# Efficient and Secure Keyed Hash Function Scheme Based on RC4 Stream Cipher

Hassan Noura[1], Ola Salman[2], Ali Chehab[2], and Raphaël Couturier[3]

[1]Arab Open University, Department of Computer Studies, Beirut, Lebanon
[2]Dept. of Electrical and Computer Engineering, American University of Beirut, Beirut 1107 2020, Lebanon
[3]FEMTO-ST Institute, Univ. Bourgogne Franche-Comté, Belfort, France

***Abstract*—High number of rounds is needed for the existing message authentication algorithms, such as keyed hash functions like Hash-based Message Authentication Code (HMAC) or block cipher based functions like Cipher-based Message Authentication Code (CMAC) and Galois Message Authentication Code (GMAC). Moreover, the employed compression functions consist of several operations to achieve two main properties: confusion and diffusion. This large number of rounds introduces high overhead for resource-limited systems like Internet of Things (IoT) or delay-sensitive systems that have real-time requirements like Intelligent Transparent Systems. In this paper, a new lightweight message authentication algorithm is proposed to reduce the number of rounds to one. The proposed compression function is based on the RC4 stream cipher to reduce the required overhead in terms of latency and resources. Finally, the security and performance analysis shows that the proposed keyed hash function is resistant towards existing security attacks with low resources overhead.**

***Index Terms*—Lightweight message authentication algorithm; RC4; Dynamic key dependent cryptographic primitives.**

## I. Introduction

Existing Message Authentication Algorithms (MAAs) can be based on keyed hash functions, such as HMAC [1], that can use any conventional un-keyed hash function, or a robust block cipher such as Advanced Encryption Standard (AES) [2] with Cipher Block Chaining (CBC) mode, such as Cipher-based Message Authentication Code (CMAC) [3] and Galois Message Authentication Code (GMAC) [4]. AES can be used with different key sizes: 128, 192 and 256 bits. The key size defines the number of iterations $r$ = 10, 12, 14 iterations for a 128, 196, 256-bits key, respectively. Besides, existing block cipher based authentication algorithms are faster compared to those based on a hash function. Moreover, let us indicate that the message authentication process follows the CBC operation mode, which cannot provide the parallelism.

However, the existing MAAs are based on a round function structure requiring a large number of rounds $r$ to ensure the desirable cryptographic properties. Besides, this round function consists of several confusion and diffusion operations. Unfortunately, the modern applications present constraints in terms of latency and resources, being deployed on constrained devices. Thus, the new MAAs should meet the trade-off between security and efficiency.

The main goal of this work is to present a simple and secure MAA that it is based on the proposed one-round compression function. Therefore, in this paper, we design a lightweight keyed hash function that overcomes the previously stated challenges by introducing a new flexible dynamic key-dependent one round compression function. It is based on the RC4 Key setup algorithm and pseudo-random number generator. The proposed solution can ensure the desirable cryptographic performance such as message and key avalanche effect, resistance against collision with minimum computation and complexity. Consequently, the proposed solution can be applied for different types of applications and networks.

The rest of this paper is organized as follows. In section II, we review the related work to message authentication. In Section III, the description of the proposed MAA solution is presented. Section IV studies the strength of the proposed MAA solution concerning several desirable cryptographic properties. Then, performance analysis is provided in Section V. Finally, Section VI concludes the paper.

## II. Related Work

Different solutions have been proposed in the literature to ensure data integrity based on unkeyed hash functions, which are variations of the Secure

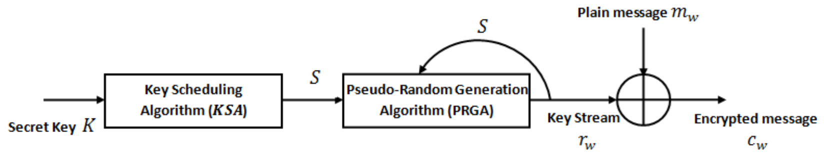

Figure 1: RC4 Algorithm

Hash Function (SHA), such as SHA-2 [5], SHA-3 [6], BLAKE [7], Grostl [8], Skein [9], and Keccak [6] which became SHA-3.

