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

Speck-R: An Ultra Light-Weight Cryptographic Scheme for Internet of Things

Lama Sleem* · Raphaël Couturier

the date of receipt and acceptance should be inserted later

Abstract Lightweight cryptography (LWC) is an interesting research area in the field of information security. Some limitations like: increased components usage, time consumption, power consumption and memory requirement mandate the need for lightweight cryptography. One of the proposed algorithms in this field is Speck which was designed by the National Security Agency (NSA) in June 2013. In this paper, we propose a new ultra-lightweight cryptographic algorithm based on Speck known as Speck-R. Speck-R is a hybrid cipher, combining ARX architecture with a dynamic substitution layer. The novelty in this paper resides in adding a key-dynamic substitution layer that changes according to a dynamic key. With this modification, the number of rounds can be reduced from 26 (in Speck) to 7 (in Speck-R). Thus, the main contribution of this paper consists in reducing the execution time of Speck by at least 18% on limited devices to reach a reduction of 77% while keeping a high level of security. To backbone Speck-R's security, different security and statistical tests are exerted on Speck-R. In addition, a real hardware implementation on three different famous IoT devices is also presented where Speck-R outperformed Speck in terms of execution times. Finally, extensive tests show that Speck-R possesses the necessary criteria to be considered as a good cipher scheme that is suitable for lightweight devices.

Keywords: Security; Encryption; Internet of Things; Cryptography; Randomness ; Confusion.

1 Introduction

The Internet of Things has been a buzzing word in all fields (medical, academic, industry etc..) and is generally defined in different ways. Generally, it is an infrastructure of heterogeneous computing devices that communicate in spite of all the existing differences. They can communicate among each other, with people, and other services through the Internet without the intervention of humans [1,2]. Another definition was given by the European Technology Platform on Smart Systems Integration (EPoSS) which stated that the IoT is a world wide network of interconnected objects uniquely addressable, based on standard communication protocols [3]. The word *"things"* stands for the devices that link the physical and digital words while connected to the Internet [4]. These *things* usually include embedded devices that are placed in different places for different reasons. They can be placed at hospitals, universities, industries and even in our own homes. Some of the IoT devices are RFID tags, wireless sensors, actuators etc..
However, most of these aforementioned IoT devices suffer from many limitations. They are low-resource devices with limited computing power, battery (lifetime), memory and computational speed. Thus, a careful attention has to be paid to such devices especially for data processing. Having huge amount of

Sleem L.
Univ. Bourgogne Franche-Comté (UBFC), FEMTO-ST Institute, CNRS, France
Tel.: +33-07-69475406
E-mail: lama.sleem@univ-fcomte.fr
*Corresponding author

Couturier R.
Univ. Bourgogne Franche-Comté (UBFC), FEMTO-ST Institute, CNRS, France

data exchanged among all nodes of the wireless networks will lead to new risks and more challenges. A limited amount of resources is one of them (i.e. energy), and many other combination of factors that suffer from financial constraints at the end. However, the amount of resources that are dedicated to security are very low, in fact, they are just a fraction of the total available resources. As a consequence of the negligible cost for these devices, security flaws have started to emerge. Researchers started to think differently, and enhancements were added to the network and the application layers [5] to limit the effects of the threat. However, this would not eliminate the fact that the limited devices are still vulnerable to more attacks comparing them to more sophisticated devices. One of the attacks is simply draining the battery of these devices to put it in a Denial of Service state and eventually shut it down [6].
To ensure the safety, confidentiality and to stand against different kinds of attacks, the data exchanged should be encrypted. However, the conventional ways of encryption in such stringent environments do not work and will not be efficient. Therefore, lightweight cryptography was suggested to meet the requirements in such limited devices. Many research works have been targeting lightweight methods to encrypt the data in an efficient manner. For example, the National Institute of Standards and Technology has initiated a process to solicit, evaluate, and standardize lightweight cryptographic algorithms that are suitable for use in constrained environments where the performance of the current NIST cryptographic standards is not acceptable anymore [7]. However, lightweight cryptography must require less resources and must remove any computation overhead that is caused by the encryption/decryption process. To clarify and classify the lightweight cryptography, several works have been proposed. In [8], the authors differentiated the research works into (1) lightweight cryptography, (2) ultra-lightweight cryptography and (3) IoT cryptography. Another classification was presented in [2] and [9]. According to these works, it can be said that a lightweight algorithm must have the following criteria:

- Small block size: 64-bit or less.
- Small key size: 80 bits or more.
- Simple round function: Avoid heavy computations and use simple operations.
- Simple key scheduling: The key scheduling must also be simple without complications in the design.

1.1 State-of-the-art

Encryption can be mainly divided into symmetric or asymmetric algorithms. In real implementations, the symmetric-key scheme is preferred as it needs less computational complexity, memory consumption, and resources when comparing it to the asymmetric one. Symmetric ciphers can be divided into two classes: **Stream ciphers and Block ciphers**. AES is one of the most used algorithms in cryptography [10], and most of the time, it is used with the Counter mode (CTR) variant [2]. In this case, the ciphering process is independent from the plain-text, thus, the block cipher operates as a stream cipher. However, block ciphers in general use multiple rounds to reach the desired security level. Round functions themselves can be based on Feistel Networks (FN) or Substitution-Permutation Networks (SPN). The aim of increasing the number of rounds is to preserve the confusion and diffusion properties.
Until today, many efforts have been exerted to transform AES into lightweight block cipher. It needs computation power and is not practical for limited devices that suffer from many limitations. For example, in [11], a proposal aims at providing a hardware ASCI implementation of AES-128 with 2400 Gate Equivalents (GE). Efficient software AES implementations are also presented for 8-bit in [12], 16-bit in [13] and 32-bit in [14]. Additionally, AES optimized instructions were added to the instruction set of Intel's [15,16]. In [17], the authors proposed an optimization method for AES on FPGA. However, FPGA is considered more powerful than other limited devices. To our knowledge, there are no further optimization techniques to AES, and there might be no more possible optimization. In addition, according to NIST, the current optimization of AES is not sufficient for these kinds of limited devices [18]. A new project to find the best lightweight candidate has been launched indicating the required properties and parameters that should be respected (for example, having the GE less than 2000).

To wrap the limitations for AES in particular, it is necessary to take into account: (1) the complexity of the implementation itself, (2) the number of rounds needed to ensure its security (3) the presented implementation in [19] which shows that AES drains the battery on different limited platforms (ZigBee and WirelessHART). Therefore, it can be concluded that AES is not really suitable for constrained devices, such as IoT ones, as stated in [18,20,21].

Therefore, efforts are being re-directed towards finding a better solution that can be used in such constrained devices. Reducing the number of rounds, GEs, execution times, memory requirements have been the most demanded criteria for any new proposal. For example, some of the proposed works are presented in Table 1: TEA [22], XTEA [23], HIGHT [24], FeW [25], Simon [20], Speck [20], PRESENT [26], Rectangle [27], LEA [28], Prince [29], AES, and RC5 [30].

However, what all of these ciphers have in common is that they are all based on static substitution and diffusion primitives. Which means that the substitution layer of the diffusion layer does not change during the process of encryption. This will require a higher number of rounds to reach the desired level of security. Increasing the number of rounds will probably end in increasing the execution times and the memory consumption which is a limitation in such targeted hardware. Thus, there is a trade-off between the number of rounds and the execution times which will be dependent on the number of rounds denoted by r.

Therefore, using such algorithms with limited hardware abilities can prevent the device from guaranteeing its main functionality and might impact the whole system [18]. As a conclusion, proposing a new lightweight cipher that can deal with such limitations is the sole aim of this work. Based on the works previously done, Speck-R is proposed. It will ensure a low computational overhead on IoT nodes while preserving a high level of security. According to different surveys and comparisons done between different ciphers, Speck performs well on different limited hardware platforms [31]. It is one of the best performing ciphers concerning the energy consumption and the throughput on both 8-bit and 16 bits. It is also one of the best software efficient ciphers among more than 50 different tested algorithms. For this reason, we considered enhancing the performance of the Speck, since it possesses all the necessary values for a lightweight cipher. Thus, it can be said by convention that having a better cipher than Speck is the same as having a better cipher than all the other proposed ciphers (already compared in [31]). In addition to that, the urge for updating Speck rises from the fact that the reduced version of Speck [32] has been attacked by either differential fault attacks or linear attacks [33, 34, 35].

Table 1: List of some lightweight cryptographic algorithms

Algorithm	**No. of rounds**	**Key size**	**Block size**	**Structure**
TEA [22]	64	128	64	FN
XTEA [23]	64	128	64	FN
HIGHT [24]	32	128	64	GFS
FeW [25]	32	80/128	64	FN-M
Simon [20]	32/36/42/44/52/54/68/69/72	64/72/96/128/144/192/256	32/48/64/92/128	FNl
Speck	22/23/26/27/28/29/32/33/34	64/72/96/128/144/192/256	32/48/64/92/128	FN
PRESENT [26]	31	80/128	64	SPN
RECTANGLE [27]	25	80/120	64	SPN
LEA [28]	24/28/32	128/192/256	128	FN
Prince [29]	11	128	64	SPN
AES	10/12/14	128/192/256	128	SPN
RC5 [30]	12	128	32/64/128	FN

1.2 Main Contribution

In this paper, we propose a new reduced ultra-lightweight cipher based on Speck and we name it **Speck-R**. Speck is one of the famous ARX (Addition/Rotation/XOR) lightweight ciphers which was proposed by the US National Security Agency (NSA) in 2013. Its aim is to offer security in constrained devices. Besides, it is well-known for its fast execution time, its security and the simple operations it uses. Speck-R is a hybrid cipher that has more or less the same properties as Speck, but with an added twist its

"Dynamicity" that helps overcoming the differential and linear attacks that previously succeeded on reduced versions of Speck. In Speck-R, a dynamic key-substitution layer was added into the structure of the original Speck. Based on that, an **enhancement/reduction of the Speck algorithm** is proposed, that results in a lower number of rounds (7) in Speck-R compared to Speck (26). The main contribution here is enhancing the execution time of Speck, in the new version Speck-R, which ranges between 18% and 77% depending on the hardware used while keeping a high security level. The new proposed cipher ensures (a) the confidentiality of the transmitted/stored data content in a robust way to protect it against attacks and (b) maintain a fast execution time in order to cope with the advanced demands of new devices. Finally, extensive security tests have been exerted to validate that Speck-R possesses all the necessary requirements that makes it a good lightweight cipher candidate.

1.3 Organization

This paper is divided as follows. The main features of the proposed image encryption algorithm are described in Section 2. Then in Section 3, a deeper look into Speck and its variants is presented. After that, Section 4 discusses the proposed cipher. Then, the added cryptographic layer is explained in Section 5 and specific tests are exerted to prove the robustness of the added substitution layer. After that, randomness tests done using Practrand are explained in Section 6. Security results that have been conducted to evaluate the efficiency of this algorithm are also explained in Section 7. Then in Section 8, a performance analysis of Speck-R is presented and a comparison with the original Speck is given. A discussion about the efficiency of the proposed algorithm against the best known types of attacks is investigated in Section 9. Finally, Section 10 ends with a brief conclusion resuming the work.

2 Features of The Proposed Approach Speck-R

Before digging deep into the original versions of Speck, the goals of the new updated Speck are described. This algorithm will be called Speck-R, the "R" standing for "Reduced". Speck-R's two main contributions, compared to Speck, are a higher efficiency and an increased level of security. Below, the desired **system performance and security performance** are described.

