temp1)$
    $C[(i + \frac{nb}{2})] = temp2$
    $IV = S_1(IV \oplus temp2) \oplus temp1$
  end for
  $MAC = S_2(IV)$
end procedure

Algorithm 6 The corresponding authentication-decryption algorithm of the proposed third MAE variant that requires at least two input blocks

procedure MAD_3VARIANT_2BLOCKS(C, S_1, S_2, S_1^{-1}, S_2^{-1}, π, RK_1, SRK, IV, nb)
  for $i = 1$ to $\frac{nb}{2}$ **do**
    $temp1 = S_2^{-1}(C[i])$
    $temp2 = C[(i + \frac{nb}{2})]$
    $M[\pi[(i + \frac{nb}{2})]] = S_2^{-1}(temp2) \oplus temp1$
    $M[\pi[i]] = S_1^{-1}(temp1) \oplus RK1[SRK[i]]$
    $IV = S_1(IV \oplus temp2) \oplus temp1$
  end for
  $MAC = S_2(IV)$
end procedure

B. MAE Variants Comparison

The characteristics of the three variants can be summarized as follows:

- The encryption process of all variants can be executed in parallel, while the message authentication process is done in a chaining mode.
- The first variant processes one block at a time, while the second and third variants process two input blocks.
- The second variant requires more execution time than the first variant.
- The third MAE variant exhibits the minimum execution time and resources.
- The second and third variant are associated with a dynamic mixing between processed blocks, which makes the relation between the authenticated blocks more complicated.

Furthermore, the proposed variants have a greater degree of security due to their higher degree of randomness and sensitivity, which is based on the concept of dynamic cryptographic primitives and dynamic input block selection. A short comparison among the proposed MAE variants is presented in Table II, and among the existing MAE solutions in Table III.

VI. SECURITY ANALYSIS

In this section, several tests are performed to assess and confirm the robustness of the proposed MAE variants against analytic attacks. The security level of the proposed MAE variants is evaluated by measuring the randomness, uniformity, and sensitivity, to prove their resistance against statistical, plaintext/selected/known ciphertext, and brute force attacks [46], [34]. The outcome of this section shows that any modern MAE algorithm must have the following properties to be considered immune to attacks:

- To avoid chosen/known plain-text attacks and key-related attacks, the avalanche effect should be fulfilled for both the message and the key.
- To avoid statistical attacks, ciphertexts and their corresponding MAC values should have a high degree of randomness and a uniform distribution.
- Exhaustive search attacks (brute force attacks) are prevented when the secret key (K), the dynamic vector, and the various cryptographic primitives have all a large enough size ($\geq$ 128 bits).
- In addition, the produced MAC should have a length sufficiently large ($\geq$ 128 bits) to avoid birthday attacks.

In terms of randomness and uniformity, the proposed MAE variants are compared to existing MAE algorithms such as CMAC and GMAC. Several properties are evaluated for 1,000 input messages, which follow the normal distribution with a mean of 128, and a standard

Table II: Comparison between the proposed MAE approaches

Metric	First Variant	Second Variant	Third Variant
Error propagation	-	+	+
Parallel Ciphering/deciphering (block level)	Yes	Yes	Yes
Delay	More delay	Higher delay	Lower one
Randomness	+	++	+
Robustness	+	++	+

Table III: Comparison between the proposed MAE variants and existing MAE solutions

MAE algorithms	Number of pass	Number of rounds	Using Dynamic Key	Flexible message block size	Optimized Implementation
CCM	2	$r \geq 10$	No	No (128 bits for AES)	Yes
GCM	2	$r \geq 10$	No	No (128 bits for AES)	Yes
OCB	1	$r \geq 10$	No	No (128 bits for AES)	Yes
ChaCha20-Poly1305	2	20	No	No(512 bits)	Yes
Proposed Ones	1	1	Yes	Yes($N \times 8$ bits)	No

deviation of 16, and $N = 32$. For the security tests, we present the results of the first variant since the second and third variants show similar results.

A. Statistical Analysis

To guard against statistical attacks, the ciphertext and MAC must have high levels of randomness and uniformity. Several statistical tests were introduced in [21], and applied to the generated ciphertexts and MACs. To confirm the levels of uniformity, the Probability Density Function (PDF) and entropy tests are required. As for the randomness and independence properties, we evaluate the recurrence and the correlation coefficient, in addition to the percentage difference between the original and encrypted messages. The levels of randomness and uniformity of the MAC values are evaluated using the same or similar tests.

1) Cipher-text/MAC Uniformity Analysis: The ciphertext must meet the uniformity condition: the occurrence probability of all symbols in the encrypted message should be close to $\frac{1}{n}$, where n denotes the symbols' space, and is equivalent to 256 for one-byte messages. Similarly, the occurrence probability of all symbols in the MAC should be close to $\frac{1}{N}$, where N denotes the length in bytes of the MAC. This could be assessed statistically, or by visualizing the PDF of the encrypted message or the generated MACs.

The PDF of the original messages is not uniform, but the PDF of their corresponding encrypted messages follow the uniform distribution for both MAE variants, and all symbols have a probability of occurrence close to $\frac{1}{256} = 0.039$, as shown in Figure 2, with a random SK.

This result is also validated using the entropy test at the message level, as detailed in [21]. If the entropy value of the encrypted messages, using a random SK, and 256-byte input blocks, is close to 8, then, the message uniformity is satisfied. Figure 3-a) illustrates the PDF results of the entropy values, repeated for 1,000 times. The results show clearly that the encrypted messages always have an entropy close to the desired value of 8. The entropy statistical results are given in Table IV, which shows that the encrypted messages, using the first MAE variant, satisfy the uniformity property.

On the other hand, the MAC values (each of 32 bytes) for 1,000 random session keys, for the three proposed MAE variants are computed. Then, their values are converted to hexadecimal to produce 64 digits for N=32. Figure 2 (c) and (f) show, for the first variant, the recurrence and histogram values. The hex values (0 to 15) are equally distributed, which confirms their high level of randomness and uniformity.

We also ran a test to determine the number of unique bytes within each MAC value. These tests were applied 1,000 times, each time with a new dynamic key and messages. The results for the probability of unique byte elements, for the first MAE variant, as well as for $Tb = 256$, are shown in Figure 3-b).