Concerning source authentication, it can be ensured by using a digital signature or symmetric Message Authentication Algorithm (MAA) that can be one of two classes:

1) Keyed hash functions such as Hash-based Message Authentication Code (HMAC) [10], which employ any conventional un-keyed secure hash function (SHA variants).
2) Block cipher such as the Advanced Encryption Standard (AES) [2] with Cipher Block Chaining (CBC) mode such as the Cipher-based Message Authentication Code (CMAC) [3], and Galois Message Authentication Code (GMAC) [4] in authentication operation mode.

CMAC, for example, uses AES with keys of size 128, 192 or 256 bits, and a corresponding number of rounds of 10, 12, or 14, respectively. Also, HMAC uses variants of the SHA algorithm, which requires a high number of rounds (80 rounds for SHA-1, SHA-512 and 24 for SHA-3) [11].

Typically, the existing symmetric message authentication schemes are based on iterating a compression function for a large number of rounds, $r$, to reach the desired cryptographic properties. Additionally, this function consists of multiple operations to ensure the confusion and diffusion properties, which introduces a large overhead in terms of resources and delay. Thus, there is a need for a lightweight cryptographic algorithm with a high security level to speed up data processing and to reduce the execution time (latency) and required resources [12].

To address this issue, different solutions were proposed to construct secure and efficient hash functions. One type of solutions relies on the ”Chaos” theory, which is based on a non-linear dynamic system. However, the chaos-based hash functions are not practical since they are based on a non-integer and non-linear transformation, which requires a high computational complexity, limiting their benefits in many use cases [13].

Alternatively, a new class of lightweight cryptographic algorithms has emerged and it is based on reducing either the number and complexity of operations or the number of rounds. For example, in [14], [15], [16], cipher schemes based on two rounds were proposed, while in [17], [18], a more efficient solution with only one round was presented. To maintain high security, these solutions rely on the dynamic key approach [14], [15], [16]. This paper follows this logic and presents a novel one-round MAA to ensure data integrity with source authentication.

## III. The Proposed Hash Function

In this section, the proposed MAA and its corresponding compression function (totally dependent on RC4 functions) are described. In fact, RC4 involves a Key Setup Algorithm (KSA) and a Pseudo-Random Number Generator Algorithm (PRNGA), which are to be implemented sequentially. The key setup of RC4 is used to produce a substitution table ($S \triangleq \{s[0], \cdots, s[L_S - 1]\}$ with $L_S = 256$ elements varying from $s[0]$ to $s[255]$), which represents the output of KSA step. Then, this substitution table is used as input in the PRNGA step to produce the required keystream. The RC4 steps are illustrated in Figure 1. In addition, the RC4 algorithm has a variable key length, which ranges between 64 and 256 bits.

The selection of RC4 is based on its simplicity, which means less computation and resources overhead requirement. However, RC4 suffers from different security issues as an encryption algorithm and it is considered as a weak cipher. In contrast, in this paper, RC4 is used to construct an efficient MAA (compression function), in addition to the use of a dynamic key dependent structure (dynamic key) instead of a static one.

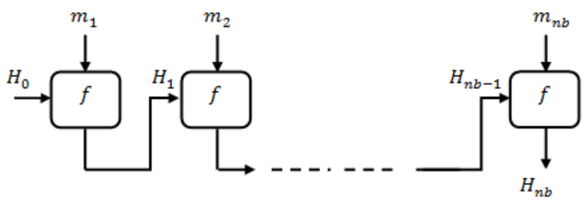

Figure 2: MAC Process Generation ($f$ represents the compression function)