System Performance:

- Lightweight: The minimum required number of iterations, for recent lightweight cryptographic algorithms, is 4 such as the Hummingbird2 cipher [36]. For Speck, the minimum number of rounds is 22. In fact, Speck is based on ARX (Addition/Rotation/XOR) which is a class of cryptographic algorithms that has three simple arithmetic operations: namely modular addition, bitwise rotation and exclusive-OR. In both industry and academia, the ARX cipher has been gaining a lot of interest and attention in the last few years. By using combined linear (XOR, bit shift, bit rotation) and non-linear (modular addition) operations and iterating them for many rounds, ARX algorithms have become more resistant against differential and linear cryptanalysis. In this proposal, we aim at adding a dynamic substitution layer that increases the security of the cipher, yet keeps it ultra-weight. The proposed cipher avoids using a static diffusion operation such as the MixColumn transformation of AES [37] or the key-dependent integer/binary diffusion operations of [38,39], since such operations consume a high percentage of the execution time [39,40]. Moreover, Speck-R is realized in CTR mode, thus it can be processed in parallel, whether for encryption or decryption. CTR mode decreases the latency and enables a fast execution time.
- Flexibility: As the original Speck, it operates at the block-level, which can have a flexible number of bits exactly as the original cipher. This chosen block size can be set according to the user's requirements and the network abilities. The proposed approach is set according to the devices' characteristics, while taking into account the block size used in lightweight ciphers.
- Simple hardware and software implementations: As stated earlier, ARX ciphers are easy to implement and are highly recommended for small, limited devices, especially those dedicated to the IoT. This makes the corresponding hardware and software implementations simple and efficient.

- Low error propagation: In this proposal, each block is treated once at a time. The block is split into two parts, semi-blocks, thus, any error occurring in a block, will only affect the block itself. It will not affect the whole blocks in the image and the error will not propagate across the whole image/data. Speck-R is designed to be in the CTR mode, thus avoiding any chaining process that itself propagates any error across the system. Thus, low error propagation is guaranteed.
- Large key space: Since the original Speck has different versions using different key sizes, the key can range between 64 and 256 bits. Therefore, adapting the same criteria of Speck, Speck-R is resilient against brute-force attacks according to [41].

These enhancements added to the cipher reduce the delay of the encryption and decryption processes and simplify their corresponding hardware implementations. Every primitive in this proposal has its own impact on the security and efficiency of the proposed cipher scheme.

Security Performance:

- Key dependent approach: Speck-R is based on key-dependent substitution primitive that ensures the simplicity of key scheduling in addition to the required cryptographic properties.
- Dynamic key approach: Speck has already proven to be a secure cipher that possesses a secure key. A dynamic substitution layer is added to the cipher and it changes according to the number of iterations the cipher undergoes. The substitution layer is set to be dynamic, which means that it is built according to a previously chosen key. In contrast to the existing cipher solutions, the proposed approach is based on a dynamic key, which is variable and changes in a pseudo-random manner for each new session. The periodic interval of a session depends on the application or on user requirements. For example, a new session can be established for each new input image. Therefore, the cryptanalysis process against the proposed cipher is very challenging because of the unpredictability of the cipher primitive as it changes according to the dynamic key. Changing the key each time results in a different substitution layer that will change itself with the number of iterations. Adding a dynamic layer did not only result in having a more secure cipher, but it also reduced the number of iterations from 26 to 7 which is the main goal behind this proposal.
- Speck original security: Until 2018, there were no published "attacks" on full-round Speck but only on the reduced-round variety. These kinds of attacks aim at finding the maximum number of rounds that will make Speck susceptible to theoretical attacks. According to Speck's designers, the cipher is designed to be resilient against standard chosen-plaintext and chosen-ciphertext attacks as well as related-key attacks. Let us take the total number of rounds that have been attacked, as a percentage of the total number of rounds. As for 2018, there are no published works that attack more than 70-75% of the number of rounds through Speck. More than 70 papers have been published, the best are 19 of 27 rounds for Speck 64/128. (70.3%) [33]. According to Speck's original designers, they made a trade-off between the desired level of security and the efficiency of the cipher, thus, it can be said that Speck-R has the same properties as Speck. Based on the analysis done, stepping to appropriately balance efficiency and security has been reached. It can also be noted that the number of rounds considered in the Speck were based upon making it robust against differential attacks. They set the number of rounds to leave a security margin similar to AES-128's at approximately 30% [42].

Accordingly, Speck-R is meant to be a good lightweight and a flexible cipher candidate based on Speck. This is justified since the trade-off between system performance and the security level is reduced in addition to its simple hardware and software implementations.

3 Deeper into Speck

Speck and Simon were proposed by NIST in 2013 after seeing that traditional cryptography was no longer well-suited for the emerging reality. Speck and Simon are both block ciphers proposed to address the challenges in the constrained devices. They were first proposed by a group of researchers in 2013 [43] and many crypto-analysts have since then been trying to prove that these new algorithms tend to be secure. Their results confirmed that Simon and Speck are both secure. The major issue is that most of the proposed lightweight ciphers lack the main criterion which is **flexibility**. This is typically what Speck

and Simon aimed at. In fact, heterogeneous networks connect nowadays millions of small devices, thus, the main aim is to ensure that the cipher will work properly and efficiently anywhere and on any device. After all, we do not know what sort of new devices will exist in 2030. However, regardless of what the device is capable of doing, it will surely support simple operations based on AND, OR, and XOR. These operations are performed efficiently on small devices like FPGAs, since they help any proposed cipher to improve its performance. For example, PRESENT [26] which succeeded on ASIC did not perform well on constrained devices. Additionally, most of the proposed ciphers had fixed block size and key size. Additional flexibility is needed here, therefore Speck and Simon proposed using different sizes of blocks and keys. In this work, Speck was chosen over Simon, since it is more efficient in terms of software efficiency than the latter. Speck uses modular addition for its non-linearity, which is stronger in terms of cryptography than the Simon's AND operation and is better suited in software implementation. In Speck, Feistel structure was generalized containing five different block sizes of 32, 48, 64, 96 and 128 which can be further divided into ten variants along with size key used. However, the flexibility and simple design ended up with algorithms that have exceptional performance on high-end platforms as well. Speck has the highest throughput on 64-bit processors of any block cipher implemented in software.

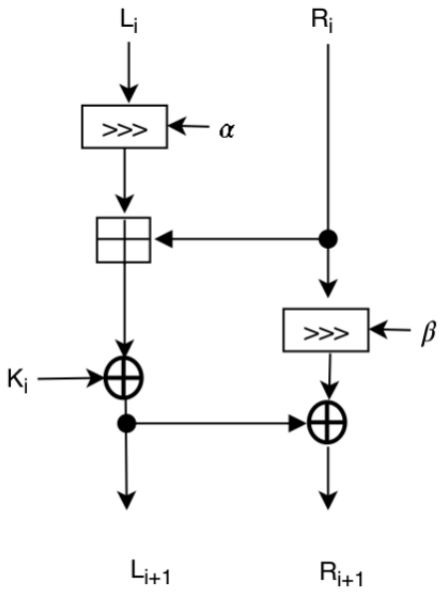


Fig. 1: A representation for the general round of Speck cipher.

The round function consists of XOR, modulo addition and rotation operations. Below, a general round function of Speck is demonstrated in Figure 1 where L_i and R_i are the left and right half intermediate values respectively of the input for the i_{th} iteration. K_i is the n bit key used in the i_{th} round, $>>>\alpha$ and $<<<\beta$ denote circular right and left shift by α or β bits, $\bigoplus$ is the XOR operation and $\boxplus$ is the modulo addition. The outputs for the i_{th} round are L_{i+1} and R_{i+1} and the round function can be described as follows:

$$L_{i+1} = ((L_i >>> \alpha) \boxplus R)) \oplus k_i \quad and \quad R_{i+1} = (R_i <<< \beta) \oplus L_{i+1}.$$

3.1 The different versions of Speck

In this subsection, the five different variants of the Speck family are represented. The rotation parameters (α, β) for Speck are either (7,2) or (8,3). In fact, Speck is usually denoted by Speck2n/mn where 2n is the block size and n $\in$ 16, 24, 32, 48, 64 and mn resembles the size of the key used where m $\in$ belongs to 2, 3, 4 depending on the desired security. Concerning the key schedule for the Speck family, the key used is 2, 3 or 4 words. The key schedule expands the initial m-word master key (l_{m-2},..., l_0, k_0) into i_{th} number of rounds (k_0, k_1,..., $k_{i_{th}}$), then two sequences are generated of words k_i and l_i according to the following algorithm:

$$l_{i+m-1} = ((k_i \boxplus (l_i >>> \alpha)) \oplus i \;\; and \;\; k_{i+1} = (k_i <<< \beta) \oplus l_{i+m-1}.$$

In Table 2, the different versions are represented. It can be clearly seen that the round function of Speck can have different key sizes and blocks and the number of rounds depends on the key used. In this work, the version of Speck64/96 that operates in CTR mode with 26 rounds was chosen. In Figure 2 a detailed scheme of the round function with the key schedule is represented.

Block size (bits)	Key size (bits)	α	β	Number of Rounds
32	64	7	2	22
48	72	8	3	22
48	96	8	3	23
64	96	8	3	26
64	128	8	3	27
96	96	8	3	28
96	144	8	3	29
128	128	8	3	32
128	192	8	3	33
128	256	8	3	34

Table 2: Different parameters of the Speck family.

4 The Proposed Speck-R

In this section, the proposed cipher algorithm is presented. First, the concepts that are used in this algorithm are explained, then, the updates added to the original cipher are presented. Lastly, some details about the core of the ciphering layers used are added.

4.1 Key derivation Speck-R:

As can be seen, Speck lacks any substitution operation according to what its authors wanted, to keep the algorithm as simple as possible. However, in our opinion, when iterating for 22 rounds or more (in our case 26 rounds), this can be reduced to minimize the execution time. The aim is to optimize the number of rounds used in the cipher as much as possible.

The proposition falls within the symmetric key schemes where both communicating parties (sender, receiver) share the same key. The main advantage behind this is the low complexity compared to the public key schemes. Parameters that are needed in Speck-R are: a Nonce N, a Dynamic key DK and the Key K. All the notations used are shown in Table 3 and the initialization phase is demonstrated in Figure 3. These steps are sufficient to preserve high sensitivity since a little change will lead to completely different parameters and substitution tables.

– Initialization Function: It is a function used to generate the seed that will be used later on in the pseudo-random generators. It can be any function the user defines, a hash function (like SHA-512), or a key stream cipher. A hash function will be useful here, to avoid any collisions, but it will be expensive in terms of time. In this work, a pseudo random generator that is simple but yet known to

