TRESC: Towards redesigning existing symmetric ciphers

Hassan Noura, Ola Salman, Nesrine Kaaniche, Nicolas Sklavos, Ali Chehab, Raphael Couturier

TRESC: Towards Redesigning Existing Symmetric Ciphers

Hassan N. Noura[1], Ola Salman[2], Nesrine Kaaniche[3], Nicolas Sklavos[4], Ali Chehab[2], and Raphaël Couturier[1]

[1]Univ. Bourgogne Franche-Comté (UBFC), FEMTO-ST Institute, CNRS, Belfort, France
[2]American University of Beirut, Electrical and Computer Engineering, Lebanon
[3]Department of Computer Science, University of Sheffield, UK
[4]University of Patras, Computer Engineering & Informatics Department, Greece

***Abstract*—Recently, the security of existing symmetric cryptographic algorithms and protocols has been threatened by new performance challenges and vulnerabilities. In this paper, we propose a dynamic key-dependent approach, "TRESC", to make existing symmetric ciphers more efficient and robust. This can be done by using dynamic substitution and permutation primitives to reduce the number of rounds while providing better resistance against cryptanalysis and implementation attacks. In this paper, the Key Setup Algorithm (KSA) of Rivest Cipher 4 (RC4) and its modified variants are applied for the construction of these dynamic key-dependent substitution and permutation primitives. The selection of the RC4-KSA is due to its lightweight implementation since it requires simple permutation operation with minimal overhead. The proposed dynamic cryptographic solution can be integrated in any existing symmetric cipher such as Advanced Encryption Standard (AES), SIMON and SPECK. The security and performance analysis show the robustness and effectiveness of the proposed solution, which strikes a good balance between the required security level and system performance.**

Keywords: Key-dependent cryptographic primitives; cryptographic analysis; dynamic cryptographic approach.

I. Introduction

Recently, the rapid development and spread of the communication and computing technologies challenge the existing cryptographic algorithms [1], [2]. In this context, these algorithms, mainly deployed to protect the proliferated data, exhibit several performance and security limitations. To protect data secrecy, the existing solutions rely on symmetric key cryptography due to its efficient memory use and reduced computational complexity compared to the asymmetric one. In fact, symmetric ciphers are divided into two types: block ciphers and stream ciphers. Stream ciphers are typically less secure compared to block ciphers, which divide the data into separate blocks of fixed size (usually 128 bits). Note that a block cipher can be considered as a stream cipher when applied with Output FeedBack (OFB) or Counter (CTR) mode [3].

The confusion and diffusion properties are key requirements for secure cipher schemes [4]. The confusion property plays a vital role in preventing conventional crypatanalysis attacks such as linear and differential attacks. Thus, this property can be ensured by applying the substitution operation. Otherwise, the diffusion property can be ensured by using the permutation or the matrix multiplication operation. In most existing ciphers, the substitution and permutation operations use static S-box and P-box, respectively. In this context, there are two kinds of cipher structures: static and dynamic [5]. The first one employs static confusion operations that can achieve the maximum difference propagation probability and the maximum input-output correlation probability to defend against differential and linear attacks. Also, it uses static diffusion functions with high linear branch number as a linear mixing transformation, as in AES, SAFER, and 3-WAY algorithms [6]. This type of ciphers has proven security against cryptanalysis attacks. However, the static structure is prone to future potential attacks, which can be overcome by using a dynamic cipher structure [5].

The dynamicity helps in achieving a high security level and it should be implemented in a way to result a low overhead in terms of computational complexity and resource requirements. Thus, the main goal of this paper is to strike a good balance between system performance and security level by introducing the dynamic key-dependent approach into existing cipher schemes. The proposed dynamic approach is designed to be incorporated into the substitution and permutation operations since they require less overhead compared to the optimized diffusion mixing operation. As such, existing ciphers will be complemented with new lightweight, simple and dynamic key-dependent substitution and permutation techniques. This will lead to the generation of different substitution/permutation tables during the encryption/decryption process. In this paper, we use the Key Setup Algorithm (KSA) of RC4 and its modified version towards producing dynamic substitution and permutation tables.

For each new session, a session key is generated and hashed to produce a new dynamic key, which is then divided into 3 sub-parts; the first one is used as the traditional session key to produce a set of round keys. The second part is used to produce the cryptographic primitives (substitution and permutation tables). The third sub-key part is used to produce the update cryptographic primitives (three permutation tables), which are used to update the cryptographic primitives for

each new message (or a set of messages). In fact, the update mechanism of the cryptographic primitives depends on the selected configuration and the required security level. In addition, the maximum overhead introduced in terms of latency is reached when the S-box and P-box are changed for each data input block. Moreover, the proposed key-dependent substitution and permutation construction algorithm includes a large key space to resist the different attacks such as brute force attack. To assess the cryptographic properties of each generated cryptographic primitive, a set of security tests were performed. The results showed that the produced primitives satisfy the desired cryptographic properties such as randomness, sensitivity and independence. Therefore, the proposed solution makes the existing ciphers more efficient and robust.

The rest of this paper is organized as follows. Section II reviews related dynamic key-dependent cryptographic approaches. The proposed dynamic cryptographic primitives derivation is presented in Section III. Section III-C presents an example of integrating the proposed dynamic key-dependent algorithm into Advanced Encryption Standard (AES) [7]. Section V analyzes the properties of the proposed approach, especially in terms of cryptographic performance, flexibility and the required number of iterations. Finally, Section VI concludes this work.

II. Related Work

In this section, we review the existing lightweight and key-dependent ciphers, and substitution and diffusion primitives.

A. Lightweight key-dependent Cipher

Existing symmetric ciphers rely on static cryptographic primitives that are independent of the secret key. Traditional symmetric ciphers, such as the Advanced Encryption Standard (AES) [8], consist of running a round function across several rounds. This round function consists of several substitution and permutation operations. Being computationally expensive, existing symmetric ciphers require a large number of rounds to guard against cryptanalysis attacks. For example, the Hummingbird2 cipher requires a minimum of 4 rounds to be immune against existing security attacks [5]. This imposes a high overhead in terms of latency and resources. However, given the huge amount of generated data in the emerging network technologies, especially in the Internet of Things (IoT) era, lightweight cryptographic algorithms are required [9]. In this context, recently, new ciphers have been proposed, requiring less number of rounds. Examples of these ciphers are Simon and Speck [10]. However, these ciphers present high latency, and require high resources and computational overhead [11].

Having time-critical applications running on tiny IoT devices, decreasing the number of rounds is not enough to guarantee a high performance level. Moreover, the security of static key ciphers could be threatened by attackers who are equipped with ever-increasing capabilities [12], [13]. The attackers would try to recover the secret key given that the static ciphers use fixed cryptographic primitives [14]. Thus, dynamic lightweight cipher schemes are required to strike the balance between security and performance[15], [16], [17], [18], [19], [20].

A recent solution was to apply the "chaotic" approach to design lightweight ciphers [11], [21]. However, chaotic cryptography requires specific hardware implementations, floating-point computations, conversion operations, finite periodicity, and a multi-round structure. More recently, another approach has been proposed based on the dynamic key approach with low number of rounds [5], [22], [23]. These ciphers reduce the computational complexity, yet, they require hardware optimization. In this paper, we aim at redesigning existing ciphers by introducing the dynamic key approach while reducing the required number of rounds. Thus, the proposed solution does not require any specific hardware implementation and can be easily adopted.

B. Existing Substitution Construction Techniques

Random key-dependent S-boxes generation techniques are presented in [24], [25], [26]. These techniques enable the generation of variable S-boxes under the control of a secret key. In [27], a recursive generation technique of key-dependent S-boxes is continuously performed until a good S-box is found, which requires a variable overhead. This technique discards the produced S-box if it possesses a weak cryptographic performance, until an S-box with a good performance level is found. In [25], a pseudo-random S-box key-dependent method is presented. Moreover, another method based on chaos theory is presented in [28], where the S-boxes are obtained by using a discretization of the exponential or logistic maps. Similarly, another approach is proposed in [29] and it is based on the 2D discretized chaotic baker map. The 2D map was further extended to a 3D one in [30]. Another method is presented in [31] and it consists of iterating a continuous chaotic map with key-dependent starting points. In [32], a dynamic S-box generation method is presented based on the Lorenz system and a special shifting method. A hyper-chaotic based method for generating a dynamic S-box is presented in [33]. However, these methods exhibit many limitations inherited from the pseudo-random generators and chaotic functions. Thus, constructing pseudo-random dynamic primitives is not a straightforward task. Moreover, the performance and security of these methods are not guaranteed [28]. For example, the chaotic based methods use floating chaotic functions that typically introduce high computational complexity. Moreover, the presented methods require specialized hardware implementations.