For the first variant, about 13.4% of the MAC values have 32 unique bytes, approximately 29.5% have 31 unique bytes, and approx 30.1% have 30 unique bytes. For the second and third MAE variants, similar results were obtained, which confirms that the MAC values follow a uniform distribution with a high degree of randomness. Note that increasing the value of N improves the randomness and uniformity degrees.

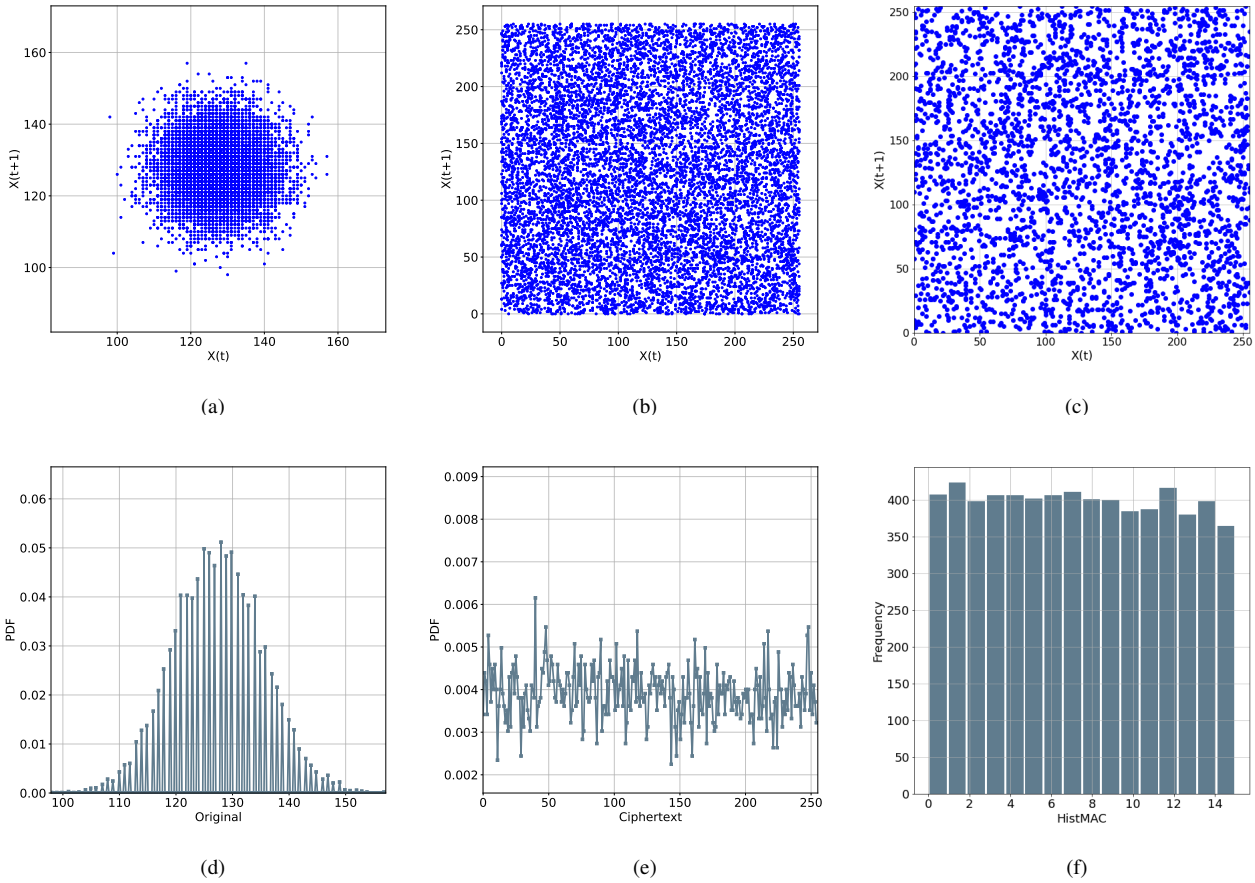


Figure 2: Statistical analysis results: the recurrence of (a) the original message and (b) ciphertext, using the first MAE variant with a random dynamic key $N = 32$. The corresponding PDF of the (d) original message and of the (e) produced ciphertext. The recurrence (c) and histogram distribution (f) of 1,000 MAC values in hexadecimal format for the first MAE variant.

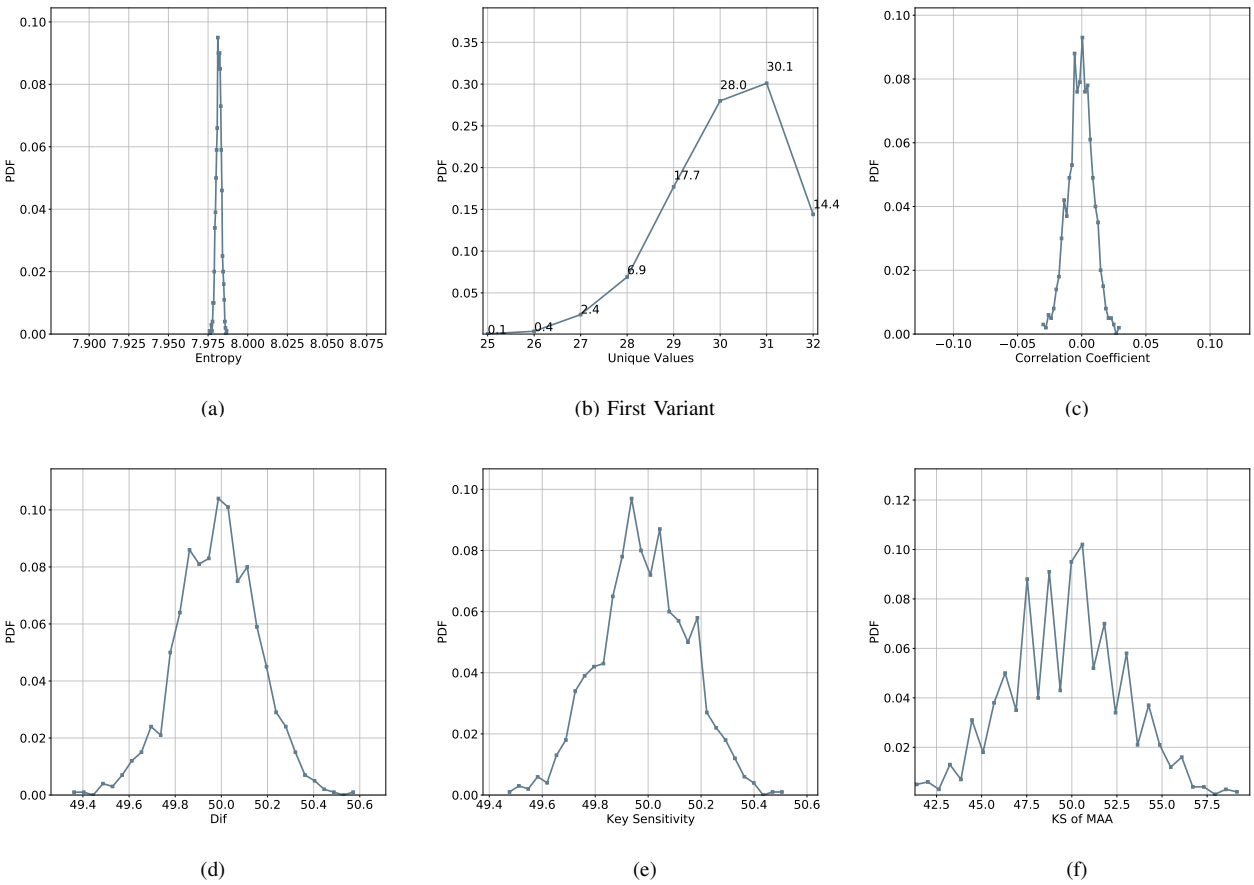
2) Ciphertext Randomness Analysis: In addition to the previous recurrence tests, two additional tests can be performed to evaluate the randomness level of the encrypted messages:

1) The correlation between plaintext and ciphertext
2) The percentage difference between the original and encrypted messages

The correlation between the original and encrypted images were computed for each new input message. The PDF of the correlation coefficient between plaintext and ciphertext, for 1,000 random session keys, are shown in Figure 3-(c) for the first variant, and the obtained values are always close to 0, which confirms the high level of randomness.

The difference test, which evaluates the percentage difference between plaintext and ciphertext, should be close to 50% at the bit level. Figure 3-(d) this difference for the first MAE variants. Table IV provides the numerical statistical results of the correlation and difference tests for the proposed MAE variant. The results confirm that the generated ciphertexts are independent of the original ones.

*3) **Statistical Tests of "TestU01" and "Practrand"** :* Two statistical randomness tests were used to evaluate the randomness of the obtained ciphertext, TestU01 [47] and "Practrand" [48]. The tests included 100 different secret keys, the input plaintext consists of all zeros, and the message length is 2^{17}. The two tests are extremely difficult randomness tests, and the results show that both MAE variants produced ciphertexts that passed all randomness tests of "TestU01" and "Practrand".



(a) (b) First Variant (c)

(d) (e) (f)

Figure 3: Statistical analysis results: (a) PDF of the ciphertext entropy, (b) PDF of the different ASCII characters, (c) PDF of the correlation coefficient, and (d) PDF of the percentage difference between original and encrypted messages; (e) PDF of the key sensitivity (KS) for the first message encryption, and (f) authentication algorithm after changing a random bit of the secret key. All these results are obtained for 1,000 random session secret keys and by using the proposed first variant with $Tb = 256$ bits.

B. Avalanche Effect Analysis

The message and key sensitivity tests are performed to confirm that different ciphertexts and MAC values are produced when a slightly different SK is used, and to validate that the update process produces different cryptographic primitives. The avalanche effect consists of measuring the difference in percentages between the encrypted messages or MACs, for a minor variation in secret key or message. At the bit level, a 50 % difference is required.

1) Key Avalanche effect: This test computes the change in the ciphertexts or MAC values for a 1-bit change in SK. Note that all cryptographic and update primitives are generated from SK. Hence, this one-bit change in SK produces a different dynamic vector DV, and consequently, a different set of cryptographic and update primitives, which eventually leads to different ciphertexts and MAC values. To measure such difference, we compute the Hamming distance between every pair of ciphertexts or MAC values, which basically counts the number of differing bits between two bit streams via the XOR operation. We use the equation:

$$KS_w = \frac{\sum_{i=1}^{Tb} B_i}{Tb} \times 100\% \tag{2}$$

where the Hamming distance KS_w is obtained using the session keys SK_w and $SK_{w'}$. We denote $X_w = MEA_{K_w}(M_w)$ and $X_{w'} = MEA_{K_{w'}}(M_w)$ as the resulting ciphertexts or MAC values using SK_w and $SK_{w'}$, and their binary representations as $BV_w = Bin(X_w)$, $BV_{w'} = Bin(X_{w'})$, and $B = BV_w \oplus BV_{w'}$. The Tb value indicates the number of bits in the MAC value in the case of the message

authentication algorithm. We introduce a 1-bit difference between SK_w and SK'_w in the Least Significant Bit (LSB) of a randomly chosen byte.

Figure 3 (e)-(f) shows the session key sensitivity for 1,000 random keys for both variants. According to the results, the difference between both ciphertexts or MACs, for both variants, is very close to the desired value of 50%, with the majority of the cases = 50%. Table IV shows the numerical statistical results for both MAE variant. The results are similar to the ones with CMAC and HMAC. In conclusion, an attempt to decrypt a message with an incorrect dynamic key, with just one-bit difference, will not provide any useful information about the original message.

2) Message Avalanche Effect: This test computes the change in the ciphertexts or MAC values for a small change in the input message. However, the proposed schemes, by design, require a new SK for a new input message, even if the same message is to be encrypted again. Accordingly, and based on the key sensitivity, the message sensitivity is inherent in the proposed structure.