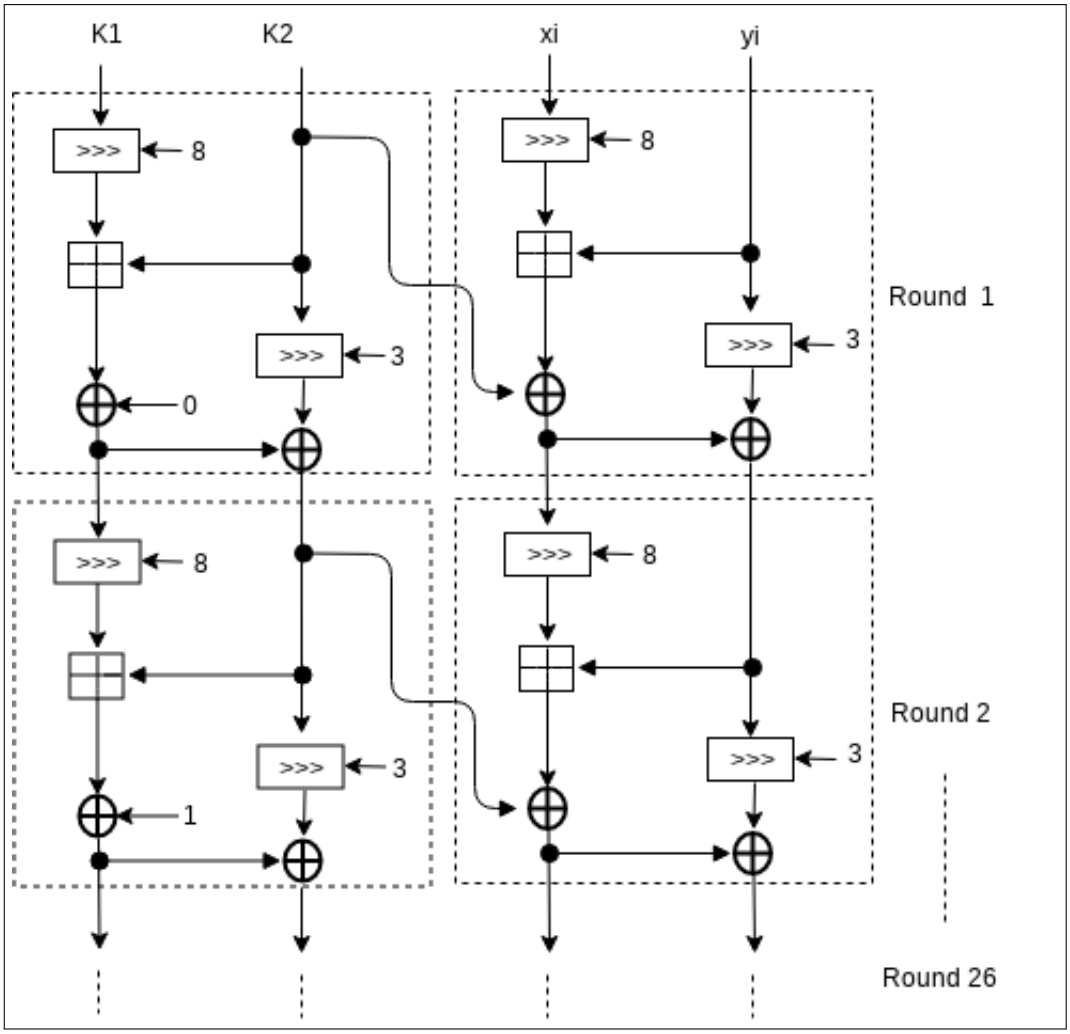


Fig. 2: The original Speck64/96 cipher.

be efficient is used, which is Splitmix64. It is a split-table pseudo-random number generator that is based on object-oriented arithmetical and logical operators [44].

- Nonce: Denoted by N, which is needed in any counter mode cipher. A pseudo-random generator is used to generate this Nonce from a seed. It is important to generate a new Nonce for each input image. N can be sent to the receiver encrypted using the shared public key of the other entity if the asymmetric approach is used. Another way for sharing N is to have a good synchronization between the sender and the receiver where each entity derives separately with no need for transmission and starting from the same seed. In Speck-R, Nonce has a length of 64 bits (8 bytes). The 64 bits are split into parts with 32 bits, where the first 32 bits represent N[0] and the second 32 bits represent N[1]. After the generation of the Nonce, N[1] will remain the same, whereas N[0] will be incremented by 1 after every iteration (i.e. from one block to another).
- Key: Denoted as K, which will be used as the main key to extract the round function keys, just as Speck. A pseudo-random generator is used to generate K. The same key schedule as in Speck is used, that is, the key generated which is 96 bits will be expanded. In every round of Speck-R a unique key will be used which is derived from K. However, in the original version of Speck, there were 26 round different keys, while in the proposed version there are only 7 round keys, each of 32 bits, denoted by K_r.
- Dynamic Key: Denoted as DK is also generated by using a pseudo-random generator. It has a length of 256 bytes, which will be later on used to generate three different substitution boxes that are used in the encryption process.

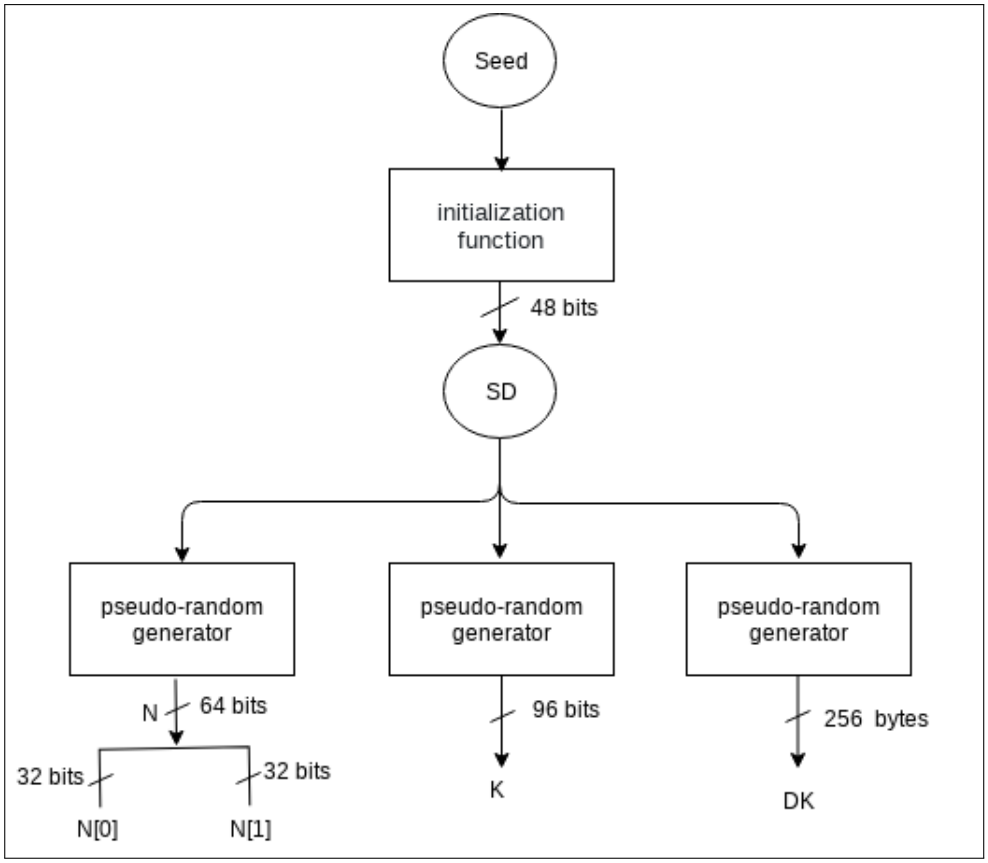


Fig. 3: Keys and parameters required in the proposed Speck-R.

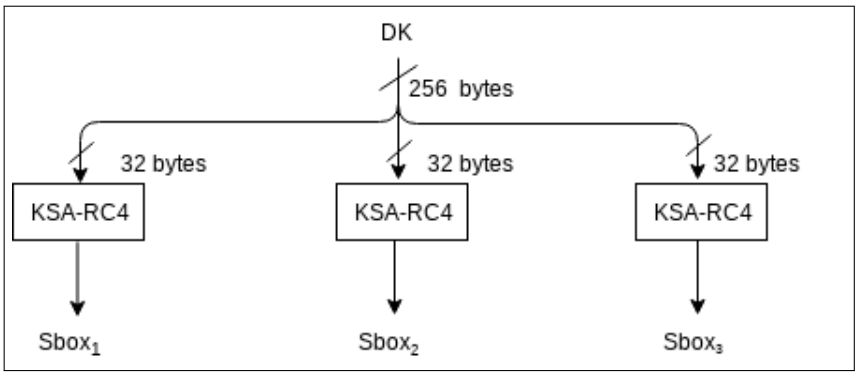


Fig. 4: A high level scheme for the Sboxes generation.

– Substitution Boxes: Denoted as $Sbox_1$, $Sbox_2$, $Sbox_3$. The cryptographic strength of three Sboxes will be explained more clearly in Section 5. However, to generate these three Sboxes RC4 will be used. RC4 is not used here in the context of a stream cipher, that mixes the plain text with the output key stream. It is iterated according to the DK previously produced to generate three robust Sboxes. RC4 is used since it is well known for its simple hardware and software implementation. Key Setup Algorithm (KSA) which is the initialization phase of RC4 is used specifically to generate the three dynamic Sboxes. This is demonstrated in Figure 4, and the algorithm is shown in Algorithm1.

Algorithm 1 KSA for RC4

procedure RC4_KSA($K = \{k_1, k_2, \ldots, k_L\}, L$)
  for $i \leftarrow 0$ to 255 **do**
    $S[i] \leftarrow i$
  end for
  $j \leftarrow 0$
  for $i \leftarrow 0$ to 255 **do**
    $j \leftarrow (j + S[i] + k[j \bmod L]) \bmod 256$
    $swap(S[i], S[j])$
  end for
  return S
end procedure

Table 3: Summary of the notations used.

Notation	Definition
$Seed$	Seed used as an input for an initialization function
SD	Seed used as an input for the pseudo-random generators
N	Nonce 64 bits
$N[0]$ *or* N_R	First 32 bits of N
$N[1]$ *or* N_L	Second 32 bits of N
K	Key of 96 bits used in Key schedule of Speck-R
K_r	The key used in every round
DK	Dynamic key of 256 bytes used to build $Sbox_1$, $Sbox_2$, $Sbox_3$
$Sbox_1$	The first produced dynamic substitution table
$Sbox_2$	The second produced dynamic substitution table
$Sbox_3$	The third produced dynamic substitution table
Seq_i	The plain block at index i
X_L	The encrypted N_L
Y_R	The encrypted N_R
n	Number of bytes in the block
nb	Number of blocks present in the plain message
M	Number of columns of an image
N	Number of rows of an image
P	Number of plane (in gray-scale P=1)

4.2 Encryption Process

In general, Speck can operate in different encryption modes i.e. ECB, CTR, CBC, PCBC, CFB and OFB. The proposed Speck-R operates in CTR mode, however, it can be executed in other modes. Speck-R can be used for the encryption of any kind of data whether texts or images etc.. In the case of image encryption, the image is of size $M \times N \times P$ where M is the columns number, N is the rows number and P is the plane number (for grey-scale is equal to 1). Image is stored using Pixmap that stores and displays a graphical image as a rectangular array of pixel color values [45]. The general encryption process in the CTR mode is displayed in Figure 5. As can be seen, Speck-R takes one block Nonce of 64 bits, and this Nonce is split into two blocks of 32 bits. The result of the encrypted 32 bits N[0] and N[1] after passing through Speck-R will be xored with 32 bits block of the input. The encryption continues to cover all the blocks in the original plain-text/image. Let n represent the number of bytes chosen in the block, according to each version of Speck, then the total number of blocks nb will be equal to

$$nb = \lceil \frac{(M \times N \times P)}{n} \rceil \tag{1}$$

where the index of blocks i, will lie between $i \in \{0, 1, ..., nb\}$.

In the rest of this work, the size of the chosen block is 64 bits, and the key is chosen to be 96 bits. In Figure 6 a closer view on the round of Speck-R is represented. The proposed encryption algorithm can be divided into three major operations which are, (1) Encrypting the Nonce, (2) Passing across the substitution layer, and (3) Xoring the plain text with the resulted substituted value. The following operations are explained below.

1. Encrypting the Nonce: After the generation of N, it will be divided into two smaller blocks, $N[0]$ and $N[1]$ that are denoted by N_R and N_L. Each block will undergo the same round function of Speck that is represented by the following equations:

$$X_L = ((N_L >>> 8) \boxplus N_R)) \oplus K_r \tag{2}$$

$$Y_R = (N_R <<< 3) \oplus X_L. \tag{3}$$

As can be seen, this step is just an ordinary Speck round, where K_r, is the key to be used for every round and $K_r \in \{ K_1, K_2, K_3, K_4, K_5, K_6, K_7 \}$. This key is generated by using the key expansion method used in Speck.

After the encryption of N_L and N_R, Speck-R swaps the two outputs X_L and Y_R. That is to say that X_L will take the place of Y_R and vice versa. This will increase the randomness in the encrypted nonce and will lower the probability of any sequential relation with the next block, if it exists. $N[1]$ will remain constant throughout all the blocks, however, $N[0]$ will be incremented by 1 from one block to another. That is to change the Nonce value from one block to another, which adds more immunity to the cipher. Many ciphers in CTR mode use a static Nonce whereas it is not the case in Speck-R.