In this paper, we aim at redesigning existing cipher schemes to become more efficient and robust. For this reason, we propose a new dynamic key-dependent, efficient and secure method to generate dynamic substitution and permutation tables, with the aim of attaining the desired cryptographic

properties. Thus, the contributions of this paper can be summarized as follows:

- Proposing a new dynamic lightweight key-dependent method for generating dynamic substitution/permutation tables with minimum overhead in terms of latency and resources.
- Reducing the number of rounds, which minimizes the required latency and resources overhead for the encryption/decryption process. This reduction helps in achieve the balance between security level and performance.
- Integrating the dynamic key-dependent approach into existing symmetric ciphers to achieve a high level of security and to defend against modern cryptanalysis attacks.

III. Construction of Dynamic Cryptographic Primitives

In this section, we describe the dynamic key generation, the construction of the cryptographic and update cryptographic primitives, and the integration into the existing ciphers. Let us indicate that TABLE I represents all of the notations used in this paper.

A. Dynamic Key Generation

For each new session, to produce the dynamic key ($DK = h(SK)$), the session secret key SK is hashed, Where h represents any secure cryptographic hash function (e.g. SHA-512). Moreover, the size of the nonce is 512 bits. Therefore, a large set of different nonces can be employed (2^{512}). Consequently, the proposed approach has a low nonce collision. NIST recommends 4 secure Deterministic Random Bit Generator (DRBG) algorithms that can be employed for key and Initial Vector (IV) generation for cryptographic applications. These DRBG algorithms are detailed in the NIST SP800-90A specification, where two of them are based on keyed hash function. These DRBG based hash functions can be employed for the nonce generation in this paper. Indeed, the generated nonces ensure a high periodicity with a low collision probability. Given that a new session key is produced for each new session, different dynamic keys will be produced even with the same nonce. This is due to the fact that the dynamic key depends of the nonce and the session secret key. As such, the attacker cannot know if the nonce is regenerated and consequently this is not an issue in practical deployments of the proposed solution. In the worst case, if the attacker recovers the nonce, the secret key cannot be recovered (one-way hash function) from the collected ciphertext.

The obtained dynamic key is longer than the session key and it is split into 4 sub-keys, as described next (see Fig. 1).

1) The sub-key Dk_{RK}: it consists of the first l least significant bits of DK and it is used as input to the key expansion algorithm to generate a set of round keys (as the session key in the traditional case). l represents the size of the session key and it can be equal to 128, 192 or 256 in case of AES.
2) The second sub-key represents the next 128 most significant bits of DK as it is divided into:
   - The sub-key Dk_S: it consists of the first 64 least significant bits of the second sub-key. It is used to produce a primary substitution table S via KSA, as described in [5].
   - The sub-key Dk_{US}: it consists of the first 64 most significant bits of the second sub-key. It is used to produce a permutation table U_S by using the Modified Key-Scheduling Algorithm (MKSA) of RC4. This permutation table has 256 elements with pseudo-random values varying between 0 and 255.
3) The sub-key Dk_P: it consists of the second 128 most significant bits of DK. It is used to produce a primary permutation table π via MKSA, as described in [5], [34].
4) The sub-key Dk_{UP}: it consists of the first 128 most significant bits of DK and it is used to produce the update permutation table U_π, using MKSA. This update table has a length related to the employed permutation table in the adapted cipher. For example, it is equal to 16 in the case of AES (shiftRows).

Note that the dynamic key is updated after τ message blocks and the cryptographic primitives can be updated after δ message blocks, with $\delta \leq \tau$. τ and δ are configured based on the application requirements. U_S is used to permute the produced substitution table S and U_π is used to permute the produced permutation table π, as follows:

$$\begin{aligned} S &= S(U_S) \\ \pi &= \pi(U_\pi) \end{aligned} \tag{1}$$

B. Dynamic Cryptographic Primitives Construction

1) Dynamic Substitution Primitives: In the proposed approach, the KSA step of RC4 is used to initialize a substitution table $S \triangleq \{s[0], \cdots, s[255]\}$ with 256 elements, using a secret key. The size of the secret key is variable and ranges between 64 and 256 bits. For a secure use of RC4, the key size must be at least 128 bits. The pseudo-code for the KSA step of the RC4 algorithm is shown in Algorithm 1. The parameter L_K is the length of the dynamic key in bytes; i and j are iteration variables. A numerical example of a produced dynamic substitution table with its corresponding inverse is presented in Fig. 2.

The cryptographic performance of the proposed substitution table is quantified for a Dk_S with 64-bit length. The obtained results, illustrated in Fig. 6, show that the produced substitution tables satisfy the desired cryptographic properties (LP_F, SAC, BIC and DP_F) as presented in Section IV-B. Therefore, each S-box is obtained by applying KSA for a secret key Dk_S of size ≥ 64 bits.

On the other hand, the inverse substitution table is computed for the decryption process, which is feasible since the produced substitution table is bijective. Hence, the inverse substitution table, S^{-1} is also bijective, and can be obtained using the produced S-box by using the following operation:

TABLE I: Table of notations

Notation	Definition
SK	Secret session key
l	The size of the session key
DK	The produced dynamic key
Dk_{RK}	A dynamic sub-key, which is used as the input of the key expansion algorithm to generate a set of round keys
Dk_S	A dynamic substitution sub-key and it is used to produce a primary substitution table S.
Dk_{US}	A dynamic update substitution sub-key and it is used to produce an update substitution table U_S by using the MKSA of RC4.
Dk_P	A dynamic permutation sub-key and it is used to produce a primary permutation table P
Dk_{UP}	A dynamic update permutation sub-key and it is used to produce an update permutation table U_π by using the MKSA of RC4.
Lp_k	Length of the dynamic permutation sub-key Dk_P and Dk_{UP}
L_k	Length of the dynamic substitution sub-key Dk_S and Dk_{US}
len	Number of elements in a permutation table
L_S	Number of elements in a substitution table
r	Number of rounds (iterations)
π	Permutation table
π^{-1}	Inverse permutation table
S	Substitution table
S^{-1}	Inverse substitution table
$\oplus$	Exclusive-OR (XOR)
h	Secure cryptographic hash function (SHA-512 is used as an example in this paper)
δ and τ	The dynamic key is updated after τ message blocks and the cryptographic primitives can be updated after δ message blocks, with $\delta \leq \tau$.

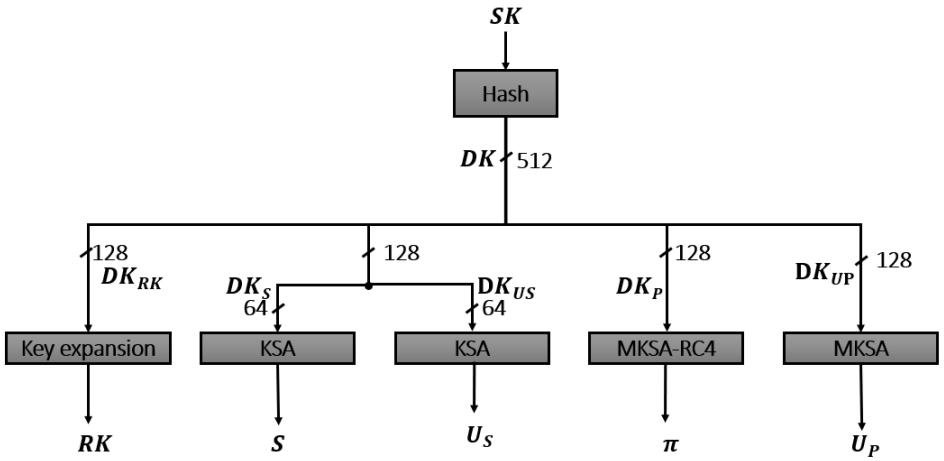


Fig. 1: The proposed key derivation method and the cipher primitives construction technique