## A. *Proposed Compression Function*

The proposed MAC process($hDK$) uses the Merkle and Damgrad principle (MD), which is illustrated in Figure 2. This principle consists of a compression function $f$ that is applied in an iterative block process. First, the input message $M$ of finite length $|M|$ is padded if necessary to ensure that the length of $M$ is multiple of $Tb$, where $Tb$ represents the block length and it can be equal to 128, 256, 512, 1024). $M$ is subsequently divided into $nb$ blocks ($m_1,\ m_2,\ \ldots,\ m_{nb}$), where $nb \geq 1$. Then, each block $m_j,\ 1 \geq j \geq nb$ and a chaining block $H_{j-1}$ are taken as inputs to the proposed function. Furthermore, the initial vector ($IV$) can be constructed from several application-dependent parameters such as nonce, counter, and identity. Also, $IV$ might require padding to be a $Tb$-bit complete block. The MAC value will then be calculated according to the following equation:

$$H_j = h(H_{j-1}, m_j,) \qquad j = 2, 3, \ldots, nb \qquad (1)$$

where $H_0 = K \oplus IV$ and $K$ represents the authentication session key. The last output $H_{nb}$ can be exploited directly as MAC. Additionally, every $m_i$, $K$ and $IV$ is divided to $nBytes = \frac{Tb}{8}$ blocks of 8-bits (bytes) as input.

Figure 3 illustrates the proposed compression function, which consists of four steps:

1) Mixing the input block $m_j$ with its previous compressed block $H_{j-1}$ to produce the seed $X$.
2) Iterating the KSA algorithm of RC4 with initial state table ($S_o$), $X$ as a seed for each input block to produce a substitution state table $S$.
3) Iterating the PRGA algorithm of RC4 with the produced $S$ to derive a block of keystream $RK$.
4) $RK$ is substituted by using $S$ to obtain the $j^{th}$ compressed block $H_j$.
5) Initial state table ($S_o$) is updated for the next block (KSA iteration) and it will be equal to $S$.

The pseudo-code of the proposed compression function, *compression_Function*, is described in Algorithm 1.

**Algorithm 1** The Proposed RC4 Compression Function

```
1: procedure PROPOSEDCF ($H_{j-1}$, $m_j$, $Tb$)
2:     $X \leftarrow (H_{j-1} \oplus m_j)$
3:     $S \leftarrow KSA_{RC4}(X,\ S_0)$
4:     $[RK,\ S] \leftarrow PRGA_{RC4}(S,\ \frac{Tb}{8})$;
5:     $H_j \leftarrow Substitution(RK,\ S)$
6: end procedure
```

As a summary, the mixing between the input data block and secret key forms a substitution table, which is updated after producing the required keystream block. Then, this keystream block is substituted by using the update substitution table to form the output block at the last step of the proposed compression function. Moreover, the initial substitution state table is updated after each iteration. Consequently, the proposed algorithm is very simple and ensures a high-security level since the required cryptographic primitives are modified in each block and depend on its input block, which helps to ensure message and key avalanche effect.

## IV. Security Analysis

Several security tests are included in this section to evaluate and to prove the robustness of the proposed MAA. Therefore, in this section, randomness, uniformity and sensitivity tests are applied to validate that the proposed MAA reaches the desired cryptographic properties.

### A. *Uniformity Tests*

In this part, two different tests were applied to validate the randomness distribution of MAC value and its uniformity.

*1) MAC value distribution:* The security of any MAA is strongly related to the uniform distribution of the MAC value. To verify the uniformity of the MAC value with respect to the original text, a simulation of input message in ASCII code is performed using the abstract paragraph of this article. The distribution of the original paragraph, as depicted in Figure 4-a), is distributed in the range of ASCII codes, and its corresponding MAC value distribution is spread out randomly (see in Figure 4-b)). Similarly, another test has been performed on an input message consisting of a string of zeros. The