C. Collision Resistance of the Proposed Approach

For this essential property, we determine the possibility of two separate input data producing the same MAC value. This test was run 1,000 times with 1,000 different session keys, and each time a different bit in the message is changed. Then, for each variant, we compute and compare the MAC values of the original and modified messages. Also, we check the number of identical ASCII characters at the same locations using the following equation:

$$Diff = \sum_{i=1}^{N} D\{MAC(i), MAC'(i)\}, \qquad (3)$$

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

The MAC values of the original and modified messages are represented by MAC and MAC', respectively. Also, $MAC(i)$ corresponds to the ith ASCII character of MAC. The results are shown in Table V, and it can be seen that only three characters at maximum are equal, for Tb = 128 ($N = 16$) and 256 (N =32). This confirms that the proposed MAE variants are resistant to collision and analytic attacks, including birthday attacks, meet-in-the-middle attacks, and differential attacks, according to [49].

VII. Cryptanalysis

In this section, we assess the resistance of the proposed MAE variants against cryptanalysis, mainly analytic attacks as they manifest in the existing confidentiality and authentication attacks.

A. Key Space Analysis

The secret-key space of the proposed MAE variants is sufficiently large; it is equal to 2^{Tb}, with Tb =128, 160, 196, or 256. According to [34], the key space should be greater or equal to 2^{128} to prevent brute force attacks, which is the case with the proposed MAE variants. The secret key can be 2^{128}, 2^{196}, or 2^{256} bits long, and DV has a size of 512 bits. These values are sufficiently large to prevent brute force attacks.

B. Resistance against confidentiality attacks

- **Resistance against statistical attacks**: The use of variable cryptographic primitives and a variable pseudo-random selection of input blocks complicates the task of statistical attacks. This was validated by the high levels of randomness and uniformity, as shown in Figure 2 and Figure 3, respectively. Accordingly, the statistical attacks on the ciphertext or MAC will not return any relevant information.
- **Resistance against linear attacks**: This can be validated based on the following points:
  1) Figure 3 (c)-(d) proved the independence of the original and encrypted messages. Also, the non-linearity between the original and encrypted blocks is enhanced since two different key-dependent substitution tables (S_1 and S_2) are used for each input image.
  2) The relationship between the original and encrypted-authenticated blocks is further enhanced by selecting the input blocks in a pseudo-random manner.
  3) The secret key sensitivity has been satisfied and all cryptographic and update primitives depend on this SK. Moreover, any secure key derivation function can be used to produce DV (512 bits), and to ensure a strong collision resistance. The probability of generating all cryptographic and update primitives is $\frac{1}{2^{512}}$.
- **Resistance against differential attacks**: Launching differential attacks against the proposed variants is very challenging since 1) the SK is changed for each new session, 2) the cryptographic primitives are changed for each input message, and 3) the

Table IV: Statistical results for 1,000 random keys using the proposed first variant of the proposed MAE algorithm.

First variant of the proposed cipher scheme				
	Min	*Mean*	*Max*	*Std*
Dif	49.361	50.01	50.61	0.174
KS	49.47632	50.000763	50.54077	0.168
H_E	7.976	7.98	7.99	0.0016
ρ	-0.0299736	8.479536e-05	0.031	0.01

First variant of the proposed message authentication algorithm				
	Min	*Mean*	*Max*	*Std*
KS	40.625	50.176	59.38	3.036
H_{MAC}	4.578	4.88	5.0	0.082
UV	26	30.15	32	1.27

Table V: Percentage distribution of ASCII characters with the same value at the same location in the MAC value for random LSB bit of secret key when Tb =256

Number of Hits					
Tb \ hits	0	1	2	3	4
128	94.4%	4.9%	0.6%	0.1%	
256	88.3%	11.2%	0.5%	-	

schemes exhibit high levels of randomness, uniformity, and key sensitivity.

- **Resistance against key-related attacks**: The key sensitivity of both MAE variants have been validated according to Figure 3 (e)-(f). As a result, key-related attacks are extremely difficult to be performed.

C. Message Authentication Attacks

1) Pseudo-collision Resistance: In a pseudo-collision attack, the attacker uses vulnerabilities in the compression function to alter the message and its associated MAC value [50], [51]. The proposed variants are immune to such attacks since 1) they are based on variable cryptographic primitives, 2) pseudo-random selection of the input blocks, and 3) the compression function is non-linear and non-commutative.

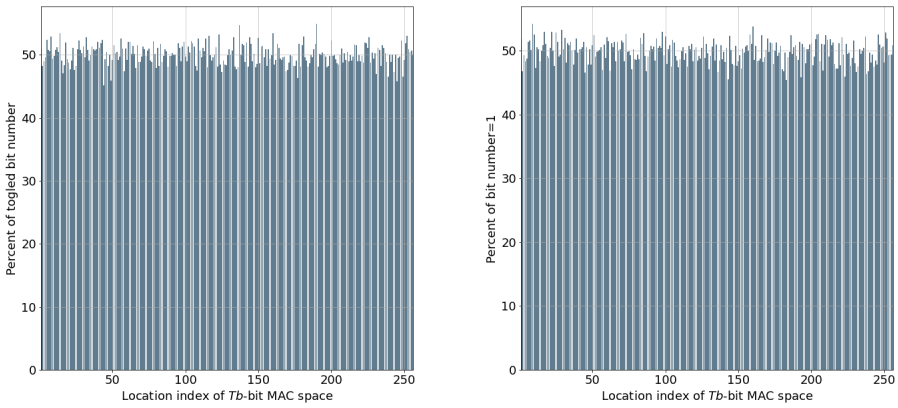
2) Resistance to Birthday Attacks: Birthday attacks are conventional attacks that seek to identify two messages with identical MAC values in less than $2^{\frac{Tb}{2}}$ trials (where Tb is the size of the input block and is also equal to the size of the MAC value) [3]. The smallest MAC size in the proposed MAE variants is 128, which requires 2^{64} trials for a brute force attack. This makes them collision-resistant and thus, immune to birthday attacks.

3) Resistance to Meet-in-the-Middle Attack: Given a message with nb blocks, $M = (m_1, m_2, \ldots, m_{nb-1}, m_{nb})$, the meet-in-the-middle attack aims at finding a block m_i that can replace one of the nb blocks without changing the final MAC value [52], [53]. The MAC values were calculated by replacing a random data block (one bit) with a random m_i. This test was repeated 1,000 times, and two statistical tests were conducted, the difference percentage between both produced MAC values, and the percentage of each bit being equal to 1 versus its corresponding index in the MAC. Figure 4-a) and -c) show that the difference percentage between the MAC values for any modification in any block is at least 50%. Moreover, the percentage of each bit being equal to 0 or 1 for the obtained MACs is also close to 50%, as shown in Figure 4-b) and -d). Note that similar results were obtained for Tb =128 and 512. This confirms that the use of pseudo-random input block selection, and variable cryptographic primitives for every message, renders the proposed MAE variants immune against authentication attacks.