Algorithm 1 KSA algorithm of RC4

Input: L_K length of key; $K \triangleq \{k_1,\ k_2,\ \ldots,\ k_{L_K}\}$; L_S length of state array S
Output: $S \triangleq \{S[0],\ S[1],\ \cdots, S[L_S-1]\}$
procedure $S = \mathsf{RC4\text{-}KSA}(K, L_K, L_S)$
  for $i \leftarrow 0$ to $L_S - 1$ **do**
    $S[i] \leftarrow i$
  $j \leftarrow 0$
  for $i \leftarrow 0$ to $L_S - 1$ **do**
    $j \leftarrow j + s[i] + k[i \pmod{L_K}] \pmod{L_S}$
    $\mathsf{SWAP}(S[i], S[j])$ ▷ swap values of S[i] and S[j]
  return S

$$S^{-1}[S(j)] = j,\ j = 0,\ 1,\ \ldots,\ 255 \tag{2}$$

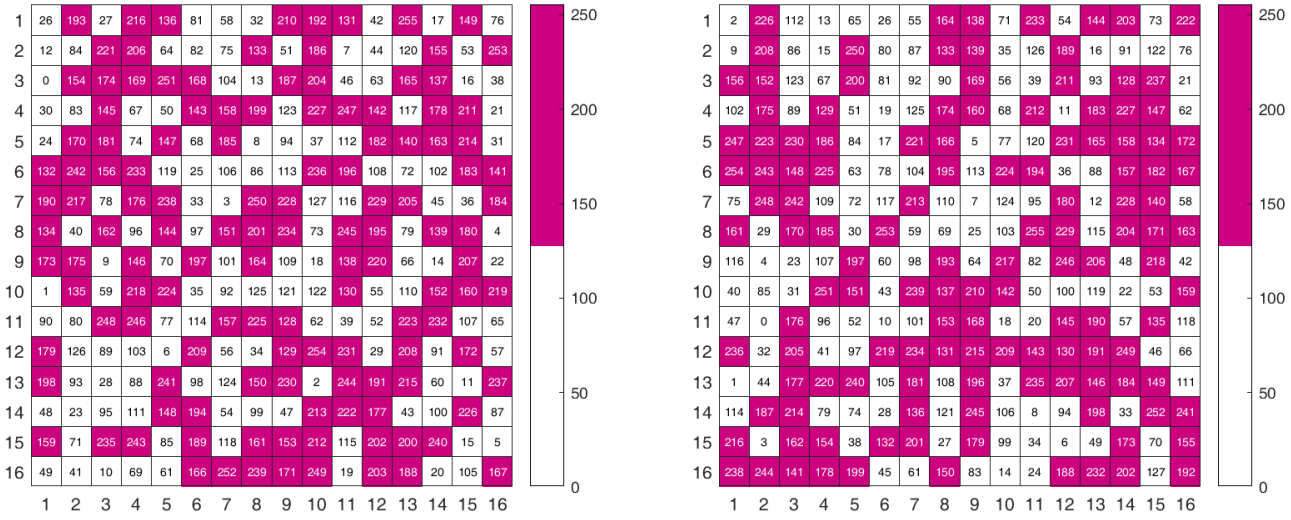
where S represents the produced S-box and $0 \leq S(j) \leq 255$.

2) Dynamic Permutation Primitives: The permutation operation in a symmetric cipher can be applied at the bit or byte level. As for the substitution table generation, the proposed method uses the modified KSA of RC4 to produce the permutation tables. To do so, an input key of length Lp_k is passed to the generation function to obtain a dynamic permutation table π with len elements (see Algorithm 2). This permutation table will be updated using U_π.

For decryption, the inverse permutation table π^{-1} has to be computed. This is possible since π is bijective. The inverse π^{-1} of π can be obtained given that $\pi^{-1}[\pi(i)] = i$, with $\pi(i)$ being the value of π at the i^{th} index and $1 \leq \pi(i) \leq len$. The permutation is a swap function, with $(\pi(i))$ being the permuted byte or bit position of the i^{th} element.

C. Proposed Dynamic Cryptographic Approach

In this section, we present a proof of concept of the proposed solution modification in a block cipher. For this



(a) (b)

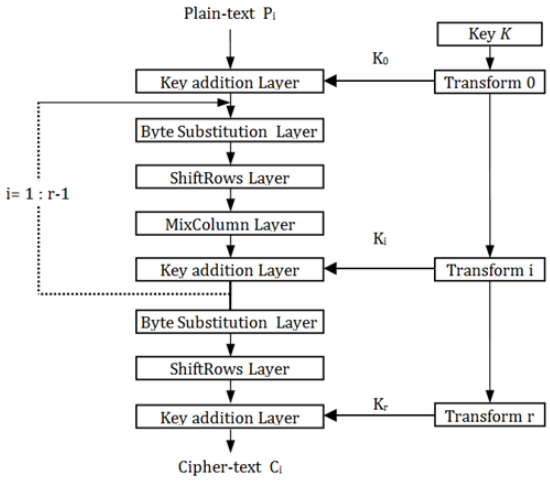
Fig. 2: Example of a produced dynamic substitution table S-box (a) and its corresponding inverse one (b) by using KSA of RC4.

Algorithm 2 Proposed Modified KSA (MKSA)

procedure MKSA($K = \{k_1,\ k_2,\ \ldots,\ k_L\}, Lp_k, len$)
  for $1 \leftarrow 1$ to len **do**
    $\pi\ [i] \leftarrow i$
  $j \leftarrow 1$
  for $i \leftarrow 1$ to len **do**
    $j \leftarrow (j + \pi[i] + k[j \bmod Lp_k + 1])\ \bmod len + 1$
    $swap(\pi[i],\ \pi[j])$
  return π

purpose, we selected the actual cipher standard AES that is widely deployed in various applications. AES is an iterative symmetric block cipher shown in Fig. 3, operating on input / output data block sequences of 128 bits. First, the plain text message is divided into 16 bytes blocks. Then, each block is reshaped into a 4×4 matrix, called state, which is the input data for AES, where each byte represents a value of $GF(2^8)$. The key size can be of size 128 bits, 192 bits, or 256 bits. The key size defines the number of encryption / decryption iterations: 10, 12, and 14 iterations for a 128, 196, 256-bit key, respectively. In addition, the round function of AES consists of 4 distinct byte-oriented processes, SubBytes, ShiftRows, MixColumns, and AddRoundKey, as described below.

1) SubBytes (SB): This operation applies a non-linear transformation at the byte level (8-bit input/output) based on a static substitution table S-box.
2) ShiftRows (SR): This operation rotates all the lines of the state to the left (0 for the first line, 1 for the second, 2 for the third, and 3 for the fourth). It can be considered as a byte permutation operation and uses a specific permutation table.
3) MixColumns (MC): In this operation, each column of the input matrix is multiplied by a static MixColumns



(a)

Fig. 3: AES algorithm

matrix in $GF(2^8)$. This operation ensures the diffusion at the column level.

4) AddRoundKey (AK) : The input state (matrix) is mixed (Xor) with the produced sub-key of the current round. Note that all sub-keys are related only to the secret key.

Furthermore, the secret key is mixed with the input state plain block before the first round, and in the last round, the MixColumns function is eliminated. The decryption scheme is similar to the encryption one, but it is performed in the reverse order. Moreover, in the decryption process, the diffusion, permutation, addition, and substitution primitives are replaced by their inverses.

Indeed, the substitution process (using look up table S-box) and key expansion (employing S-box in the round key generation as in AES) are affected by the dynamic S-boxes generation technique. Consequently, this will lead to dynamic characteristics of the cryptographic algorithm's output, leading to higher robustness against attacks since the round keys are dynamic. Moreover, for the same input, the substitution's output changes in different sessions, which complicates the attacker's task in disclosing the secret key. Moreover, we use a dynamic permutation table to increase the dynamicity and to reach better immunity against attacks.

Therefore, the proposed method presents low complexity, as shown later in Section V, which results into low latency and reduces the required resources, while providing high robustness against attacks. In fact, the execution time and energy consumption overhead can be reduced by decreasing the number of rounds r to the minimum required level to reach the avalanche effect.