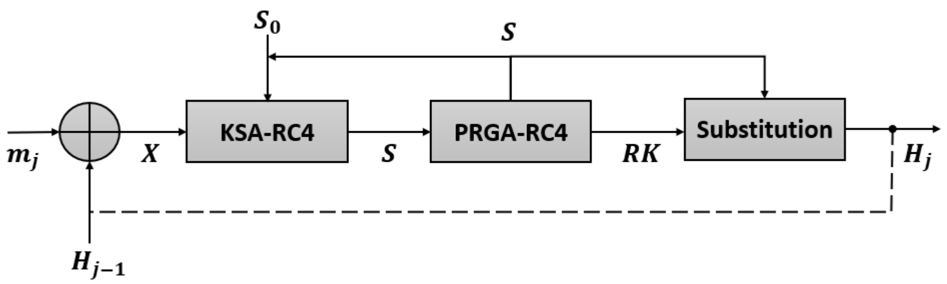

Figure 3: Proposed Compression Function

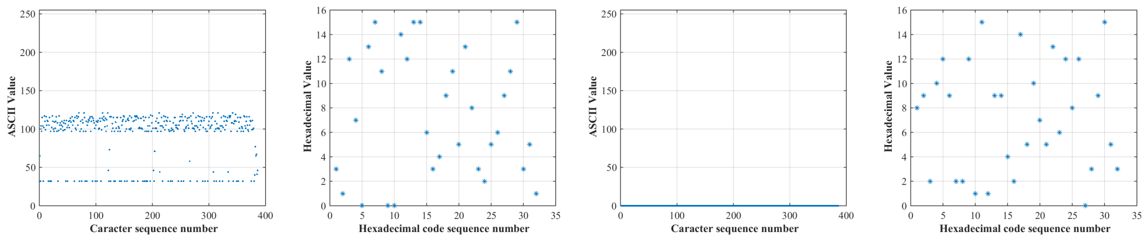

Figure 4: Spread of message and hash value: Distribution of a random(a) and zeros(b) message in ASCII code; Distribution of the MAC value in hexadecimal format of the random(c) and zeros(d) message.

results, as illustrated in Figure 4-c), show that even in this special condition the output MAC value still has a random distribution (see in Figure 4-d)). These results indicate clearly that MAC values are randomly uniform distributed.

*2) Unique Test:* In order to check the uniformity of the obtained MAC value, another test is applied by simply computing the length of unique elements of the obtained MAC value for $Tb$=128 and 256 respectively. Table I and II present the corresponding percentages of unique elements for 10000 MAC values, where each value is obtained from a random secret key and message.

Table I: Percent of the different number of ASCII characters for N=10000 with $Tb = 128$ (16 bytes)

| Diff ASCII | 16 | 15 | 14 | 13 | 12 |
|---|---|---|---|---|---|
| Percent | 62.22 | 30.75 | 6.42 | 0.59 | 0.02 |

According to these results, the distribution of the percentages of unique elements verifies its uniformity since $\approx 92.312\%$ of MAC values have at least 15 different elements for $Tb = 128$, which indicates that strong uniformity is obtained. Better results of the percentages of unique elements are obtained for $Tb \geq 256$. Increasing the MAC space will increase uniformity, randomness and collision level.

Table II: Percent of the different number of ASCII characters for N=10000 with $Tb = 256$ (32 bytes)

| Diff ASCII | 32 | 31 | 30 | 29 | 28 | 27 |
|---|---|---|---|---|---|---|
| Percent | 13.43 | 28.22 | 30.2 | 17.85 | 7.38 | 2.28 |

| Diff ASCII | 26 | 25 | 24 |
|---|---|---|---|
| Percent | 0.56 | 0.05 | 0.03 |

## B. Key Avalanche Effect

Sensitivity refers to a huge change in the MAC value with respect to a slight change in the secret key $K$, the initial vector $IV$ or the original message itself. Any efficient MAA $h_K$ is considered robust against related key attacks if it ensures the sensitivity of $K$ and $IV$. In particular, when the message is authenticated, any small change in $K$ or $IV$ should give two completely different MAC values. The sensitivity of $K$ and $IV$ are analyzed (1000 random keys and initial vectors) by measuring the percentage of Hamming distance (difference) that can be calculated as follows:

$$KS_w = \frac{\sum_{k=1}^{Tb} h_{K_w}(M) \oplus h_{K'_w}(M)}{Tb} \times 100\% \quad (2)$$

where $Tb$ is the length of the MAC value in bit, and $h_{K_w}(M)$ and $h_{K'_w}(M)$ are the corresponding MAC values using $K_w$ and $K'_w$, respectively. All the elements of $K'_w$ are equal to those of the $w^{th}$ key $K_w$, except for the Least Significant Bit ($LSB$) of a random byte, which is flipped. Indeed, the same processing is realized for measuring the sensitivity of $IV$, which gives a similar result, since $K$ and $IV$ are mixed together to form $H_0$.