On the other hand, the existing analytic attacks have been designed to attack static cryptographic algorithms with static keys and known cryptographic primitives. Thus, they are unable to break the proposed MAE variants since they are based on the dynamic cryptographic approach.

VIII. Performance Analysis

The performance and effectiveness, of the proposed MAE variants, are evaluated in terms of the computational complexity and execution time, in addition to the influence of error propagation and the required memory consumption.



(a) (b)

Figure 4: Percent of the distribution of changed bit number between $H_n^{''}$ and H versus 1000 tests(a), and percentage of each bit to be equal to 1 versus its corresponding index in MAC (b) for the first MAE variant, respectively.

A. Computational Delay

To compute the total delay, we define the delays associated with the individual components of the proposed schemes:

1) T_{xor}: the time to perform "xor" on two blocks of N bytes;
2) T_{KDF}: the time to generate DV with a size of 64 bytes using a key derivation function;
3) T_{KSA}: the time to generate a substitution table using the KSA of the RC4 algorithm;
4) $T_{MKSA}(x)$: the time to generate a permutation table of x elements by using the modified KSA of RC4;
5) T_{PRNG}: the time to run the Pseudo-Random Number Generation (PRNG) algorithm to generate m round key blocks;

The initial computational delay to construct the cryptographic and update primitives, for each new session, is expressed by CD_{CPC} as expressed by the following equation:

$$CD_{CPC} = T_{KDF} + 3 \times T_{KSA} + 4 \times T_{MKSA}(nb) + T_{PRNG} \quad (4)$$

The permutation and substitution tables can be generated using the KSA and MKSA techniques, which exhibit a low computational delay. Furthermore, any lightweight PRNG can be employed to produce the required round keys ($RK1$ and $RK2$). The required computational delay to update the cryptographic primitives, for each new input message, is given by CD_{UCP}, as expressed by the following equation:

$$CD_{UCP} = T_{KSA} + 2 \times T_{MKSA}(nb) \quad (5)$$

We can see that the update process is faster than the initialization process, and it can be performed in parallel with the encryption-authentication process at IoT devices or application server(s). The following is an evaluation of the delay associated with the proposed MAE authentication process for both variants:

1) T_S: the time it takes to substitute a block of N bytes.
2) T_{Sl}: the time to select an input block from a vector of blocks.

Therefore, in case of the first variant, and for two input blocks for the second and third variants, the required Computational Delays (CD) to encrypt and authenticate a message, for one data block, are represented by:

$$CD_{V1} = 3 \times (T_S + T_{xor} + T_{Sl}) \quad (6)$$
$$CD_{V2} = 6 \times T_S + 7 \times (T_{xor} + T_{Sl}) \quad (7)$$
$$CD_{V3} = 4 \times T_S + 4 \times (T_{xor} + T_{Sl}) \quad (8)$$

The equations show that the second variant requires more operations compared to the first and third variants. Moreover, the third variant requires the minimum number of operations and hence, it introduces the lowest latency. Moreover, the final computational delay to encrypt and authenticate a message of nb blocks for each one of these variants (Equations 6, 7, and 8) should be multiplied only by nb.

Next, we compute the delays associated with the existing MAE schemes that rely on the AES encryption algorithm, CCM, GCM, OCB and Chacha20-poly1305. The total computational delay of AES to encrypt or authenticate one block [54] is expressed as follows:

$$CD_{AES} = r \times T_S + (r+1) \times T_{xor} + (r-1) \times T_D + r \times T_{SR} \quad (9)$$

1) T_D: the time for the AES "mix-column" step;
2) T_{SR}: the time for the AES "shift-rows" step;
3) r represents the number of rounds.

For the minimum number of rounds (r=10) and the minimum key size (128 bits), the execution time to encrypt or authenticate one block is equal to:

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

Accordingly, AES exhibits a larger delay than the proposed MAE variants. This is mainly due to the diffusion operation and the multi-round structure. Thus, CCM, GCM, OCB and Chacha20-poly1305 are slower and have a lower throughput compared to the proposed MAE variants. The results are illustrated in Table VII.

B. Execution Time on Physical Devices

The proposed MAE variants are programmed in Python and C, and their implementations are tested on three different Raspberry Pi devices (RPI-0, RPI-3 and RPI-4). The results are compared to those of CCM, GCM, and OCB, which are based on the optimized version of AES with OpenSSL in addition to Chacha20-poly1305 MAE algorithm. Figure 5-a) shows the throughput (bytes/second) of the proposed MAE variants with a message length of $l = 786432$ bytes versus different sizes of N. The results show that as N increases, the execution time decreases, and consequently, the throughput increases. Furthermore, when both Raspberry devices were considered, the third variant was the fastest for $N \geq 32$, and the first variant was the fastest for $N = 16$. Also, RPI-4 results into the maximum throughput with the lowest latency.

Table VII shows the ratio of throughput of the proposed MAE variants over CCM, GCM, OCB and over Chacha20-poly1305 for different classes of Raspberry Pi devices. The results show the advantage of the proposed MAE variants over the optimized implementations of CCM, GCM, OCB and Chacha20-poly1305. Note that OpenSSL uses optimized assembly instructions to reduce the execution time. Thus, if we make use of hardware optimization for the proposed MAE variants, we could achieve an even higher performance as compared to CCM, GCM,OCB and Chacha20-poly1305.

When comparing the proposed variants against each other, we obtained the results of Figure 6. We can see that the first variant has a higher throughput compared to the second variant. For $N = 16$, the first variant achieves a higher throughput compared to the third one. Also, when compared to the first variant, with RPI-0, the second variant has a reduced throughput by a factor of 30%, while the third variant has a higher throughput by a factor of 13%, for $N = 32$. On RPI-3, and when compared to the first variant, the throughput of the second MAE variant is reduced by a factor of 38%, while the third variant achieves a higher throughput by a factor of 12%, for $N = 32$. On RPI-4, and as compared to the first variant, the throughput of the second MAA variant is reduced by a factor of 35%, while the third variant achieves a higher throughput by a factor of 10%. Regardless of the second variant's overhead delay, all variants outperform CCM, GCM, OCB and Chacha20-poly1305 in terms of performance, as seen in Table VII.