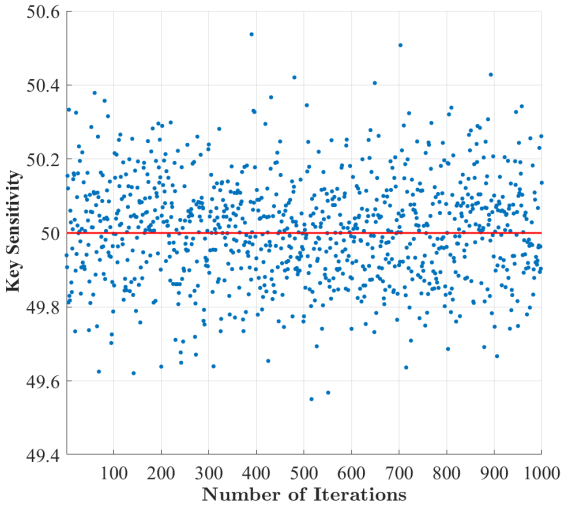
IV. Security Analysis

In this section we analyse and evaluate the desired cryptographic properties of the proposed dynamic key-dependent cryptographic approach. This security analysis is based on the methodology presented in [5], [35]. This part focuses on analyzing the produced cryptographic primitives and their update mechanism to prove that they achieve the desired cryptographic properties. In the following, the randomness tests of the generated dynamic key and cryptographic primitives are detailed. In addition, the sensitivity of the dynamic cipher primitives (update of permutation/substitution boxes) and cipher-texts are verified. It should be noticed that, for the following tests, some parameters are fixed such as the number of iterations, set to $10,000$.

A. Robustness of the Proposed Dynamic Key Generation

The proposed solution uses a secret session key SK which changes at each session time, depending on the application requirements. SK is hashed using $SHA-512$ to produce a dynamic key DK. Hence, the cryptographic hash function ensures the avalanche effect [36], as shown in Fig. 4 for a test with 1000 random secret keys.

The results are very close to 50%, and the samples follow the normal distribution, with a mean equals to 49.9285% and a standard deviation equals to 2.1871. Moreover, the minimum difference is equal to 43.3594%, which is sufficient to have different S-boxes when using different secret keys (even with a one bit of difference). Consequently, the use of a dynamic session secret key overcomes the fixed key problem and prevents the production of the same S-box, which translates into better resistance against attacks. Note that the secret key SK size can be 128, 256, or 512 bits, while the size of dynamic key DK is 512 bits.



(a)

Fig. 4: Dynamic key sensitivity of the proposed dynamic key generation versus 1000 iterations. Each iteration, only one bit of the secret key is changed.

B. Cryptographic Primitives Related Tests

The dynamic permutation/substitution tables should be random and independent compared to primary or previous obtained ones to guarantee their secure implementation in the proposed cipher variants. To assess the randomness of the permutation/substitution tables for different inputs, recurrence and correlation coefficient ρ measures are used [37], [13]. Note that these tests were applied for $1,000$ random dynamic keys.

1) Recurrence Test: To measure the randomness among a vector, the recurrence test is employed to measure the correlation between a sequence of data (vector) $x_i = x_{(i,1)}, x_{(i,2)}, x_{(i,3)}, ..x_{(i,m)}$, and another vector with delay $t \geq 1$ given by $x_i(t) = x_{(i,t)}, x_{(i,2t)}, x_{(i,3t)}, ...x_{(i,mt)}$. Thus, the correlation coefficient r_{xy} between two vectors x and y can be calculated using the following equation:

$$r_{xy} = \frac{cov(x,y)}{\sqrt{D(x) \times D(y)}} \tag{3}$$

where

$$E_x = \frac{1}{N} \times \sum_{i=1}^{N} x_i$$

$$D_x = \frac{1}{N} \times \sum_{i=1}^{N} (x_i - E(x))^2$$

$$cov(x, y) = \frac{1}{N} \times \sum_{i=1}^{N} (x_i - E(x))(y_i - E(y))$$

The recurrence of a primary substitution table that is generated by using KSA with a dynamic random key is shown in Fig. 5-a). As it can be seen, the generated substitution table has a highly dispersed recurrence map, and thus has the desired degree of randomness. Moreover, Fig. 5-b) shows the variation of the correlation coefficient between the original (primary S-box) and the updated substitution table as a function of the number of iterations, which is always close to zero. This shows that the updated S-box is independent of the original S-box. The obtained correlation coefficient values are within $\{-0.15, 0.15\}$ (always close to zero), which validates the uncorrelation (independence) of the original and updated substitution tables. Moreover, Fig. 5-c) shows the variation of the correlation coefficients between the recurrence of the permuted index versus 1,000 random dynamic keys (for $\alpha = 256$). The recurrence correlation is close to zero, and hence it can be inferred that the produced permutation table is highly random. More specifically, most values are spread evenly over $\{-0.2, 0.2\}$ (optimum value, 0). Finally, Fig. 5-d) shows the variation of the correlation coefficient (which is also close to zero) between two successively updated S-boxes for 1,000 iterations. This proves the independence of any two successive substitution tables. These results reveal that there is no correlation among the primary and the updated substitution tables.

The statistical results are presented in TABLE II. The standard deviation for all three considered cases (TABLE II) is near zero, which proves that most correlation coefficient values are close to the mean value of zero, which is the ideal value. This confirms the high uniqueness and dynamicity level of the produced substitution and permutation tables.

On the other hand, similar results are obtained with the proposed method to construct dynamic permutation tables. These results are expected since the same technique is used to produce updated permutation tables. Therefore, different permutation and substitution tables are produced for each input message (or a set of messages, depending of the configuration). This results into a high immunity against attacks such that no useful information can be disclosed by the attackers.

From the previous results, it is clear that the primary and the produced P-boxes are uncorrelated. This is also true for any two successively updated P-boxes. Therefore, the produced permutation tables possess high dynamicity and uniqueness levels, which guards against eavesdropping and prevents illegitimate users from acquiring useful information. Moreover, the results clearly show that the proposed update permutation scheme produces independent/un-correlated versions of the same cipher primitive (for example a permutation table), which enables the resistance against current cryptanalysis techniques.

Algorithm 3 The proposed update selection table cipher primitive algorithm

Input: Number of blocks it
Two substitution tables S, π_S,
Two permutation tables(π, π_p)
Output: Update substitution and permutation tables (S and P, respectively)

1: **procedure** UP_TABLE_PRIM(S, U_S, P, U_p)
2: **if** $it\%\delta = 0$ **then**
3: $\pi \leftarrow Perm(\pi, U_p)$
4: $S \leftarrow Sub(S, U_S)$
5: **return** S and P

Using the proposed simple update process, the cryptographic primitives are changed for every input frame, regularly and randomly. Therefore, each input frame will be encrypted using a set of completely different selection tables and diffusion matrices, which obscures the relation between the original frames and their encrypted versions. Hence, threats related to chosen-plaintext attacks and chosen-ciphertext attacks are successfully mitigated. This is due to the fact that choosing a set of original or encrypted frames will not help the adversary in acquiring the used primitives nor the the secret key itself.

In the following, the cryptographic properties of the proposed dynamic key-dependent construction of substitution and permutation tables are presented. The most important substitution tests are applied in order to prove its performance such as linear [38], and differential probability approximation [39], strict avalanche criterion (SAC) and output bits independence criterion (BIC) [40]. Also common tests such as randomness and key sensitivity measures were performed on produced substitution and permutation tables to assess their security level and to show how far it is consistent with existing cipher standards. In the following, we start by analyzing the substitution tests, then the common tests are described.

2) Bijectivity: in order to check the bijectivity of the obtained S-box, we compute the number of its unique elements, using the *unique* function. If the length is equal to 2^q, then, the bijectivity is attained of the S-box under test; else the S-box is not bijective. Thus, given that we used the swap function and the permutation layer is bijective, so the bijectivity is preserved.