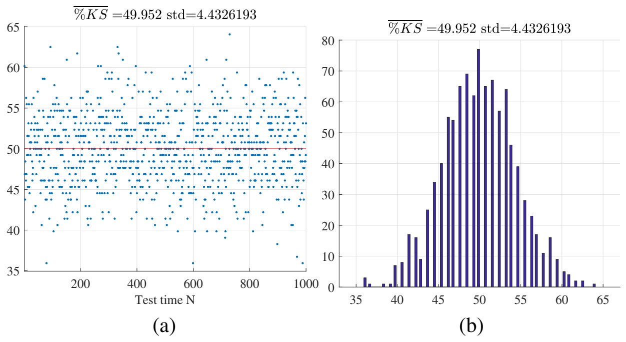

Figure 5: Changed random bit of the secret key; Percent of the changed bits (1000 times) and its corresponding distribution (b), respectively

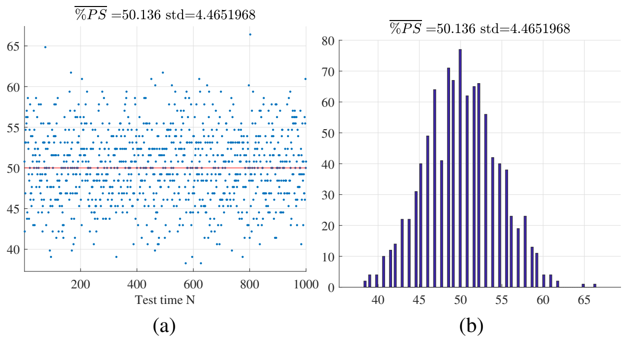

Figure 6: Percent of the changed bits number versus 10000 random secret keys; changed random bit of the the message (a) and its corresponding distribution (b), respectively

## C. Message Avalanche Effect

Similarly, the original message sensitivity is performed and calculated as follows:

$$PS_w = \frac{\sum_{k=1}^{Tb} h_{K_w}(M) \oplus h_{K_w}(M')}{T} \times 100\% \quad (3)$$

where all the elements of $M'$ are equal to those of $M$, except for a random Least Significant Bit ($LSB$), which is flipped and $w = 1,\ 2,\ \ldots,\ 1000$.

In Figure 5 and Figure 6, the results of the sensitivity test of the secret key and original message versus 1000 random keys and messages are shown, respectively, having only a $LSB$ of a random byte changed of the secret key $K_i$ or $M$. The majority of samples are closer to the optimal value in bit level (50%). Additionally, $\approx$ 88% of the samples have $KS$ and $PS \geq 45\%$.

Another example was done to prove the message avalanche effect of the proposed MAA. The MAC of the message with the following conditions using a random session key is computed:

- C1: The original paragraph (abstract of this paper);
- C2: Replacing the first character A from the original paragraph with B;
- C3: Modifying the word HMAC in the original paragraph to CMAC;
- C4: Replacing the full stop from the original paragraph with comma;
- C5: Adding a blank space to the original paragraph.

The corresponding percentages of changed bits are presented in Table IV among the obtained MACs. These results clearly indicate that a very small change in the original message produces an enormous change in the corresponding MAC value.

Finally, $KS$ and $PS$ follow a normal distribution. Their minimum, maximum, average and standard deviation are presented in Table III. Similarly, the same result is obtained when changing a single bit in $IV$. Therefore, the proposed MAA ensures a high level of message and key avalanche effects, which is one of the main cryptographic proprieties.