2. Substitution Layer:
The main component of Speck-R is the added substitution dynamic layer. As explained earlier, the DK will be used to generate three different substitution boxes ($Sbox_1$, $Sbox_2$, $Sbox_3$), by using the Key scheduling Algorithm (KSA) for RC4. In fact, a substitution table is a non-linear component added to the cipher to reach the confusion property. Let us then use a dynamic substitution layer that is built upon a dynamic key that changes for every input. At the beginning of the encryption, two counters are initialized, $it_1 = 0$ and $it_2 = 0$. The counters will be incremented in every round by one element. When it_1 reaches 2000, then the substitution table $Sbox_1$ will be replaced by the resultant of the substitution operation of $Sbox_1$ by $Sbox_2$. In other words, by using $Sbox_2$, $Sbox_1$ will undergo a substitution operation. This can be represented as $Sbox_1 = Sbox_2[Sbox_1]$. Then, it_1 is set back to 0. If the number of blocks was very large, and it_2 reaches a value of 2000×2000, then, $Sbox_2$, will be subjected to a substitution operation, using $Sbox_3$. This will be represented as $Sbox_2 = Sbox_3[Sbox_2]$. The following steps are explained in Algorithm 2. The main aim of adding this layer is to decrease the round number of Speck from 26 to 7 in Speck-R.

3. Xor the plain-text: As in any CTR cipher, the ciphered final result, is the plain text xored with the encrypted Nonces/counters. After passing through the substitution layer, the plain text/image will be divided into blocks each of 64 bits. Then, the first and second 32 bits, $Seq_{i,L}$ and $Seq_{i,R}$, will be xored with the substituted values of Y_R and X_L, respectively. This is represented by the following equation:

$$Out_{i,L} = Seq_{i,L} \oplus Sbox_1\ [Y_R]; \tag{4}$$

$$Out_{i,R} = Seq_{i,R} \oplus Sbox_1\ [X_L]; \tag{5}$$

Finally, to get the whole result, all the blocks are concatenated, and aligned using the Pixmap. The encryption is simple, efficient, dynamic and easy to be implemented. As a conclusion, this is a simple cipher that reaches the confusion and diffusion properties with just 7 rounds via a dynamic key dependent substitution layer. The efficiency and robustness are demonstrated in the following sections. The whole encryption algorithm can be summarized in Algorithm 3.

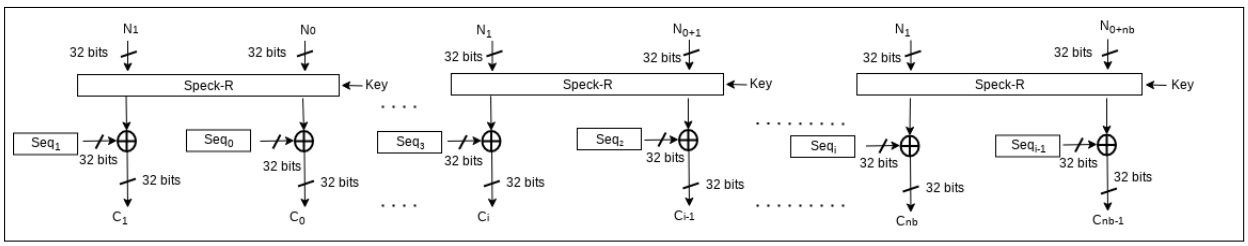


Fig. 5: The general scheme of the proposed cipher Speck-R.

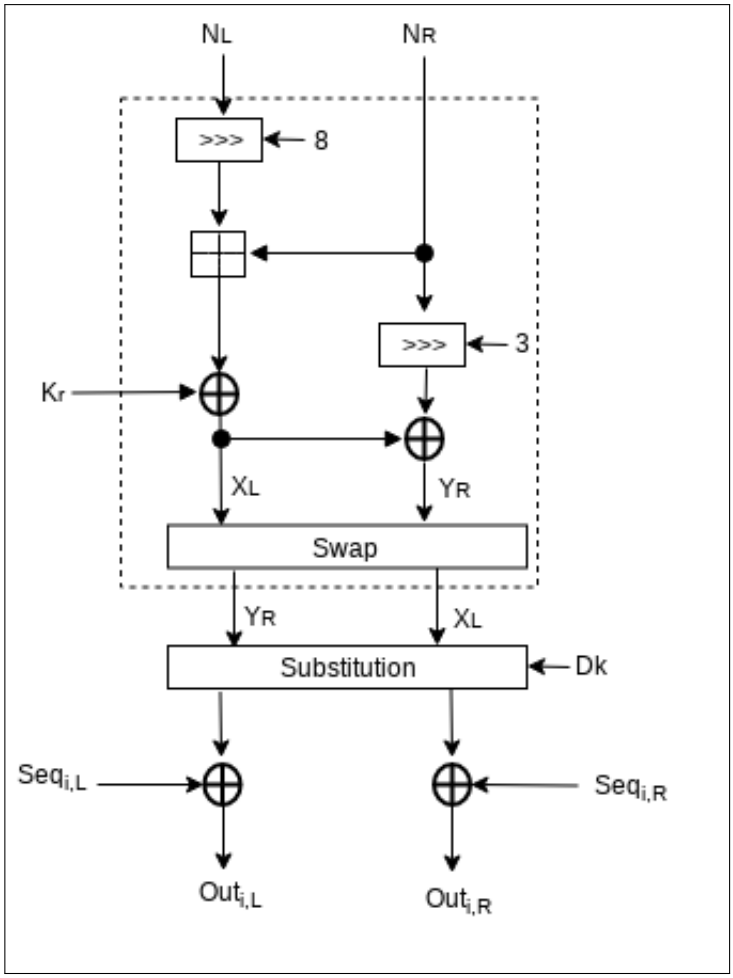


Fig. 6: The proposed encryption round of the cipher Speck-R.

Algorithm 2 Dynamic Substitution Layer

```
procedure SUBSTITUTION({Sbox_1 , Sbox_2 , Sbox_3})
    for i ← 0 to nb do
        Encrypt(Block [i])
        it_1 ← it_1 + 1
        it_2 ← it_2 + 1
        if it_1 =2000 then
            Sbox_1 ← Sbox_2 [Sbox_1]
            it_1 ← 0
            if it_2 =2000 × 2000 then
                Sbox_2 ← Sbox_3 [Sbox_2]
                it_2 ← 0
            end if
        end if
    end for
end procedure
```

4.3 Decryption Process

In a similar way to the encryption process, the decryption process needs 7 rounds. The ciphered scheme will be xored with encrypted Nonces. There is no need to use an inverse substitution layer, since the same substitution tables will be used. The decryption process is exactly the same as the encryption which gives a great advantage to the scheme in terms of software/hardware implementation. No complicated re-evaluation for the inverse substitution tables is needed here. The decryption process is shown in Figure 7. In the next sections, the robustness of Speck-R will be proved and extensive tests are executed.

Algorithm 3 Encryption Process of Speck-R

procedure SPECK-R_ENCRYPTION(Seq)
  $N \leftarrow PRNG\ (seed)$
  $K \leftarrow PRNG\ (seed)$
  $K_r \leftarrow Key\ Expansion$
  $DK \leftarrow PRNG\ (seed)$
  $Seq\ [nb] \leftarrow plaintext$
  $Sbox_1,\ Sbox_2, Sbox_3 \leftarrow RC4-KSA\ Initialization\ Algorithm$
  for $i \leftarrow 0$ to nb **do**
    $X_L = ((N_L >>> 8) \boxplus N_R)) \oplus K_r$
    $Y_R = (N_R <<< 3) \oplus X_L$
    $Swap(X_L, Y_R)$
    $N_L \leftarrow N_L$
    $N_R \leftarrow N_R\ ++$
    $Out_{i,L} \leftarrow Sbox_1\ [Y_R] \oplus Seq_{[i],L}$
    $Out_{i,R} \leftarrow Sbox_1\ [X_L] \oplus Seq_{[i],R}$
    $Update\ Sbox_1 \leftarrow Substitution\ (Sbox_1,\ Sbox_2,\ Sbox_3)\ Algorithm$
  end for
end procedure

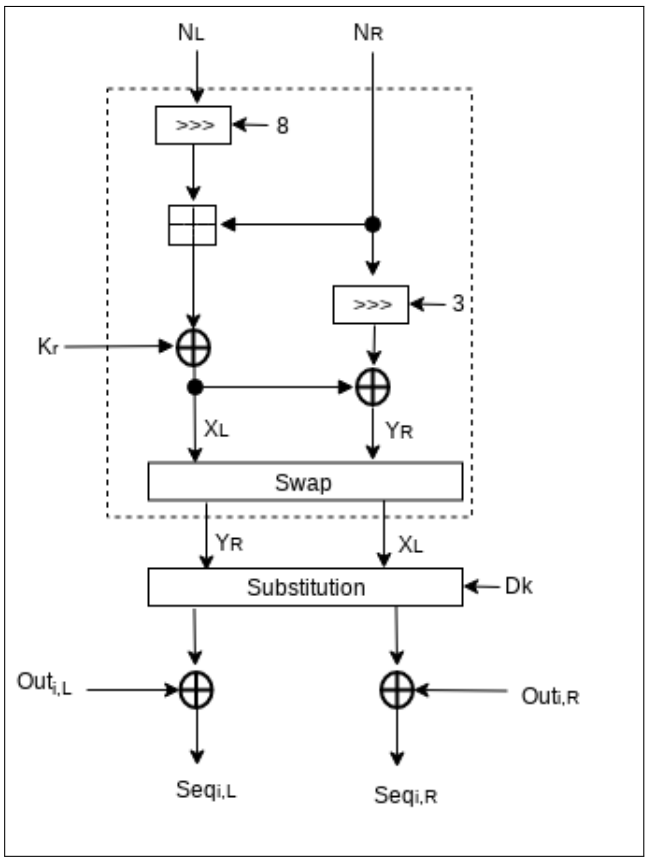


Fig. 7: The proposed decryption round of the cipher Speck-R.

5 Cryptographic Strength of Cipher Layers

In this section, we evaluate the performance of the proposed dynamic layer. In general, the substitution operation is used to ensure the confusion property and to introduce non-linearity in any cipher scheme. The proposed cipher needs three substitution tables: $Sbox_1$, $Sbox_2$, and $Sbox_3$. Mainly, $Sbox_1$ is used directly onto data encrypted/decrypted, while the other two *Sboxes* are used to manipulate the first Sbox. $Sbox_2$ is used to change $Sbox_1$ after a predefined number of iterations and $Sbox_3$ is used to alter $Sbox_2$ after another predefined number of iterations. As mentioned before, the initialization phase of RC4-KSA, is used to generate the dynamic substitution layer [46]. It is described in Algorithm 1, where

the dynamic input key DK with length L bytes is introduced to produce the three substitution tables. In the work presented, the first 32 bytes are taken as input to produce $Sbox_1$, the second 32 bytes produce $Sbox_2$ and the third 32 bytes of DK results in $Sbox_3$. The size of the produced substitution tables is 256 elements, which is 32 bytes. However, to demonstrate a strong substitution layer, based on information theory analysis [47,48,49], four main properties have to be insured which are (a) Linear Probability approximation Boolean Function (LPF), (b) Differential Probability approximation Function (DPF), (c) Strict Avalanche Criterion (SAC) and (d) output Bits Independence Criterion (BIC).

5.1 Linear Probability Approximation Boolean Function (LPF)

LPF was first introduced in [48], in the proposition of a linear cryptanalysis for DES block cipher. The basic idea behind it, is to find a linear relation or approximation that relates some bits of the plain-text with its corresponding ciphered ones. Finding a linear relation between the plaintext and the ciphertext will make the key exposed and easier to be extracted.