3) Linear Probability Approximation Boolean Function (LP_F): One of the important properties of the substitution layer is its non-linearity for resisting linear cryptanalysis attacks. LP_F is used to measure the degree of non-linearity for

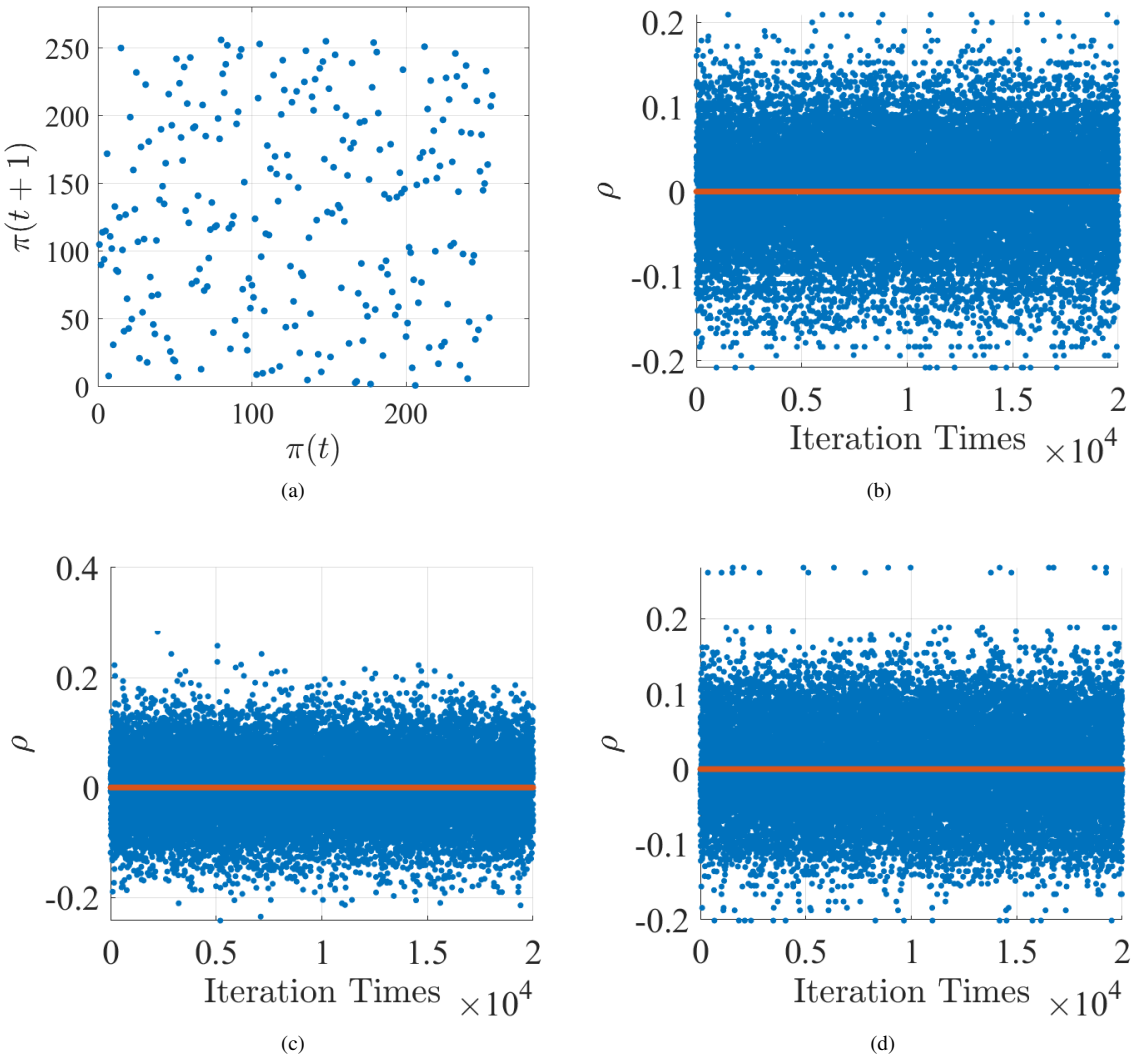


Fig. 5: (a) The recurrence plot of a randomly generated primary substitution table. The variation (20000 times) of the correlation coefficient of (b) the recurrence of produced dynamic substitution tables, (c) the correlation coefficient between the primary substitution table and its updated version (permuted version), and (d) the correlation coefficient between two successive substitution tables.

TABLE II: Statistical results for 1000 update permutation iterations

Coefficient Correlation Tests	Min	Mean	Max	Std
ρ of the recurrence of produced dynamic substitution tables	-0.2083	0.0010	0.2093	0.0623
ρ between the primary substitution table and its updated version (permuted version)	-0.2016	0.0028	0.2681	0.0623
ρ between two successive substitution tables	-0.2421	0.0001	0.2835	0.0626

TABLE III: Statistical results of the first proposed substitution algorithm

Test	Min	Mean	Max	Standard deviation
LP_F	$2^{-5.6601}$	$2^{-4.8067}$	$2^{-3.6601}$	0.3234
DP_F	$2^{-4.6781}$	$2^{-4.4939}$	$2^{-3.8301}$	0.1527
SAC	0.4854	0.5014	0.5166	0.0054
BIC	0.4939	0.5020	0.5098	0.0027
KS	45.8008	49.3245	52.8320	1.0933

a given substitution function [38]. Decreasing LP_F tends to increase the linear attack complexity. The non-linearity degree of a given confusion layer F is calculated using the linear probability approximation Boolean function LP_F according

to the following equation:

$$LP_F = Max_{\alpha,\beta\neq 0}[LP_F(\alpha,\beta)] = Max_{\alpha,\beta\neq 0}\ \{\frac{card\{i/i\circ\alpha = S(i)\circ\beta\} - 2^{n-1}}{2^{n-1}}\} \quad (4)$$

where $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_n], \beta = [\beta_1, \beta_2, \ldots, \beta_n], \alpha, \beta \in [1, 2^{n-1}]$ and card is the cardinal. Theoretically, $LP_F(\alpha,\beta) \approx \frac{1}{2^{n-1}}$ and so decreasing LP_F leads to increasing the complexity of the linear attack.

Furthermore, the immunity of the S-box against the linear cryptanalysis depends on the uniformity of LP_F versus α or β. The low LP_F signifies that the complexity of the linear attack is much higher. For example, LP_F of the AES confusion layer is equal to $2^{-6} = 0.015625$. In Fig. 6-a), the variation of LP_F versus 1000 different dynamic keys are shown.Besides, the LP_F obtained maximum, minimum, and average are: $2^{-3.66}$, $2^{-5.66}$, and $2^{-4.8}$, respectively. These results obtained for the LP_F criterion show clearly that the majority of the produced substitution values are close to the average value. This means that the proposed dynamic substitution technique satisfies the non-linearity property and thus ensures the resistance against linear attacks. In fact, having a linearity probability ($\approx 2^{-5}$) higher than the minimum value achieved by AES S-box (2^{-6}) is not an issue since dynamic substitution tables are employed.

4) Differential Probability Approximation Function (DP_F): Differential uniformity is one of the important properties of the S-box to resist the differential cryptanalysis attacks [39]. Hence, the produced S-box should ensure the differential uniformity; an differential input should uniquely map to a differential output, which can provide a uniform mapping probability. The differential uniformity degree of a given substitution function S is calculated using the differential approximation probability DP_F [41], given by:

$$DP_F = Max_{(\Delta i\neq 0,\Delta s)}[DP_F(\Delta i,\Delta s)] = Max_{\Delta i\neq 0,\Delta F}\{\frac{card\{i/S(i)\circ S(i\oplus\Delta i) = \Delta s\}}{2^n}\} \quad (5)$$

where $\Delta i \in [1, \ldots, 2^{n-1}], S(i), \Delta s \in [0, 2^{n-1}]$.

In fact, DP_F is the maximum probability of output difference Δf, when the input difference is Δi. In Fig. 6-b), the variation of DPF versus 1000 different dynamic keys are shown and the probability of $DP_F < 2^{-4}$ is 0.8477, and only 0.0156 of the substitution layers have $DP_F > 2^{-3.5}$. Besides, the DP_F obtained maximum, minimum, and average are: $2^{-3.8301}$, $2^{-4.678}$, and $2^{-4.4939}$, respectively. These results show that the majority of the produced substitution tables reach the required security level to resist against differential attacks. Similarly, having a differential probability ($\approx 2^{-4.7}$) higher than the minimum value achieved by AES S-box (2^{-6}) is not an issue since dynamic substitution tables are used.

5) Strict Avalanche Criterion (SAC): SAC was introduced by Webster and Tavers in 1985 when they described the avalanche effect [40]. A cipher scheme is said to satisfy the SAC if each time a single input bit is complemented, at least half of the output bits are changed. SAC is certainly considered as mandatory property, so any strong cipher scheme should use a substitution table that can achieve this criterion. It is used to quantify the degree of local avalanche effect at the byte (or word) level. The average SAC value (average of 8x8 values of the dependency matrix) is given in Fig. 6-c) for 1000 iterations and with 64-bit key length. It can be observed that the produced substitution tables have SAC values very close to the ideal value of 0.5. This shows that the majority of the produced substitution tables, using KSA with dynamic sub-key, meet this criterion.