C. Flexibility

The proposed MAE algorithms are designed to process blocks with a variable length, N, which may be changed based on the device restrictions and the application requirements. With powerful devices, the value of N can be increased to achieve a higher resistance to attacks.

D. Memory Consumption

The encryption and authentication processing of one block requires the input block (N bytes), two sets of m round keys ($RK1$ and $RK2$) for ($2 \times N \times m$) bytes, and two permutations tables (π and SRK), each with nb elements. The values of π vary between 0 and nb and the values of SRK vary between 1 and m. We also need two update permutation tables ($\pi 3$ and $\pi 4$) of nb elements, and an update substitution table. Therefore, the required memory consumption to encrypt-authenticate one block, for the first variant or two blocks for the second variant is ($3 \times 256 + 2 \times m \times N$ + $4 \times nb$) bytes. For limited memory devices, we can use only three permutations tables instead of four to reduce the required memory consumption. This can be done by updating π and SRK with the same update permutation table. Moreover, substitution table S_1 can be updated through S_2 and vice versa, S_2 updated through S_1. In such a case, the required memory becomes ($2 \times 256 + 2 \times m \times N$ + $3 \times nb$) bytes. When compared to AES, one substitution table of 256 bytes is required, in addition to the input key, and r round keys. Let us indicate that a higher value

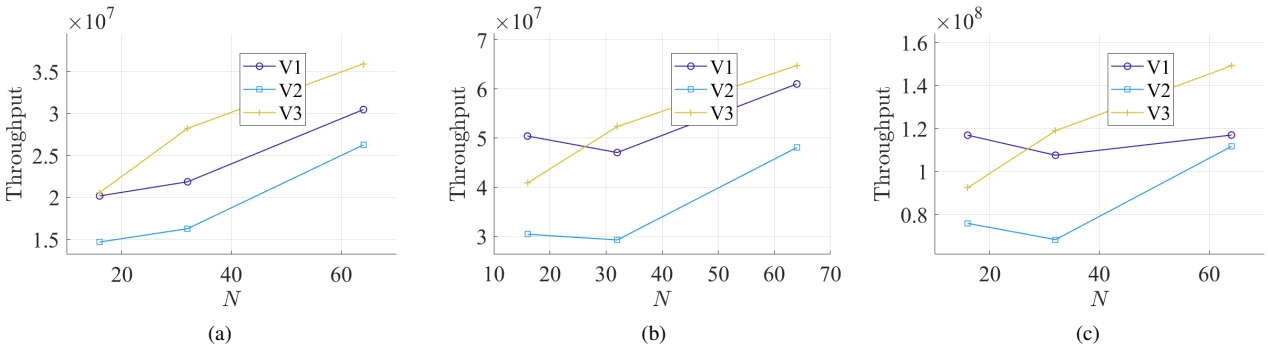


Figure 5: Average throughput in bytes per second for 10,000 times, of the proposed variants for the input message of bytes length versus N ranging from 16 to 64 on RPi-0 (a), RPi-3 (b), and RPI-4 (c).

Table VI: Throughput of the proposed MAE variants versus (CCM, GCM, OCB and Chacha20-poly1305), for $N = 32$.

RPI-Devices	**CCM**	**GCM**	**OCB**	**Chacha20-poly1305**	**V1**	**V2**	**V3**
Raspberry Pi0	5.0179e+06	6.846e+06	1.426e+07	1.422e+07	2.102e+07	1.469e+07	2.37e+07
Raspberry Pi3	1.073e+07	2.799e+07	2.862e+07	2.892e+07	4.753e+07	2.976e+07	5.287e+07
Raspberry Pi4	1.9218e+07	4.209e+07	5.625e+07	5.697e+07	1.08e+08	6.96e+07	1.18e+08

Table VII: Throughput ratio between the proposed MAE variants and (CCM, GCM, OCB and Chacha20-poly1305), for $N = 32$.

Standard MAE	**MAE Variant**	**Raspberry Pi0**	**Raspberry Pi3**	**Raspberry Pi4**
CCM	First	4.19	4.427	5.62
GCM		3.07	1.7	2.567
OCB		1.4741	1.0302	1.662
Chacha20-poly1305		1.4782	1.0331	1.6667
CCM	Second	2.93	2.77	3.624
GCM		2.15	1.2	1.65
OCB		1.6607	1.04	1.847
Chacha20-poly1305		1.6435	1.03	1.83
CCM	Third	4.731	4.93	6.136
GCM		3.47	1.9	2.8
OCB		1.92	1.237	2.1
Chacha20-poly1305		1.89	1.22	2.07

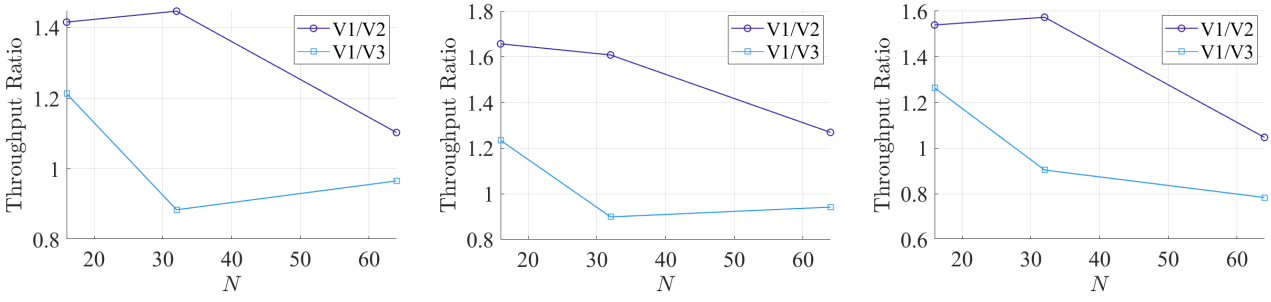


Figure 6: Average throughput ratio for 10,000 times for the first proposed MAE variant with the second and third MAE variants for the input message of bytes length versus N ranging from 16 to 64 on RPi-0 (a), RPi-3 (b), and RPI-4 (c).

of N can be used if the employed devices possess more memory. In the case of limited devices, smaller values of m and N are preferable. Finally, N should be selected to achieve a good balance between performance and security level. The proposed schemes might require more memory compared to standard cryptographic algorithms, but this overhead is acceptable for certain devices and can be reduced by decreasing the values of N and m or by using only one substitution table instead of 2.