This means that the substitution layer's immunity is directly related to the uniformity of the LPF. The lower the value of the LPF is, the higher the complexity of linear attacks and vice versa. As an example, AES cipher has an LPF of $2^{-6} = 0.015625$.
In the proposed cipher, LPF was tested to prove that a low probability exists [48]. To reach a better resistance against linear attacks, LPF should be very low. In order to evaluate the required number of necessary iterations to reach the lowest LPF value, LPF values versus the number of iterations were tested. For each iteration, the computed number corresponds to the mean of 1000 tested sub-matrices. Results showed that after 4 iteration, LPF stabilizes and reaches its minimum which is $2^{-4.8} = 0.035897$. Consequently, it can be said that the substitution layer becomes immune against linear attacks after 4 iterations.

5.2 Differential Probability Approximation Function (DPF)

Differential Probability is one of the important properties of any substitution layer to obtain the nonlinear transformation, and hence to resist differential cryptanalysis attacks [47]. In fact, this criterion studies the effect of a slight change in plaintext pairs on the corresponding ciphertext pairs. The cryptanalyst in this attack tries to leverage the high probability of occurrence that appears in the difference of two plaintexts. The substitution layer must have differential uniformity.

In this work, DPF was calculated versus different number of iterations. For each iteration, the computed number corresponds to the mean of 1000 tested sub-matrices. Results showed that to provide a better resistance against differential attacks, minimum 4 iterations are needed so that the average of DPF converges to the minimum possible value $2^{-4.5} = 0.044194$.

5.3 Strict Avalanche Criterion (SAC)

Webster and Tavares were the first to present SAC when they generalized the avalanche effect [49]. Referring to Shannon, an efficient cipher must ensure confusion and diffusion properties. That is to say, a cipher system function is satisfying SAC whenever a single input bit is complemented, the output bit should be changed at least with a probability of half.

In this work, we targeted 1000 sub-matrices to check if the substitution level reaches the SAC criterion or not. The result obtained is that after 4 iterations, the produced *Sboxes* become closer to the ideal value. Thus, it can be stated that after 4 iterations the cipher will be sensitive to any bit toggling and therefore, the avalanche effect is ensured in the level of substitution.

5.4 Output Bit Independence Criterion (BIC)

This criterion measures the level of dependence of the output bits, after they undergo the substitution process defined by [50,49]. According to this criterion, the inversion of an input bit p modifies output

bits q and r without any dependence on each other. An S-box that makes the output bits independent from each other strengthens the security.

To prove that Speck-R meets the BIC criterion, 1000 different sub-matrix were used and after four iterations, the BIC becomes very close to the desired value 0.5. This literally means that the two output bits j and k for each substituted bytes, will change independently if a single bit i is changed. Hence, under this value, the proposed substitution layer becomes immune against chosen plaintext/ciphertext attacks.

All the evaluated criteria show that four iterations are needed to reach the desired cryptographic strength. All the previous calculated values are presented in Table 4, where a comparison with the AES substitution table is made. The results showed that the proposed substitution layer possesses sufficient cryptographic performances and the obtained results of LPF, DPF, SAC and BIC are very close to the standardized solutions. The cryptographic security of the scheme relies on the property of using a new dynamic efficient substitution layer.

Test	Speck-R	AES
LPF	$2^{-4.8}$	2^{-6}
DPF	$2^{-4.5}$	2^{-6}
SAC	0.5	0.4998
BIC	0.51	0.4998

Table 4: A comparison analysis of the substitution layer of Speck-R and AES.

5.5 Validation by the Sbox Evaluation Tool

Parameters tested	AES [37]	PRESENT [26]	Klein [51]	RC4-KSA
Input size M	8	4	8	8
Output size N	8	4	8	8
S-box Balanced	Balanced	Balanced	balanced	Balanced
Correlations immunity	0	3	0	0
Algebraic immunity	4	0	0	4
Transparency order	7.860	4	0.486	7.797
Robustness to differential cryptanalysis	0.984	0.984	0.004	0.953
SNR (DPA) (F)	9.600	0.250	0.133	8.73

Table 5: A comparison between SET execution samples: AES, PRESENT, KLEIN, RC4-KSA.

Robustness evaluation of *Sbox* is not limited to those four criteria. In fact, there exist different tools to evaluate the robustness of the cryptographic substitution tables. The evaluation of substitution tables has been quite a difficult issue for researchers since the public available tools are few. Some of these tools are: (1) **Boolfun package in R** that works under Unix and the package named boolfun can be loaded for functionality related to cryptography [52, 53, 54]. (2) **Boolean functions** in Sage [55] is another tool to evaluate the S-box. It is a free and open source mathematics software, that is mainly used to evaluate cryptographic properties of Boolean functions, mainly related to linear and differential properties. (3) The third tool is actually a **module for S-box in Sage**, which only has the possibility of calculating the difference distribution table and the linear approximation matrix, in terms of cryptographic properties. (4) **VBF (Vector Boolean Functions) library**, which is not available on-line to use freely, is presented by Alverez-Cubero and Zufiria to analyze vectorial Boolean functions from cryptographic perspective that possibly could calculate various properties of S-boxes [56]. Finally, (5) **SET** (Sbox Evaluation Tool), which is available freely online http://sidesproject.wordpress.com/, proposed in 2016 and takes almost all the necessary criteria to evaluate the substitution table.
In this paper, the security of the Sbox used was tested by self-generated code to test the previously explained four criteria (LPF, DPF, SAC and BIC). Then, the robustness of the non-linear layer was also

validated by using the newest tool available, SET-tool. In Table 5, a few examples for Sboxes are given of AES, PRESENT, KLEIN, and the proposed dynamic RC4-KSA. M and N represent the input and output variables of Sboxes. A subset of tests of the available properties is shown, which were executed on a machine with Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 16 GB RAM, and Linux Debian 9 (stretch).
The chosen subset of tests are listed below, where any curious reader can have the following references to dig deeper into these tests:

Balancedness: [57,58] A Boolean function is balanced if its output is equally distributed, its weight is equal to 2^{n-1}. This can be translated as $W_f(0) = 0$ for the Walsh spectrum. In the executed test, the four Sboxes are balanced and satisfy this property.

Correlation Immunity: [59,58] A function f is said to be correlation immune of order t, denoted by CI(t), if the output of the function is statistically independent of the combination of any t of its inputs. In the executed tests, the correlation immunity was 0 in AES, KLEIN and RC4-KSA, whereas for PRESENT, it was 3 which is not the desired value.

Algebraic immunity: [58,60] High nonlinearity is a necessary condition to resist algebraic attack and the value of algebraic immunity should not be low according to the study made in [61]. Boolean functions used in crypto-systems must have high non-linearity to prevent linear attacks [62]. In the results obtained, AES and RC4-KSA have the same algebraic immunity, 4, where as for KLEIN and PRESENT, it is 0.

Transparency Order (TO) : [63] Transparency order (TO) is the only one currently available to evaluate the inability of an S-Box to thwart the DPA attack. It has been proven that the smaller the TO of an S-Box, the higher its resistance would be against the DPA attacks. The value obtained for RC4-KSA (7.797) is close to the TO of AES (7.860), however, it seems that KLEIN and PRESENT have a better TO, 0.486 and 4, respectively.

Robustness to differential cryptanalysis: [47,64] An n-iput s-output Sbox is namely as S($n \times s$) where $s > \lfloor n/2 \rfloor$. It is at least $1 - 2^{-t}$, robust against differential cryptanalysis, where t is a parameter satisfying the condition that $[(s - \lfloor n/2 \rfloor) \geq t \geq 3]$. An Sbox attains its maximum robustness when $1-2^{-t}$ is minimum. In fact, to say that the following Sbox ($n \times s$) is robust against differential analysis, the result obtained must be close to the above boundaries of the following $(1 - 1/2^n)(1 - 2^{-s+1})$. For AES 8×8: the upper boundary is: $(1 - 1/2^8)(1 - 2^{-8+1}) = 0.988$, which is very close to the value obtained 0.984, then, AES is robust against differential analysis. For PRESENT, it is the same. While for KLEIN, the upper boundary is 0, the value obtained is close to 0. For RC4-KSA, the $Sbox$ used is 8×8, thus, the value obtained (0.953) is very close to the desired value (0.988).
DPA (Differential Power Analysis) Signal-to-noise ration SNR: In [65], authors showed that the DPA signal-to-noise ratio increases when the resistance of the substitution box against linear cryptanalysis increases. Mainly, secret key algorithms consist in the repetition of several rounds, and are thus threatened by the differential power analysis (DPA). The obtained SNR are 9.6 and 8.73 which are considered good to face the differential attacks, whereas the SNR for PRESENT and KLEIN are very low, 0.25 and 0.133 respectively.

The obtained results were sufficient to indicate that the proposed construction technique of key-dependent substitution produces a robust and efficient substitution table (Sbox). Furthermore, $Sbox_1$, $Sbox_2$ and $Sbox_3$ make the proposed cipher algorithm immune against differential and linear attacks, since they are changed in a pseudo-random manner.

6 Randomness Test Validation

Randomness plays a major role in proving whether a cryptographic algorithm is secure or not. However, some tools which are usually used to prove the randomness of pseudo-random generators are not used to prove the randomness of the output by the cipher. Therefore, we propose using TestU01 and Practrand

to validate the level of randomness desired in the ciphered output of any proposed cipher. In fact, using TestU01 and Practrand will save us time to run all the implemented tests within these tools. Based on the proposal in [66], a randomness test can be applied to show whether the cipher possess the required level of randomness or not by using both tools TestU01 [67] and Practrand [68]. To prove that Speck-R acts well under these tools and possess a high level of randomness in the ciphered image, both Speck and Speck-R are implemented and their codes are adapted to both of these tools. Using the "testingRNG" [69] project released by Daniel Lemire which basically aims at testing popular random-number generators, randomness tests were conducted. According to [66], the cipher should succeed for at least 4 Terabytes of data, changing one bit in the key or the nonce from one block to another only. If the test succeeds, then, we can say that this cipher possess the required level of randomness. $P - value$ is one of the tests that is done in both tools and which is basically one of the important statistical tests. First, the seed used is generated using the "Splitmix" [44] pseudo-random generator that generates 64 bits. Then, considering the worst case scenario which is setting the plaintext to zeroes, and the nonce is changed by incrementing it by 1 after each iteration. This can be summarized as fixing the main key, round keys, plaintext, except for the nonce which is incremented by 1 from one block to another. In this strategy, and if the randomness tests are passed, it can be said that this cipher possesses a high level of randomness even after changing only one bit in the nonce. The codes were implemented in C language, and for 16 TB of data for Practrand. The C codes for the Algorithms for both Speck and Speck-R are provided in the Algorithms 4 and 5.

Algorithm 4 Algorithm to test the randomness in Speck using TestU01 and Practrand.

```
procedure SPECK(newSeed)
    seed ← newSeed
    plaintext ← zeros
    nonce ← PseudoRandomGenerator(seed)
    EncKey ← PseudoRandomGenerator(seed)
    for i ← 0 to size(plaintext) do
        newnonce ← nonce + +
        speck6496(ciphertext, plaintext, newnonce, EncKey)
    end for
end procedure
```

Algorithm 5 Algorithm to test the randomness in Speck-R using TestU01 and Practrand.

```
procedure SPECK(newSeed)
    seed ← newSeed
    plaintext ← zeros
    DK ← PseudoRandomGenerator(seed)
    Sbox1, Sbox2, Sbox3 ← RC4 − KSA(DK, Sbox, 64)
    nonce ← PseudoRandomGenerator(seed)
    EncKey ← PseudoRandomGenerator(seed)
    for i ← 0 to size(plaintext) do
        newnonce ← nonce + +
        speck − R6496(ciphertext, plaintext, newnonce, EncKey)
    end for
end procedure
```

After running the C codes in Practrand and TestU01, both of the ciphers, Speck and Speck-R succeeded the tests and no failure has been noted. Therefore, these tools showed that the proposed cipher possesses a high level of randomness, with a lower number of rounds and a high security level. In the following section, excessive tests are exerted to prove that Speck-R is also useful to be used for ciphering images which hold many intrinsic properties.