6) Output Bits Independence Criterion (BIC): BIC is another desirable property of the encryption algorithms described by Webster and Tavers [40]. The BIC specifies that two output bits $j,\ k$ must independently change when a single i input is changed. The mean value results of BIC (average of 8x8 values of the BIC matrix without the diagonal part) as a function of the number of iterations rs (for each iteration, a new pseudo-random byte is used) is given in Fig. 6-b). It can be observed that the majority of the mean values of BIC are all around the optimal value of 0.5. The results in Fig. 6-d) clearly show that the majority of the produced S-boxes satisfy the BIC criteria. Thus, the proposed dynamic substitution primitives construction method achieves the required cryptographic properties (see tables III and IV) and can successfully overcome the known chosen plain/cipher text attacks.

TABLE IV: Comparison analysis of the substitution layer

Test	Average (proposed solution)	AES (static)
LP_F	$2^{-4.8}$	2^{-6}
DP_F	$2^{-4.49}$	2^{-6}
SAC	0.5022	0.4998
BIC	0.5018	0.499

Now, the common tests such as key sensitivity and randomness level are presented to validate that the produced substitution and permutation tables satisfy the desired cryptographic properties.

7) Sub-Keys Sensitivity: An appropriate dynamic key-dependent substitution generation technique should be sensitive to its dynamic secret key in the sense that a change of one bit should give a completely different output S-box. The sensitivity is examined according to the percent Hamming distance measured in bits between two S-boxes resulting from two input secret keys K_w and K'_w. All the elements of K'_w are equal to those of the w^{th} key K_w, except a random Least Significant Bit (LSB), which was flipped. The percent Hamming distance is defined and used to quantify the

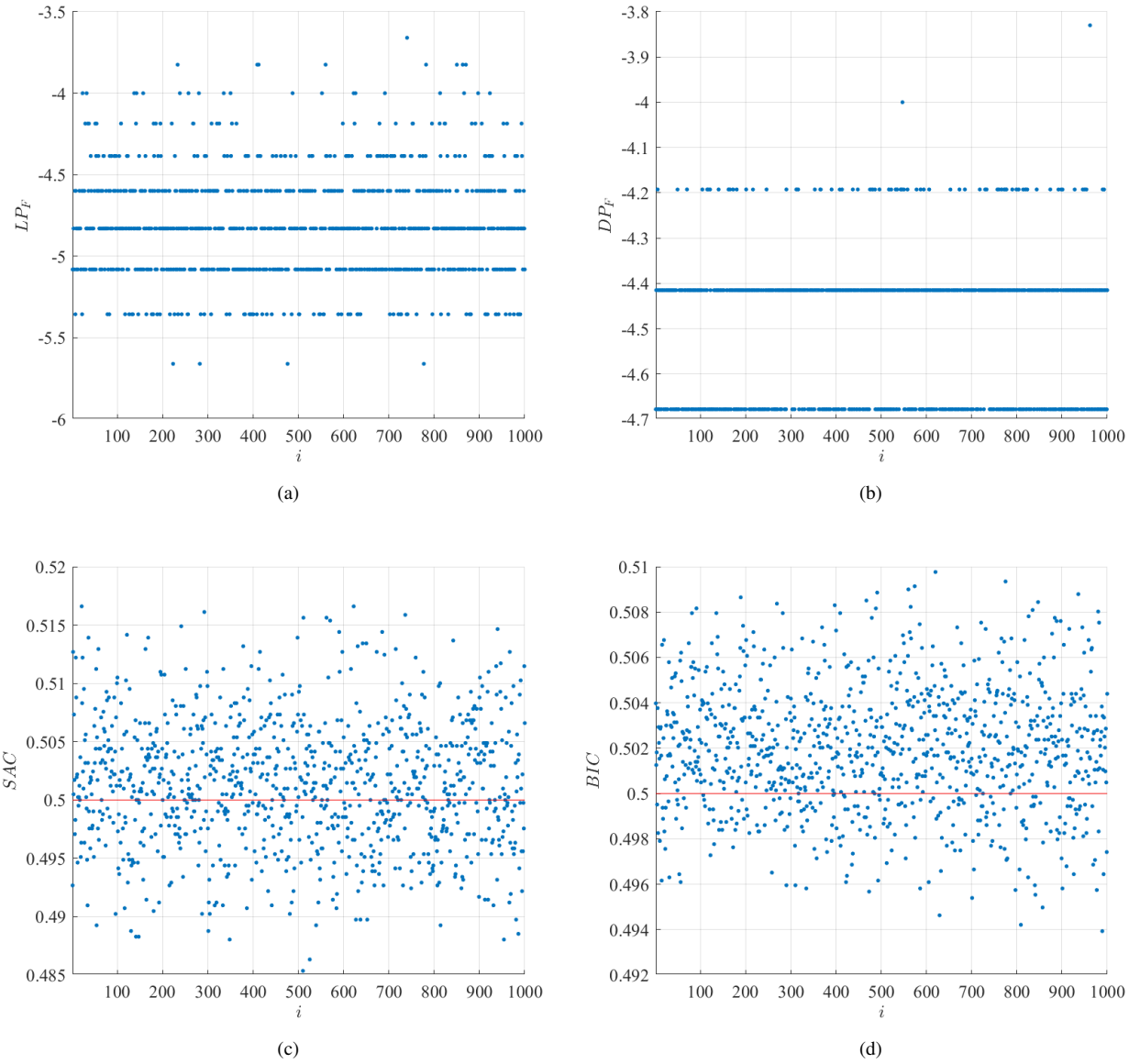


Fig. 6: Variation of the LPF (a) and DPF (b), SAC (c) and BIC (d) versus versus 1000 secret keys, where each key has 8 bytes length.

sensitivity as in Eq. (6).

$$KS_w = \frac{\sum_{k=1}^{T} S_{K_w} \oplus S_{K'_w}}{T} \times 100. \quad (6)$$

where $w = 1,\ 2,\ \ldots,\ 1000$, T is the length in bits of the S-box and it is equal to 256×8, and S_{K_w} and $S_{K'_w}$ are the corresponding S-box using K_w and K'_w, respectively. All the bits of K_w are equal to K'_w, except the least significant bit of a random byte that was flipped.

In this test, the Hamming distance is used to measure the difference at the bit level between the two produced dynamic substitution tables S-boxes ($S1$ and $S2$) using DK_S and DK', respectively. To realize these tests, 1000 different random dynamic keys are used. For each one of the 1000 dynamic substitution sub-keys, $K_w, w = 1, 2, 1000$, the percent Hamming distance is computed between two corresponding S-boxes $S1$ and $S2$. In Fig. 7-a), the percent Hamming distance over 1000 random dynamic sub-substitution keys is shown. The probability of dynamic key possessing $KS \geq 48\%$ is 99.6% and only 0.4% of the produced S-boxes possess KS between 45.8% and 48%. Besides, maximum, minimum averages and standard deviation obtained from the PD_H are: 52.83, 45.8, 49.32, respectively, and the standard deviation is 1.093. These results indicate that the percent Hamming distance between S and S' is close to 50% for overall control parameter vectors, which means that the secret key sensitivity is satisfied.

8) Number of Fixed Points: The effect of the number of Fixed Points (FP) of the permutation operation is critical

since more rounds would be needed to achieve the avalanche effect. This means that when the number of fixed points is high, a poor diffusion of the permutation table is realized as the elements (bits or bytes) in these blocks are left unchanged when producing the output blocks. Also note that the expected number of fixed points in a random permutation is one [42]. Moreover, the percentage distribution of the non-fixed points for the produced permutation table with a length equals to 64 is shown in Fig. 8-b). This shows that the number of FP is decreased, and the produced permutation tables have $FP \leq 5$ for a permutation table of 64 elements. These results indicate that the presented technique produce permutation tables with a low number of fixed points. As a summary, the proposed solution provides important properties such as flexibility and dynamicity.

In addition, the correlation coefficient between S_{K_w} and $S_{K'_w}$ allows for measuring the degree of similarity between them. In fact, the coefficient of correlation varies between -1 and 1. A correlation coefficient close to +-1, means the presence of a strong linear correlation between both functions, while a coefficient of correlation close to zero, means practically the absence of correlation between both corresponding ones. In Fig. 7-b), the correlation coefficient between S_{K_w} and $S_{K'_w}$ versus 1000 different keys is shown. The results show that the correlation coefficient is always close to zero, which indicates that no detectable correlation exists between the dynamic S-boxes, which validates the sensitivity of the dynamic secret key.