IX. Conclusions

Data security is a crucial requirement in IoT networks. Given the large amount of data and the constraints of some IoT devices, it is essential to design lightweight security solutions. In this paper, we propose a new approach for an MAE solution with three variants. The solution consists of two phases, a pseudo-random block selection, and a round function. It is based on the dynamic key-dependent approach such that a dynamic vector is produced for each new session, and used to generate a set of cryptographic and update primitives. The cryptographic primitives are updated for each new input message using the update primitives. The variants use simple cryptographic operations such as "xor", substitution and permutation. The first MAE variant works with one data block at a time, whereas the second and third MAE variants work with two at a time. The existing MAE solutions require two passes, and employ cryptographic algorithms that involve multiple rounds such as CCM, GCM and other ones such as OCB requires only one pass. The proposed approach outperforms these MAE solutions; it is based on a single pass with one round to provide the important security services of data confidentiality, integrity, and source authentication. The statistical test results confirmed that all three variants exhibit high levels of randomness, uniformity, and sensitivity, in addition to strong resistance to collision. The proposed approach achieves an excellent optimization between the security level and system performance. The results of cryptanalysis confirmed that the proposed MAE variants are resistant to different types of attacks. The computational complexity and throughput were assessed, and they were compared to the well-known MAE algorithms, CCM, GCM, OCB and Chacha20-poly1305. The results showed that the proposed MAE variants outperform these two schemes, with lower execution time and fewer required resources. For example, the third MAE variant achieves a low execution time with a reduction of at least 78.85% compared to CCM, 47.91 % compared to GCM, and close to 19 % reduction compared to OCB and Chacha20-poly1305 schemes. This was achieved by making use of common steps for the message encryption and authentication processes. On the other hand, the high security level was ensured through the pseudo-random block ordering. Finally, the characteristics and flexibility of the proposed MAE variants enable them to be adopted in emerging networks and not just within the IoT paradigm. For future work, we will work on the design of an efficient and robust control access scheme for IoT devices.

Acknowledgment

This work has been funded by the EIPHI Graduate School (contract "ANR-17-EURE-0002").

References

[1] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer-Verlag New York Inc, 2010.

[2] J. Katz and Y. Lindell, *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[3] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[4] M. A. Simplicio, B. T. De Oliveira, C. B. Margi, P. S. Barreto, T. C. Carvalho, and M. Näslund, "Survey and comparison of message authentication solutions on wireless sensor networks," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1221–1236, 2013.

[5] J. Song, R. Poovendran, J. Lee, and T. Iwata, "The aes-cmac algorithm," Tech. Rep., 2006.

[6] M. J. Dworkin, "Sp 800-38b. recommendation for block cipher modes of operation: The cmac mode for authentication," 2005.

[7] M. Dworkin, "NIST SP 800-38A: recommendation for block cipher modes of operation," *National Institute of Standards and Technology (NIST), USA*, 2001.

[8] K. Paterson, J. Foley, and D. McGrew, "Authenticated encryption with aes-cbc and hmac-sha," 2014.

[9] J. M. Turner, "The keyed-hash message authentication code (hmac)," *Federal Information Processing Standards Publication*, vol. 198, no. 1, 2008.

[10] D. McGrew and K. Igoe, "Aes-gcm authenticated encryption in secure rtp (srtp)," *draft-ietf-avtcore-srtp-aes-gcm-15 (work in progress)*, 2015.

[11] W. Stallings, "The offset codebook (ocb) block cipher mode of operation for authenticated encryption," *Cryptologia*, vol. 42, no. 2, pp. 135–145, 2018.

[12] F. De Santis, A. Schauer, and G. Sigl, "Chacha20-poly1305 authenticated encryption for high-speed embedded iot applications," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 692–697.

[13] P. Szalachowski, B. Ksiezopolski, and Z. Kotulski, "Cmac, ccm and gcm/gmac: Advanced modes of operation of symmetric block ciphers in wireless sensor networks," *Information Processing Letters*, vol. 110, no. 7, pp. 247–251, 2010.

[14] N. Doraswamy and D. Harkins, *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice Hall Professional, 2003.

[15] D. Maimut and R. Reyhanitabar, "Authenticated encryption: Toward next-generation algorithms," *Security & Privacy, IEEE*, vol. 12, no. 2, pp. 70–72, 2014.

[16] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, "Report on lightweight cryptography," *NIST DRAFT NISTIR*, vol. 8114, 2016.

[17] A. Y. Poschmann, "Lightweight cryptography: cryptographic engineering for a pervasive world," in *PH. D. THESIS*. Citeseer, 2009.

[18] H. N. Noura, L. Sleem, M. Noura, M. M. Mansour, A. Chehab, and R. Couturier, "A new efficient lightweight and secure image cipher scheme," *Multimedia Tools and Applications*, Sep 2017.

[19] H. N. Noura and D. Courousse, "Method of encryption with dynamic diffusion and confusion layers," Jun. 9 2016, wO Patent App. PCT/EP2015/078,372. [Online]. Available: https://www.google.com/patents/WO2016087520A1?cl=en

[20] H. N. Noura, A. Chehab, M. Noura, R. Couturier, and M. M. Mansour, "Lightweight, dynamic and efficient image encryption scheme," *Multimedia Tools and Applications*, pp. 1–35, 2018.

[21] H. Noura, A. Chehab, L. Sleem, M. Noura, R. Couturier, and M. M. Mansour, "One round cipher algorithm for multimedia IoT devices," *Multimedia tools and applications*, vol. 77, no. 14, pp. 18 383–18 413, 2018.

[22] H. N. Noura, M. Noura, A. Chehab, M. M. Mansour, and R. Couturier, "Efficient and secure cipher scheme for multimedia contents," *Multimedia Tools and Applications*, pp. 1–30, 2018.