7 Security Analysis

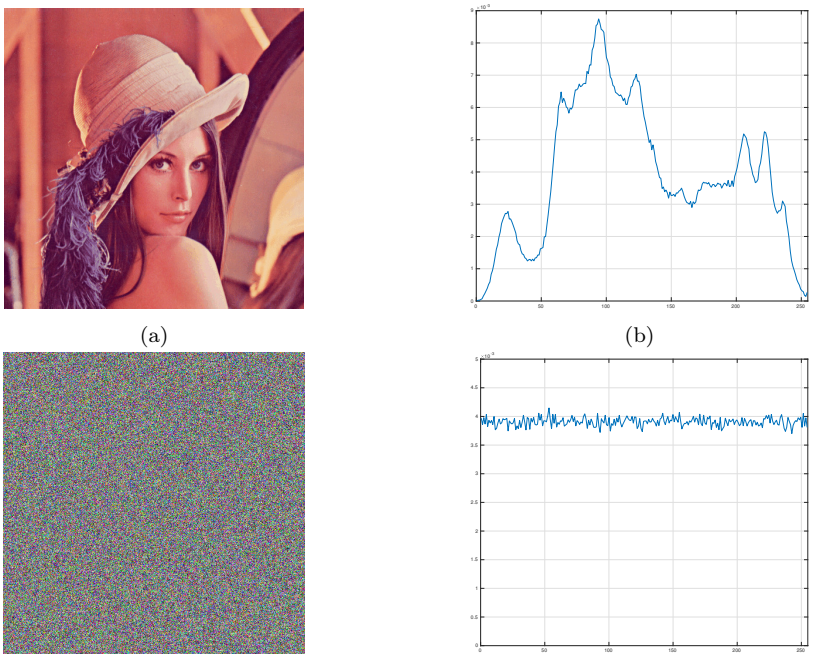
In this section, a security analysis is performed to validate the robustness of the proposed dynamic Speck-R. In fact, these tests show that Speck-R can also be used to encrypt images that possess different

intrinsic features and their data are highly correlated. These tests show its immunity against different confidentiality attacks such as statistical, differential, and brute force attacks [70]. To prove that Speck-R is efficient to encrypt images, several tests were conducted.

7.1 Statistical Analysis

A cipher scheme requires specific random properties in order to efficiently resist statistical attacks [71]. To prove the effectiveness of the proposed model, several statistical security tests were carried out to validate the uniformity and the independence properties. These tests are (1) **Uniformity Analysis**, (2) **Entropy test**, and the (3) **Correlation test.**

7.1.1 Uniformity Analysis



(a) (b) (c) (d)

Fig. 8: (a) Original Lenna, (b) PDF of original Lenna with size $512 \times 512 \times 3$, (c) Encrypted Lenna using Speck-R, (d) PDF of encrypted Lenna.

To show the Uniformity of the ciphered image, two parameters were used: (a) Probability Density Function (PDF) analysis, and the (b) Chi-square test. First, the encrypted image should possess certain random properties to resist the common statistical attacks. The most commonly used property is the PDF of the encrypted image that should be uniform. This requires each symbol to have a probability close to $\frac{1}{n}$, where n is the number of symbols. This value means that there is significantly no clue to employ any statistical attack. The PDF of the original plain-image and its corresponding cipher-image are both shown in Figure 8. It is clear that the PDF of the ciphered Speck-R image is close to 0.0039 ($\frac{1}{256}$). Additionally, in order to compute the level of uniformity of each encrypted image, the Chi-square test is applied according to equation 6:

$$\chi^2_{test} = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} \tag{6}$$

k represents the number of gray levels (here we work in grey scale images, then k=256), and o_i and e_i are the observed and expected occurrence frequencies of each gray level. This test aims at comparing

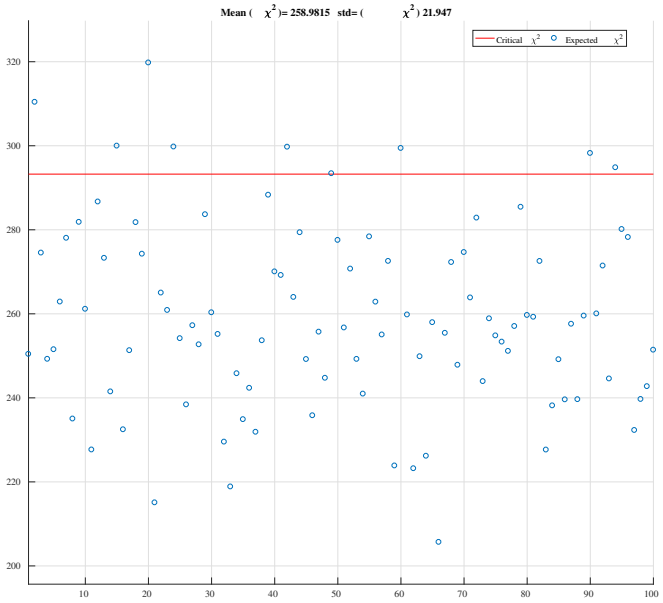


Fig. 9: The Chi-Square test for the encrypted Lenna image using 100 different dynamic keys.

the observed data with what we expect according to a specific hypothesis. Therefore, null hypothesis are formulated which are then rejected or retained with the help of statistical tests. The "significant level" is the probability value below which the null hypothesis is rejected, it can also be called the alpha level. According to [72], it is conventional to consider the null hypothesis false if the probability value is less than 0.05. In fact, having a significance level of 0.05 makes researchers 95% confident that the results represent a non-chance finding [73]. In addition, with a significance level of 0.05 and 256 number of intervals, the chi-square reaches a maximal value of 293 [74]. So, all values lower than this value are acceptable and indicate the uniformity distribution of the histogram. This criterion is verified, by testing the chi-square for the Lenna image under 100 different dynamic keys. It can be said that the redundancy of the plain image is hidden and does not provide any clue to apply statistical attacks. In Figure 9, it can be seen that mean of the chi-square value for 100 iterations mean chi-square value for 100 iterations of encrypted Lena image is approximately equal to $258.9815 \leq 293$, which confirms the uniformity property of the encrypted image under the proposed algorithm.

7.1.2 Entropy Test

The information entropy of an image, M, is a parameter that measures the level of uncertainty in a random variable [75], and it is defined using the following equation:

$$H(m) = -\sum_{i=1}^{n} p(m_i) \log_2 \frac{1}{p(m_i)} \tag{7}$$

$$H(m) = -\sum_{i=1}^{h^2} \frac{1}{h^2} \log_2 \frac{1}{h^2} = \log_2(h^2) \tag{8}$$

where $p(m_i)$ represents the occurrence probability of the symbol m_i and n is the total number of states of the information source. Note that the entropy is expressed in bits. The proposed entropy test measures the entropy at the sub-matrix level, where each sub-matrix has a size equal to h^2 bytes. This permits to quantify the uniformity at the sub-matrix level and not on the whole image. Each block can be considered as a truly random source with uniform distribution if it has an entropy equal or close to $\log_2(h^2)$. It is shown that the encrypted blocks always have an entropy close to the desired value 6 ($log_2(8 \times 8) = log_2(2^6) = 6$) in case $h = 8$. According to this, the proposed cipher ensures the uniformity and eliminates the redundancy between adjacent pixels.

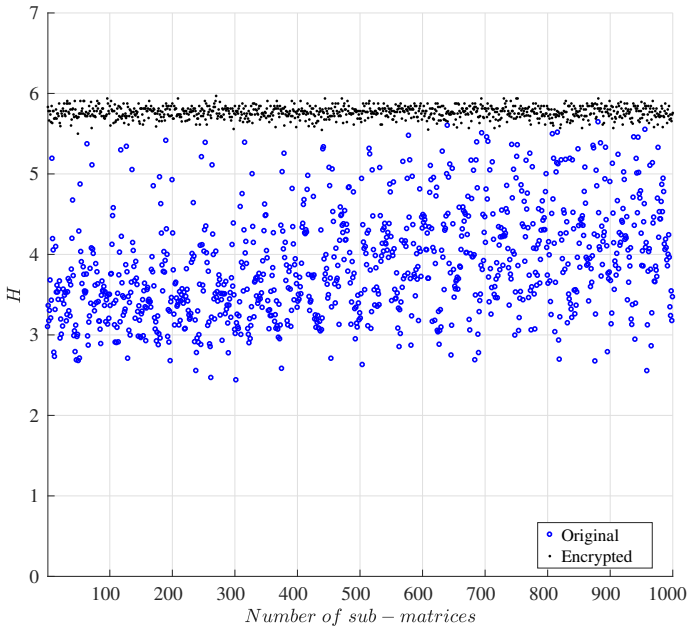


Fig. 10: The Entropy analysis for the sub-matrices of encrypted Lena image under the use of Speck-R with a random dynamic key for $h = 8$.

The Entropy analysis of original and encrypted Lena images under the use of a random dynamic key for $h = 8$ is shown in Fig. 10. The results indicate that the encrypted sub-matrices always have an entropy close to the desired value 6. This result proves that the proposed cipher ensures uniformity and eliminates any redundancy between adjacent sub-matrices.

7.1.3 Test correlation between original and cipher images

The high linear correlation among original image pixels must be removed to resist statistical attacks. Removing spatial redundancy will certainly result in an efficient cipher scheme [76,77]. Having a correlation coefficient close to zero means that the cipher scheme exhibits a high degree of randomness. The correlation test is performed by taking randomly $N = 4,066$ pairs of adjacent pixels from the known Lenna plain image and their corresponding cipher image. The correlation is done in horizontal, vertical and diagonal directions. The correlation coefficient r_{xy} is calculated using the following equations:

$$r_{xy} = \frac{\text{cov}(x, y)}{\sqrt{D(x) \times D(y)}} \tag{9}$$

where

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

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

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

Obviously, the correlation between adjacent pixels in the plain image is high and its corresponding correlation coefficient is close to 1. Whereas, the correlation in the ciphered imaged is close to 0. Fig. 11 shows the correlation between adjacent pixels in the different directions for a random secret key for the

original and ciphered Lenna image, which clearly shows that the proposed scheme drastically reduces the spatial redundancy.

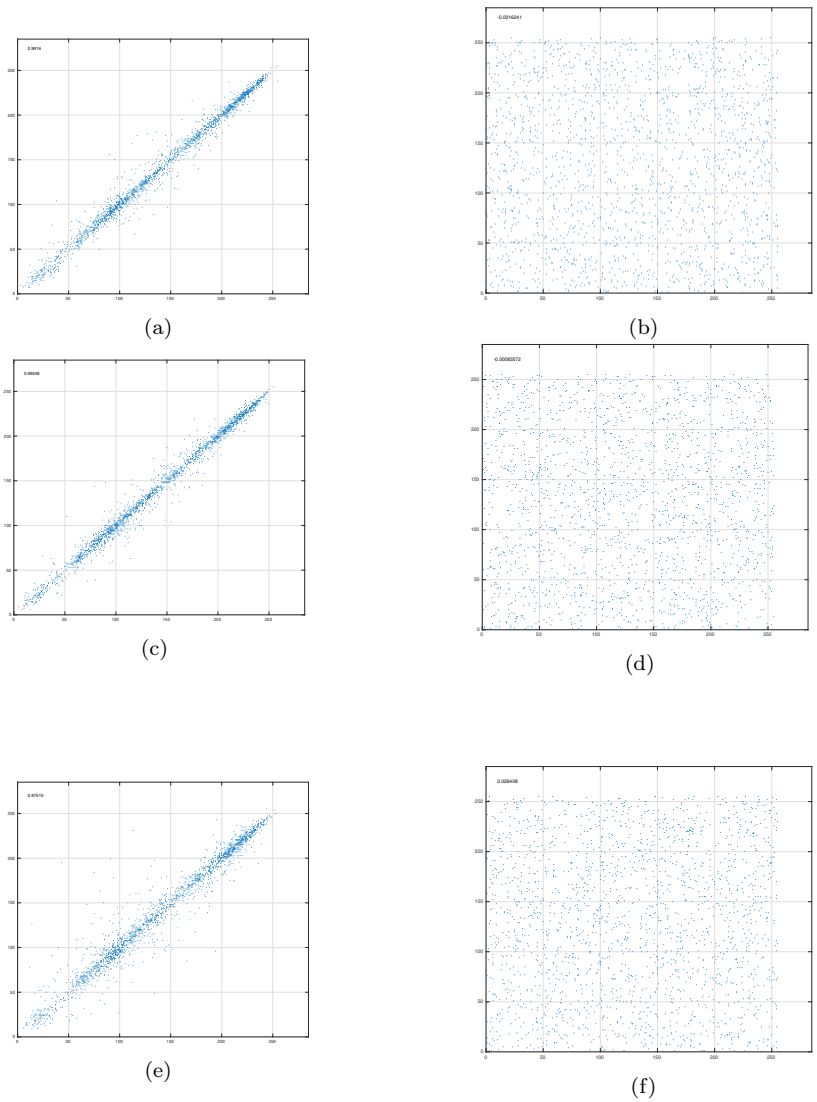


Fig. 11: Correlation in adjacent pixels in original Lenna: (a) horizontally, (c) vertically and (e) diagonally and the correlation in adjacent pixels in ciphered Lenna: (b) horizontally, (d) vertically and (f) diagonally.