In conclusion, the proposed key-dependent dynamic substitution algorithm possesses the most suitable properties for being used in any block or stream cipher algorithm that employs a static S-box. Moreover, similar results are obtained with the proposed technique to construct dynamic permutation tables in terms of randomness, uniqueness, and key sensitivity (see Fig. 8). Let us indicate that the desired cryptographic performance of permutation tables are unique number of permutation tables in addition to a high randomness order.

V. Performance Analysis

The proposed dynamic cryptogprahic solution employs dynamic substitution and permutation tables instead of the static ones. Therefore, the proposed solution introduces a small overhead to generate these dynamic tables (primary and update cryptographic primitives) for each τ input blocks. In fact, the dynamic key generation requires a round of SHA-512 with a secret session key as input. The generated dynamic key is used to generate dynamic substitution and permutation tables, which are based on the KSA and MKSA of RC4 that exhibit linear computational complexity $O(n)$, where n is the size of the S-box for the substitution process and the number of bits/bytes per block in case of permutation. Furthermore, to reduce the initialization latency overhead (dynamic key and construction of cryptographic primitives), an update process is presented and it is based on only one operation (permute the permutation table by employing the update permutation table and substitute the substitution table by using the update substitution table) for each δ input blocks.

The required initialization time for the proposed scheme is quantified and compared to the original AES encryption time. These measures were performed on different machines such as desktop machine with Intel(R) Core(TM) i7-4910MQ @ 2.90GHz, Raspbery Pi Zero and 3 in addition to Arduino Uno. The proposed solution uses the optimized AES implementation, which uses AES-NI instructions. On the Intel machine, the standard AES initialization is very efficient since it uses only AES-NI instructions [43]. In addition, on Raspeberry Pi Zero and on Arduino Uno, a dedicated optimized C code is used since AES-NI instructions cannot be implemented. The obtained results in terms of execution times are reported in TABLE V. This table illustrates the variation of the introduced latency overhead versus the employed devices such as Arduino Uno, Raspberry Pi Zero, Raspberry Pi 3 and Intel I7. This table contains the average execution times of the construction of cryptographic primitives in addition to the execution times of the AES encryption of one block (of 16 bytes) for 1000 times. Moreover, the reported overhead ratio is computed according to the following equation:

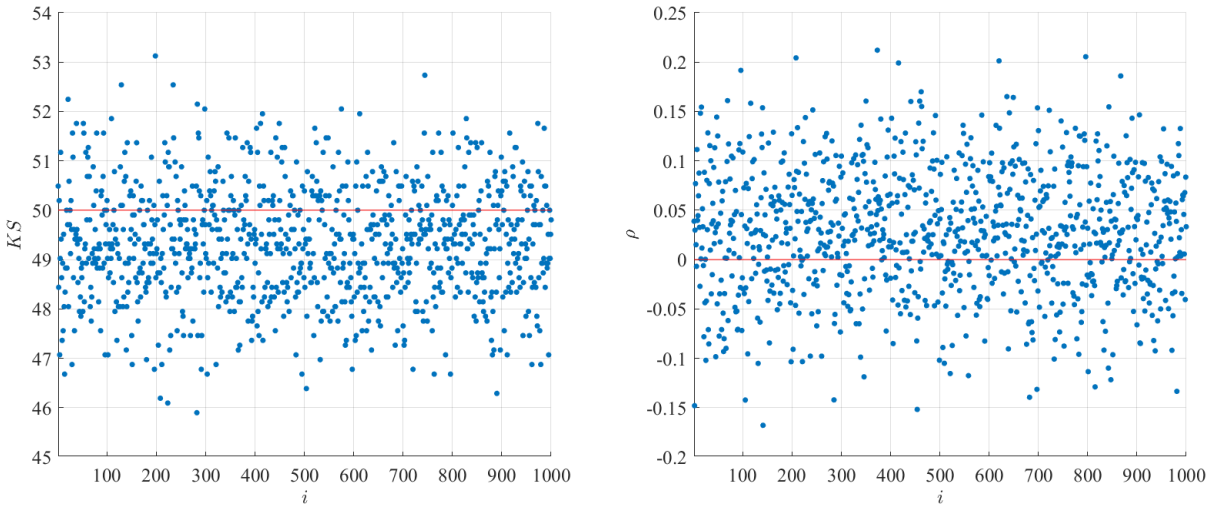
$$R = \frac{\text{Required time to produced cryptographic primitives}}{\text{Encryption time of an input block}} \times 100 \quad (7)$$

It can be seen from the results that with our proposed solution, if the number of encrypted/decrypted blocks with the same produced cryptographic primitives increases, the required latency and resources overhead decreases. Thus, the proposed solution incurs the maximum overhead when the cryptographic primitives are changed for each input data block with a maximum level of security. Furthermore, the overhead introduced by the proposed solution decreases exponentially as a function of the number of blocks nb. Moreover, the obtained results indicate that the introduced overhead by the proposed solution is lower than the one introduced by standard AES for the constrained devices such as Arduino Uno and Raspberry Pi Zero compared to Raspberry Pi 3 and Intel I7. This can be explained by the fact that Arduino Uno and Raspberry Pi Zero devices are limited and do not support hardware optimization. Differently, the powerful devices support hardware optimization and thus the overhead of standard AES is reduced to become comparable to our proposed solution's introduced overhead.

A. Memory Consumption

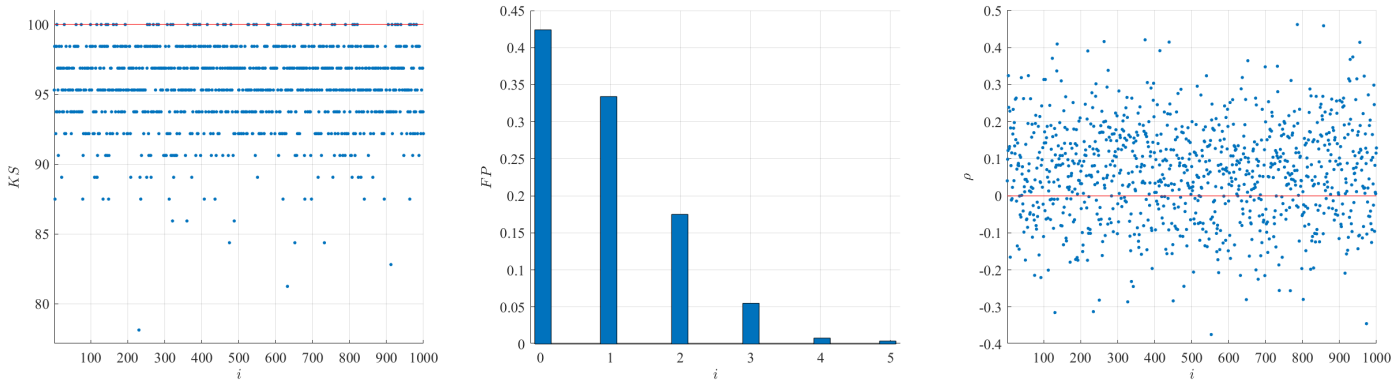
The memory overhead in the proposed dynamic key dependent cryptographic approach is imposed by the generation of two primitives:

1) An update substitution table U_S of 256 bytes length,
2) and one update permutation table U_π, having a length that depends on the number of permutation tables in the adapted cipher such as 16 in the case of AES (shiftRows).



(a) (b)

Fig. 7: (a)-(b) Variation of the key sensitivity at the bit level and the coefficient correlation $\rho\{S1,\ S2\}$ between two substitution tables ($S1$ and $S2$) versus 1000 random dynamic keys for the proposed permutation construction technique, respectively.



(a) (b) (c)

Fig. 8: Variation of the key sensitivity (a) and fixed points (b) at the element level in addition to the coefficient correlation $\rho\{P1,\ P2\}$ (c) between two permutation tables ($P1$ and $P2$) versus 1000 random dynamic keys for the proposed permutation construction technique, respectively.