## D. Collision resistance

Collision resistance refers to the difficulty of finding two distinct inputs to the MAC values whose outputs are the same. Generally, the resistance against collision is verified using a test conducted for a MAC value randomly generated from a paragraph of the message and stored in ASCII format. A bit will be randomly selected from the chosen paragraph and flipped; the output MAC value of the modified message will be also stored in ASCII format. A comparison between the two MAC values is achieved by counting the number of identical positions of ASCII characters (i.e. having the same value in the same location) as follows:

$$Diff = \sum_{i=1}^{n} D\{H(i), H'(i)\}, \quad (4)$$

Table III: Statistical Results of $PS$ and $KS$

(a)

**Statistical Results of $PS$**

| **Tb** | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|
| **min** | 33.59 | 37.11 | 42.38 | 43.16 |
| **max** | 66.406 | 62.5 | 59.57 | 56.83 |
| **Avg** | 49.98 | 49.97 | 49.98 | 49.98 |
| **STD** | 4.428 | 3.15 | 2.2164 | 1.6123 |

(b)

**Statistical Results of $KS$**

| **Tb** | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|
| **min** | 32.03 | 38.67 | 41.99 | 42.28 |
| **max** | 69.53 | 61.72 | 57.42 | 56.15 |
| **Avg** | 49.97 | 49.97 | 49.98 | 49.98 |
| **STD** | 4.4 | 3.12 | 2.22 | 1.553 |

| Case | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|
| C1 | 0 | 59.3750 | 53.1250 | 49.8438 | 48.4375 |
| C2 | 59.3750 | 0 | 48.4375 | 55.4688 | 48.4375 |
| C3 | 53.1250 | 48.4375 | 0 | 42.9688 | 57.8125 |
| C4 | 49.8438 | 55.4688 | 42.9688 | 0 | 46.0938 |
| C5 | 48.4375 | 48.4375 | 57.8125 | 46.0938 | 0 |

Table IV: Distribution of changed bit number under different conditions

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

Simulation results, presented in Table V, indicate that the maximum number of equal characters (hits) is 3 for block lengths of 128 and 256, and 4 for block lengths of 512 and 1024. Consequently, collision resistance is ensured, which makes our proposition immune to birthday, meet-in-the-middle and differential attacks [19].

Additionally, it is obvious from the obtained results that the uniformity of the obtained MAC value and the independence based on the percentage of hamming distance between both hash values is close to 50% for the different values of $Tb$. This demonstrates the collision resistance nature of the proposed algorithm.

The security level of the proposed MAA relies on the use of the dynamic key dependent cryptographic primitives that preserves the unpredictability and high sensitivity of message, secret key and initial vector. Moreover, the statistical properties of the proposed MAA (such as the uniformity of the produced MAC value, key sensitivity and the avalanche effect) are achieved as described in this section. This provides immunity against statistical attacks.

In the following, we present the cryptanalysis of the proposed hash function to prove its robustness and demonstrate that it can be a good and lightweight candidate to ensure data integrity and source authentication.

### *E. Key space analysis*

The size of the secret key of the proposed MAA depends on $Tb$. So, the key space of the secret key is $2^{Tb}$, where $Tb$ =128, 160, 196, 256, 512, 1024 bits. From a security viewpoint, the key space for a MAA should not be less than $2^{128}$ in order to resist brute force attacks [1]. Therefore, the proposed flexible scheme is sufficiently large to make the brute-force attack infeasible.

### *F. Resistance to birthday attack*

The birthday attack is one of the common classic attacks on cryptographic hash functions, which can be applied on any algorithm. The main goal of this attack is to find two messages with identical MAC values with less than $2^{\frac{Tb}{2}}$ trials ($Tb$ is the size of hash value) [20]. In fact, the proposed MAA algorithm is flexible so that the length of the MAC value can be increased. If the MAC value size is set to 128, the difficulty of the attack is $2^{64}$. By increasing $Tb$, the proposed MAA becomes more resistance against brute force attack.