Moreover, for 16 iterations, the mean of the correlation in its three directions was calculated and it is clearly shown that the scattering effect of Speck-R removes any spatial correlation in the ciphered image. In Figure 12, it is clear that the mean is always close to zero which validates our proposal and renders this cipher immune against statistical attacks.

7.2 Visual Degradation

The degradation of the original image must be verified, in a way that the visual content of the ciphered image is not recognized. For this aspect, two parameters are well known to measure the visual quality of encryption and these are the Peak Signal-to-Noise Ratio (PSNR) [78] and the Structural Similarity Index (SSIM) [79].

PSNR is derived from the Mean Squared Error (MSE), which represents the cumulative squared error between an original and encrypted image. A low PSNR value demonstrates that a high difference

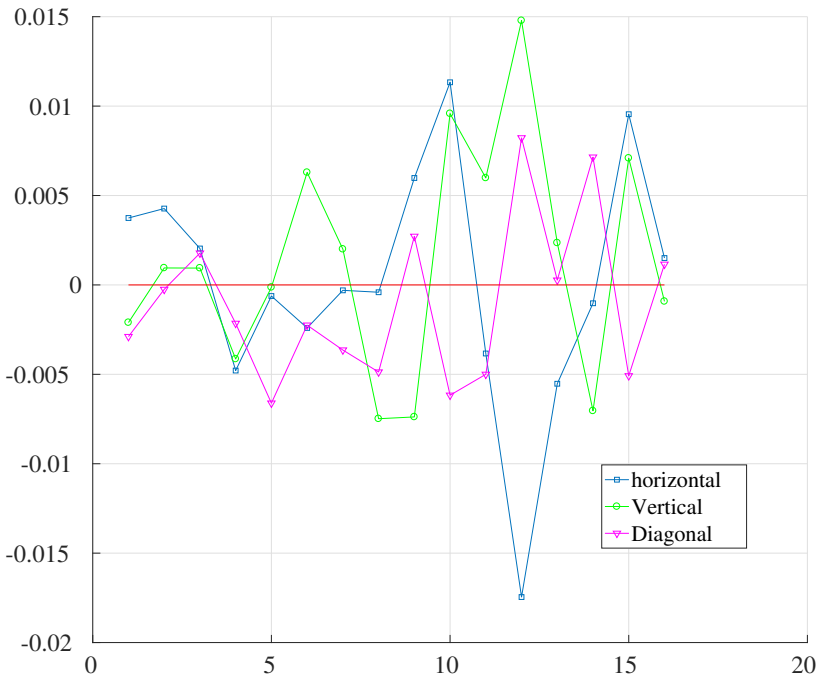


Fig. 12: The mean of the correlation of the encrypted Lenna after 16 iterations.

between the original and the cipher image exists.
SSIM index [80] is defined after the Human Visual System (HVS), which has evolved, so that we can extract the structural information from the scene. Thus, the perceived quality of the image by the human eye is highly dependent on the loss of structural information in the image. The SSIM value lies in the interval [0, 1]. A value of 0 means that there is no correlation between the original and the cipher image, while a value close to 1 means that both images are approximately the same.

In this context, PSNR and SSIM were measured between the original and the encrypted Lena image for $1,000$ dynamic keys and presented in Figure 13. As shown, the mean PSNR value is 8.62 dB. This low value confirms that the proposed encryption technique provides a high difference between the original and the encrypted images. Also, the SSIM value did not exceed 0.011, which means that a high and adequate visual distortion is achieved using the proposed encryption process.
As a conclusion, the proposed cipher scheme has a sufficient visual degradation where no useful information or any clear pattern about the original image is revealed from the encrypted image.

7.2.1 Difference Between plain And Cipher Image

Another criterion to measure the visual degradation is to measure the difference between original and encrypted images at the bit level. This value must reach a value very close to the ideal one (50%). In Figure 14, the difference between the original and cipher Lenna images for 1000 random dynamic keys is shown. The results show that the percentage difference is always close to 50%. Hence, the proposed cipher satisfies the independence criteria.

8 Performance Analysis

In this section, the performance of the proposed Speck-R is studied. In fact, the whole objective of this work is to increase the performance of the original Speck cipher, taking into account the high level of security. Looking forward to be implemented in IoT devices and small sensors, Speck-R undergoes

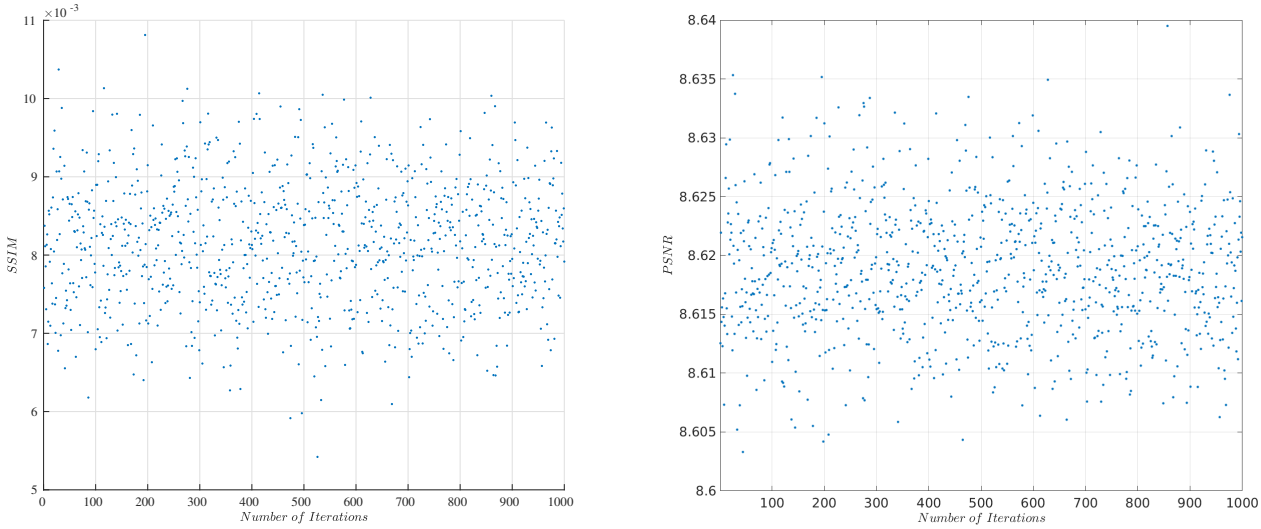


Fig. 13: PSNR and SSIM variation between the original and the encrypted Lena image versus $1,000$ dynamic keys.

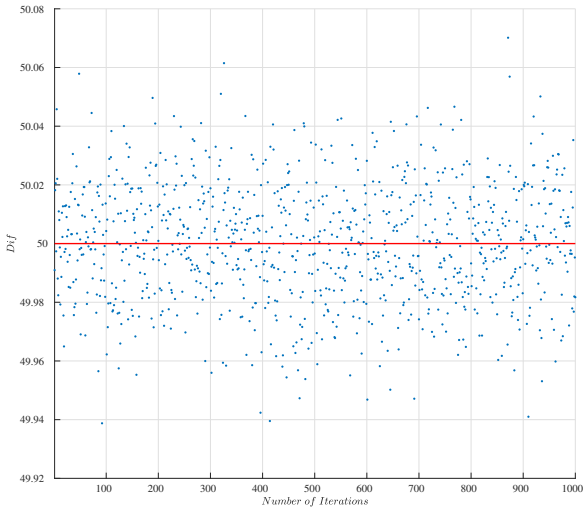


Fig. 14: Percentage difference between plain and ciphered Lena for 1000 random dynamic keys.

different tests to prove its high performance in limited constrained devices. Two different aspects are studied, (1) error propagation and (2) the execution time required to fulfill the encryption process.

8.1 Propagation of errors

In this proposal, the choice was made to work with Speck-R in CTR mode (counter mode), since it does not suffer from error propagation and the image will be resilient to different kinds of noises in the transmitting channel. This criterion should be as low as possible, which means that the error should not propagate to the whole transmitted image. Mainly, channel interferences and noises in transmission are the main causes of any errors. A bit error means toggling the '0' bit into '1' and vice versa. In the proposed cipher, if the block is affected it will only affect the bit in the exact position of the ciphered image. It will not propagate to the neighboring blocks, and this is the cost of not insuring the avalanche effect in the whole image. In Figure 16, we show the encrypted image of Lenna in Figure 15 after toggling

the Least Significant Byte (LSB) in half of the blocks in the ciphered image, then we decrypted it in Figure 17. The results show that the decrypted image is disturbed, but it is well recognizable. This shows that the error is not propagated in the image, which validates that the error is limited to its own block.

Fig. 15: Lenna $512 \times 512 \times 3$

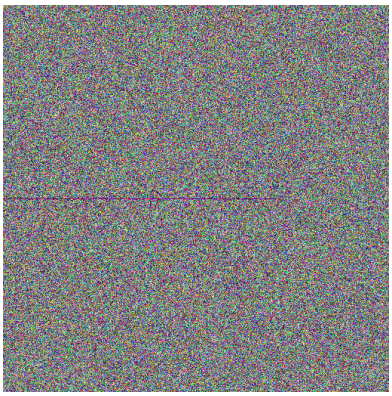


Fig. 16: Encrypted Lenna after toggling.

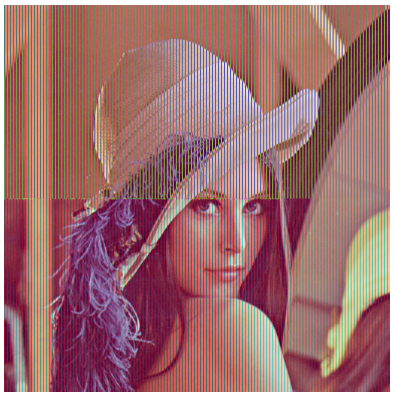


Fig. 17: Dycrypted toggled Lenna.

8.2 Execution time

To validate that Speck-R is efficient for small limited devices, we tested the execution time on three different IoT devices: ATmega323p, Teensy 3.6 and DOIT ESP32. Below, Table 6 lists some of the specifications of these three IoT chips. Figures 18, 19 and 20 represent the three microchips used.