TABLE V: Average Execution times in seconds of the required dynamic key generation and construction initialization and of the one block encryption on different devices in addition to its corresponding overhead ratio

Hardware	Dynamic key generation and construction initialization	Encryption of one block	R
Intel I7	6.54×10^{-8}	2.89×10^{-9}	37.7163
Raspberry Pi 3	6.66×10^{-6}	1.15×10^{-6}	9.6522
Raspberry Pi Zero	9.27×10^{-6}	3.73×10^{-6}	4.1421
Arduino Uno	0.6×10^{-3}	1.28×10^{-3}	0.7812

VI. Conclusion

In this paper, a dynamic cryptographic solution is proposed to generate dynamic key-dependent substitution and permutation tables towards reinforcing the security level of existing symmetric ciphers. This helps in reducing the required rounds number to 4 for AES, as an example, which in turn reduces the computational overhead, while providing better immunity against future powerful attacks. Indeed, for each session (or a set of messages), a dynamic key is generated based on the secret session key. This dynamic key is used to generate the dynamic cryptographic and update cryptographic primitives. The update cryptographic primitives are used to update the cryptographic primitives after a set of messages. The update mechanism of the cryptographic primitives is lightweight and

is used to achieve the required dynamicity with minimum possible overhead (only one operation that can be either permutation or substitution). The proposed dynamic primitives generation and update method achieves an acceptable trade-off between security and efficiency compared to static cryptographic primitives. Equally important, the proposed key-dependent substitution and permutation construction algorithms have been subjected to all possible cryptographic metrics that are essential for any substitution and permutation generation technique to be considered credible and robust. Simulation results show that maximum time overhead of the proposed cipher is incurred when the cryptographic primitives are changed for each data block and the introduced latency overhead is lower than the one introduced by a standard cipher scheme (AES) for limited devices.

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] Nicolas Sklavos, Ricardo Chaves, Giorgio Di Natale, and Francesco Regazzoni. Hardware security and trust. *Cham, Switzerland: Springer*, 2017.

[2] Wen Wen Koh and Chai Wen Chuah. A robust security framework with bit-flipping attack and timing attack for key derivation functions. *IET Information Security*, 2020.

[3] Morris Dworkin, Rebecca M Blank, Patrick D Gallagher, et al. Recommendation for block cipher modes of operation: Methods and techniques. In *NIST Special Publication*. Citeseer, 2001.

[4] Claude E. Shannon. Communication Theory of Secrecy Systems. *Bell Systems Technical Journal*, 28:656–715, 1949.

[5] Hassan Noura, Ali Chehab, Lama Sleem, Mohamad Noura, Raphaël Couturier, and Mohammad M Mansour. One round cipher algorithm for multimedia iot devices. *Multimedia Tools and Applications*, pages 1–31.

[6] D. Kwon, S.H. Sung, J.H. Song, and S. Park. Design of block ciphers and coding theory. *Trends in Mathematics*, 8(1):13–20, 2005.

[7] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Advanced Encryption Standard*. Alpha Press, 2009.

[8] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[9] Hichem Mrabet, Sana Belguith, Adeeb Alhomoud, and Abderrazak Jemai. A survey of iot security based on a layered architecture of sensing and data analysis. *Sensors*, 20(13):3625, 2020.

[10] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015.

[11] Kerry A McKay, Larry Bassham, Meltem Sönmez Turan, and Nicky Mouha. Report on lightweight cryptography. *NIST DRAFT NISTIR*, 8114, 2016.

[12] Like Chen and Runtong Zhang. A key-dependent cipher dsdp. In *Electronic Commerce and Security, 2008 International Symposium on*, pages 310–313. IEEE, 2008.

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

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

[15] Hassan Noura, Ali Chehab, and Raphael Couturier. Lightweight dynamic key-dependent and flexible cipher scheme for iot devices. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8. IEEE, 2019.

[16] H. N. Noura, A. Chehab, and R. Couturier. Overview of efficient symmetric cryptography: Dynamic vs static approaches. In *8th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–6, 2020.

[17] Hassan Noura, Mohamad Noura, Ola Salman, Rapha ël Couturier, and Ali Chehab. Efficient & secure image availability and content protection.

[18] Hassan N Noura, Ali Chehab, and Raphael Couturier. Efficient & secure cipher scheme with dynamic key-dependent mode of operation. *Signal Processing: Image Communication*, 78:448–464, 2019.

[19] Hassan Noura, Ola Salman, Ali Chehab, and Raphaël Couturier. Preserving data security in distributed fog computing. *Ad Hoc Networks*, 94:101937, 2019.

[20] Mohammad Noura, Hassan Noura, Ali Chehab, Mohammad M Mansour, Lama Sleem, and Raphaël Couturier. A dynamic approach for a lightweight and secure cipher for medical images. *Multimedia Tools and Applications*, 77(23):31397–31426, 2018.

[21] Axel Poschmann. Lightweight cryptography. *Ruhr-University Bochum, Bochum*, 2009.

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

[23] Hassan N Noura, Ali Chehab, Mohamad Noura, Raphaël Couturier, and Mohammad M Mansour. Lightweight, dynamic and efficient image encryption scheme. *Multimedia Tools and Applications*, pages 1–35, 2018.

[24] Liam Keliher and Henk Meijer. A new substitution-permutation network cipher using key-dependent s-boxes, 1997.

[25] Kazys Kazlauskas and Jaunius Kazlauskas. Key-dependent s-box generation in aes block cipher system. *Informatica*, 20(1):23–34, January 2009.

[26] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In Ross Anderson, editor, *Fast Software Encryption*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204. Springer Berlin Heidelberg, 1994.

[27] Piotr Mroczkowski. Generating pseudorandom s-boxes−−a method of improving the security of cryptosystems based on block ciphers, 2009.

[28] N. Masuda, G. Jakimoski, K. Aihara, and L. Kocarev. Chaotic block ciphers: from theory to practical algorithms. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 53(6):1341–1352, 2006.

[29] Guoping Tang, Xiaofeng Liao, and Yong Chen. A novel method for designing s-boxes based on chaotic maps. *Chaos, Solitons & Fractals*, 23(2):413 – 419, 2005.

[30] Guo Chen, Yong Chen, and Xiaofeng Liao. An extended method for obtaining s-boxes based on three-dimensional chaotic baker maps. *Chaos, Solitons & Fractals*, 31(3):571 – 579, 2007.

[31] Ruming Yin, Jian Yuan, Jian Wang, Xiuming Shan, and Xiqin Wang. Designing key-dependent chaotic s-box with larger key space. *Chaos, Solitons & Fractals*, 42(4):2582 – 2589, 2009.

[32] Fatih Ozkaynak and Ahmet Bedri Ozer. A method for designing strong s-boxes based on chaotic lorenz system. *Physics Letters A*, 374(36):3733 – 3738, 2010.

[33] Jun Peng, Xiaofeng Liao, and Du Zhang. A novel approach for designing dynamical s-boxes using hyperchaotic system. *Int. J. Cogn. Inform. Nat. Intell.*, 6(1):100–119, jan 2012.

[34] Hassan N Noura, Ola Salman, Ali Chehab, and Raphaël Couturier. Distlog: A distributed logging scheme for iot forensics. *Ad Hoc Networks*, 98:102061, 2020.

[35] Reem Melki, Hassan N Noura, Mohammad M Mansour, and Ali Chehab. An efficient ofdm-based encryption scheme using a dynamic key approach. *IEEE Internet of Things Journal*, 6(1):361–378, 2018.

[36] S William. Cryptography and network security: Principle and practice, ed. 7th. 2017.

[37] Hassan N Noura, Reem Melki, Ali Chehab, and Mohammad M Mansour. A physical encryption scheme for low-power wireless m2m devices: a dynamic key approach. *Mobile Networks and Applications*, 24(2):447–463, 2019.

[38] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 386–397, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[39] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In *CRYPTO'91*, 1991.

[40] A. F. Webster and Stafford E. Tavares. On the design of s-boxes. In *Advances in Cryptology*, CRYPTO '85, pages 523–534, London, UK, UK, 1986. Springer-Verlag.

[41] Kaisa Nyberg. Differentially uniform mappings for cryptography. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '93, pages 55–64, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[42] Muhammad Reza Z'aba. *Analysis of linear relationships in block ciphers*. PhD thesis, Queensland University of Technology, 2010.

[43] Nadeem Firasta, Mark Buxton, Paula Jinbo, Kaveh Nasri, and Shihjong Kuo. Intel avx: New frontiers in performance improvements and energy efficiency. *Intel white paper*, 19:20, 2008.