[23] R. Melki, H. N. Noura, M. M. Mansour, and A. Chehab, "An efficient ofdm-based encryption scheme using a dynamic key approach," *IEEE Internet of Things Journal*, 2018.

[24] H. Soroush, P. M. Irey IV, G. Pardo-Castellote, and S. Canup, "Next-generation cytbersecurity for advanced real-time distributed systems," 2015.

[25] T. Mizoguchi and Y. Ito, "Effect of qos degradation caused by 6to4 and ipsec on qoe for web services," in *Consumer Electronics (GCCE), 2014 IEEE 3rd Global Conference on*. IEEE, 2014, pp. 5–9.

[26] A. Nieto and J. Lopez, "Analysis and taxonomy of security/qos tradeoff solutions for the future internet," *Security and Communication Networks*, vol. 7, no. 12, pp. 2778–2803, 2014.

[27] H. N. Noura, A. Chehab, and R. Couturier, "Efficient & secure cipher scheme with dynamic key-dependent mode of operation," *Signal Processing: Image Communication*, vol. 78, pp. 448 – 464, 2019.

[28] H. N. Noura, O. Salman, R. Couturier, and A. Chehab, "Novel one round message authentication scheme for constrained IoT devices," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–17, 2021.

[29] Z. Fawaz, H. N. Noura, and A. Mostefaoui, "An efficient and secure cipher scheme for images confidentiality preservation," *Signal Processing: Image Communication*, vol. 42, pp. 90–108, 2016.

[30] P. Zhang, Y. Jiang, C. Lin, Y. Fan, and X. Shen, "P-coding: secure network coding against eavesdropping attacks," in *INFOCOM, 2010 Proceedings IEEE*. IEEE, 2010, pp. 1–9.

[31] L. Pradeep and A. Bhattacharjya, "Random key and key dependent s-box generation for AES cipher to overcome known attacks," in *International Symposium on Security in Computing and Communication*. Springer, 2013, pp. 63–69.

[32] I. Damgård, "A design principle for hash functions," in *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '89. London, UK, UK: Springer-Verlag, 1990, pp. 416–427.

[33] R. C. Merkle, "A certified digital signature," in *Proceedings on Advances in cryptology*, ser. CRYPTO '89. New York, NY, USA: Springer-Verlag New York, Inc., 1989, pp. 218–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=118209.118230

[34] W. Stallings, *Cryptography and network security: principles and practice*. Pearson Upper Saddle River, 2017.

[35] A. Langley, W. Chang, N. Mavrogiannopoulos, J. Strombergson, and S. Josefsson, "ChaCha20-Poly1305 cipher suites for transport layer security (tls)," *RFC 7905*, no. 10, 2016.

[36] J. A. Yaacoub, M. Noura, H. N. Noura, O. Salman, E. Yaacoub, R. Couturier, and A. Chehab, "Securing internet of medical things systems: Limitations, issues and recommendations," *Future Gener. Comput. Syst.*, vol. 105, pp. 581–606, 2020. [Online]. Available: https://doi.org/10.1016/j.future.2019.12.028

[37] J. A. Yaacoub, O. Salman, H. N. Noura, N. Kaaniche, A. Chehab, and M. Malli, "Cyber-physical systems security: Limitations, issues and future trends," *Microprocess. Microsystems*, vol. 77, p. 103201, 2020. [Online]. Available: https://doi.org/10.1016/j.micpro.2020.103201

[38] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[39] J. Wei, X. Chen, X. Huang, X. Hu, and W. Susilo, "RS-HABE: Revocable-storage and hierarchical attribute-based access scheme for secure sharing of e-health records in public cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2301–2315, 2021.

[40] D. Wang, H. Cheng, D. He, and P. Wang, "On the challenges in designing identity-based privacy-preserving authentication schemes for mobile devices," *IEEE Systems Journal*, vol. 12, no. 1, pp. 916–925, 2018.

[41] Y. Chen, H. Wen, H. Song, S. Chen, F. Xie, Q. Yang, and L. Hu, "Lightweight one-time password authentication scheme based on radio-frequency fingerprinting," *IET Communications*, vol. 12, no. 12, pp. 1477–1484, 2018.

[42] S. Chen, H. Wen, J. Wu, A. Xu, Y. Jiang, H. Song, and Y. Chen, "Radio frequency fingerprint-based intelligent mobile edge computing for internet of things authentication," *Sensors*, vol. 19, no. 16, p. 3610, 2019.

[43] F. Xie, H. Wen, Y. Li, S. Chen, L. Hu, Y. Chen, and H. Song, "Optimized coherent integration-based radio frequency fingerprinting in internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3967–3977, 2018.

[44] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[45] C. Wang, D. Wang, G. Xu, and D. He, "Efficient privacy-preserving user authentication scheme with forward secrecy for industry 4.0," *Science China Information Sciences*, vol. 65, no. 1, pp. 1–15, 2022.

[46] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[47] P. L'Ecuyer and R. J. Simard, "Testu01: A c library for empirical testing of random number generators," *ACM Trans. Math. Softw*, vol. 33, no. 4, pp. 22:1–22:40, 2007. [Online]. Available: http://doi.acm.org/10.1145/1268776.1268777

[48] J. D.cook, "Testing RNGs with practrand: Xoroshiro, xorshift, mt, pcg," https://www.johndcook.com/blog/2017/08/14/testing-rngs-with-practrand/, August 2017.

[49] X. Wang and H. Yu, "How to break md5 and other hash functions," in *In EUROCRYPT*. Springer-Verlag, 2005.

[50] A. Akhavan, A. Samsudin, and A. Akhshani, "A novel parallel hash function based on 3d chaotic map," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, no. 1, pp. 1–12, 2013.

[51] 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*, Oct 2009, pp. 1–7.

[52] M. Amin, O. S. Faragallah, and A. A. A. El-Latif, "Chaos-based hash function (cbhf) for cryptographic applications," *Chaos, Solitons & Fractals*, vol. 42, no. 2, pp. 767–772, 2009.

[53] A. Kanso and M. Ghebleh, "A structure-based chaotic hashing scheme," *Nonlinear Dynamics*, vol. 81, no. 1-2, pp. 27–40, 2015.

[54] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.