## V. Performance Analysis: Computation Complexity

The main objective of the proposed MAA is to achieve a high security level with the minimum possible number of operations and rounds. This requires reducing the computational complexity, encryption/decryption time and resources (especially energy) for the data confidentiality process. The execution time of the proposed MAA is presented and quantified. To assess the total associated overheads, we quantify several delays as follows:

1) $T_S$ denotes the required substitution execution time for a block of $nBytes$ bytes.
2) $T_{xor}$ denotes the required "XOR" execution time between two blocks of $nByte$ bytes.
3) $T_{PRNG}$ denotes the required time to produce $nBytes$ by iterating the employed RC4-PRNG.
4) $T_{KSA}$ denotes the required time to iterate KSA with a seed of $nBytes$ length.

Therefore, the total Computational Delay ($CD$) of the proposed scheme to encrypt one block is:

$$CD = T_{xor} + T_{KSA} + T_{PRNG} + T_S \qquad (5)$$

Besides, the delay of the XOR, substitution and PRNG operations are less than that of the "KSA" of RC4.

Table V: 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

| $Tb$ \ hits | 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) Message

| $Tb$ \ hits | 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 |

## VI. Conclusions

In this paper, we propose a new MAA solution that it is based on a dynamic, lightweight, flexible compression function. The compression function is based on the simple RC4 stream cipher that is iterated only for one round for each block. Moreover, the cryptographic primitives are related to the input message and a secret key, initial vector in addition to the current and previous message blocks. The design of the compression function is defined to attain the desired cryptographic proprieties such as message and key avalanche effect, randomness, uniformity with only one round of iteration. The advantage of the proposed scheme is that it reduces the number of rounds compared with other recent standardized algorithms (CMAC, GMAC, HMAC). Indeed, this can ensure a lower execution time and reduced latency, computing, resources and energy consumption. This is crucial for the restricted and limited applications and devices resources.

## Acknowledgement

This paper was partially supported by funds from the Maroun Semaan Faculty of Engineering and Architecture at the American University of Beirut and by the EIPHI Graduate School (contract ”ANR-17-EURE-0002”).

## References

[1] W. Stallings, “The principles and practice of cryptography and network security 7th edition, isbn-10: 0134444280,” *Pearson Education*, vol. 20, no. 1, p. 7, 2017.

[2] F. P. Miller, A. F. Vandome, and J. McBrewster, *Advanced Encryption Standard*. Alpha Press, 2009.

[3] J. Song, R. Poovendran, J. Lee, and T. Iwata, RFC 4493 (Informational), Internet Engineering Task Force, June.

[4] D. McGrew and J. Viega, “The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH,” RFC 4543 (Proposed Standard), Internet Engineering Task Force, may 2006.

[5] H. Handschuh, H. C. A. van Tilborg and S. Jajodia, Eds. Springer.

[6] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, “The keccak reference,” Submission to NIST (Round 3), 2011. [Online]. Available: http://keccak.noekeon.org/Keccak-reference-3.0.pdf

[7] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, “Sha-3 proposal blake,” Submission to NIST (Round 3), 2010. [Online]. Available: http://131002.net/blake/blake.pdf

[8] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and T. S. S., “Grøstl - a sha-3 candidate,” in *Symmetric Cryptography*, ser. Dagstuhl Seminar Proceedings, H. Handschuh, S. Lucks, B. Preneel, and P. Rogaway, Eds., no. 09031. Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2009/1955

[9] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker, “The skein hask function family,” 2009.

[10] H. Krawczyk, M. Bellare, and R. Canetti, “Hmac: Keyed-hashing for message authentication,” 1997.

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

[12] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “Simon and speck: Block ciphers for the internet of things.” *IACR Cryptology ePrint Archive*, vol. 2015, p. 585, 2015.

[13] H. Noura, A. Chehab, M. Noura, R. Couturier, and M. M. Mansour, “Lightweight, dynamic and efficient image encryption scheme,” *Multimedia Tools and Applications*, vol. 78, no. 12, pp. 16 527–16 561, 2019.

[14] 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.

[15] 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

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.