Device	ATmega328P	Teensy 3.6	DOIT ESP32
Flash Memory Size (KB)	32	256	4,096
Operating Voltage Range (V)	1.8 to 5.5	3.3V	3.3
Clock Speed (Mhz)	16	180	80
Processor	8 bit	32 bit	32 bit

Table 6: Specifications of ATmega323P,Teensy 3.6, and DOIT ESP32

Fig. 18: ATmega328P

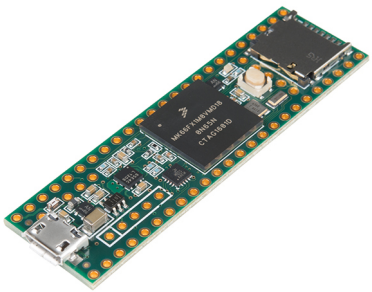


Fig. 19: TEENSY 3.6

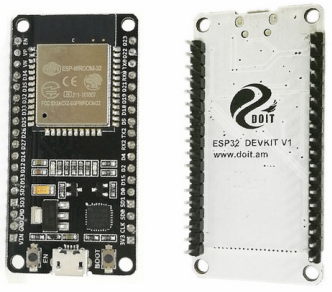


Fig. 20: DOIT ESP32

Both algorithm, Speck and Speck-R were implemented on these three IoT microchips. The results are shown in Figures 21, 22 and 23. In fact, Figure 21 represents the time result in μsec of Speck and

Speck-R when implemented on ATmega328P. It seems that Speck-R performs better on this limited 8-bit micro controller chip. The encryption process is done for 16, 32, 64, 128, 256 and 512 bytes. For 16 bytes of data, the execution time for Speck is 940 μsec, while for Speck-R it is 304 μsec, which means that it took Speck-R less than half the time for Speck to encrypt 16 bytes. Then for 512 bytes, the execution times for Speck and Speck-R are 29668 μsec and 9840 μsec, respectively. It is clear that the time required for the encryption increases proportionally to the size of data. According to equation 10, the percentage of enhancement in the execution time is 66.8% when implementing both algorithms on ATmega328P.

$$Enhancement_{executiontime}\% = 100 \times (1 - \frac{Time_{Speck-R}}{Time_{Speck}}) \tag{10}$$

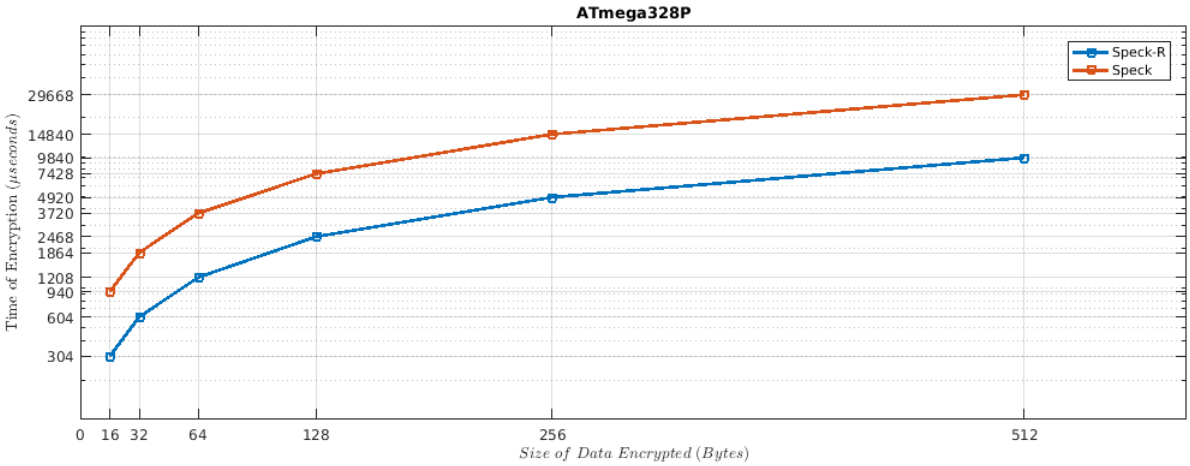


Fig. 21: Execution time (μsec) versus the size of data (bytes) encrypted of Speck and Speck-R when implemented on ATmega328p.

Then, both algorithms are implemented on the next IoT device, Teensy 3.6. An enhancement is recorded in the case of Speck-R. When encrypting 16 bytes, both execution times were the same 2μsec, but as the data get larger, reaching 65536 bytes, the execution time of Speck-R is 5728 μsec, while for Speck 7015 μsec. The percentage of the enhancement is 18.34% when using Teensy 3.6.

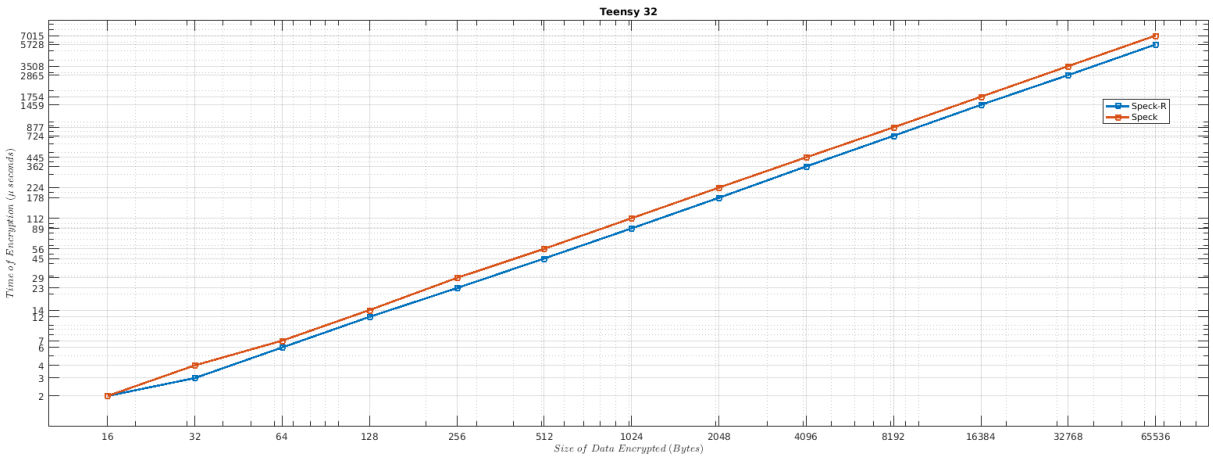


Fig. 22: Execution time (μsec) versus the size of data (bytes) encrypted of Speck and Speck-R when implemented on Teensy 3.6.

The last microchip used is the DOIT ESP32. This chip which includes WiFi and Bluetooth, is widely used by researchers. Starting by 16 bytes of data to encrypt, Speck took 9 μsec while Speck-R took 2 μsec. Then, reaching the maximum number of bytes, 4096 bytes, Speck-R with 328 μsec also possessed a higher performance comparing it to Speck having 597 μsec. In fact, the enhancement starts from 77% to reach a static 45% as the number of bytes increases.

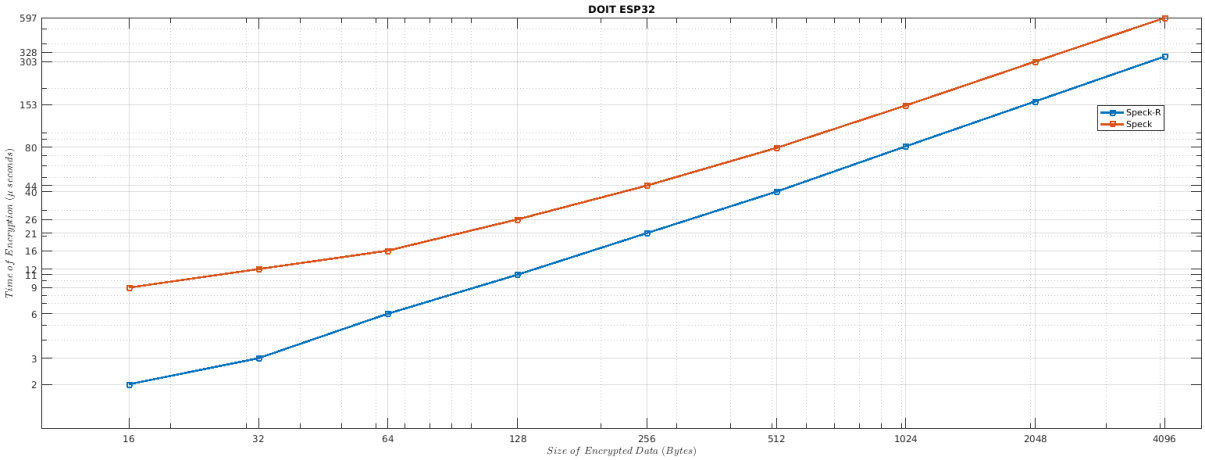


Fig. 23: Execution time (μsec) versus the size of data (bytes) encrypted of Speck and Speck-R when implemented on DOIT ESP32.

As a conclusion, not only does Speck-R possess the security level necessary, but it also has a better execution time on these IoT chips which are limited in memory and in computational ability. Having simple operations such as Xor, shift, rotation and a simple Sbox, nominates this cipher to be a good proposal for today's security challenges.

9 Discussion and Crypt-analysis

"Fully secure systems do not exist today and they will not exist in the future." — AdiShamir.
After all, researchers aim at enhancing as much as they can the level of security and strengthening the ciphers against well-known and new attacks. But, how can we know that the cipher is resilient against different kinds of attacks? First, the two main properties in this cipher are preserved namely are confusion and diffusion. Confusion is preserved by using the proposed dynamic Sboxes and the diffusion is reached by using Xor, and shifting by α and β parameters.
Different statistical tests were performed and they proved that the proposed cipher satisfies the uniformity and independence properties. Hence, a high randomness level is achieved in a dynamic manner, which makes the proposed cipher immune against statistical attacks.
Using a dynamic key for every image proves that the cipher exhibits a high level of immunity against key-related attacks. Especially as the cryptographic parameters change as the dynamic key changes. Even if a cryptanalyst has a complete knowledge of the used primitives for a plain image, he/she will fail to extract information about the future plain images from the future cipher images, since they lack the dynamic key that is changed for every input image.
Also, the proposed cipher is immune to brute force attack since Speck itself can be used for different keys, starting from 64 reaching 256 bits. We chose to work on 96 bits, but this proposal can work on any other Speck version.
To sum up, dynamicity of the proposal and using different layers of Sbox adds more randomness and makes the proposed cipher scheme immune against the current and future powerful attacks such as chosen/known plain/cipher text attacks. In conclusion, the security level of the proposed cipher scheme is confirmed.

10 Conclusion

In this paper, a new Speck version was proposed which is called Speck-R. Speck-R comes to serve mostly the limited devices which are characterized by limited abilities and restricted power. Speck has been one of the most successful proposals in the domain of lightweight cryptography. In Speck-R, a dynamic confusion layer of substitution based on a dynamic key is added. The *Sboxes* are built using a dynamic key and then changed according to the number of iterations. Adding dynamicity to the proposal adds immunity against different powerful attacks. In fact this is the difference between the static versions of the previous proposals for a reduced Speck and the proposed Speck-R. We chose the CTR-mode, 64 bits block, and 96 bits key Speck version. The main contribution of this work is to reduce the number of rounds of Speck from 26 to 7 while maintaining a high level of security. Decreasing the number of rounds will result in a decrease in execution times. Extensive tests validated the proposal and supported it. Moreover, it can be noted that at least 18.34% to 77% enhancement compared to Speck in terms of execution times on three different limited chips is obtained. As for the future work, we look forward to enhancing the Simon cipher in terms of execution times. Adding dynamicity to Simon must be tested and proved to be efficient as well. In this track, we will propose two enhanced ciphers targeting software and hardware, based on Simon and Speck together.

Acknowledgement

Part of the simulations was conducted on the servers of the "Mésocentre de calcul de Franche-Comté". We would like to thank them for accepting our request and for giving us access to their machines. This paper is also partially supported from the EIPHI Graduate School (contract "ANR-17-EURE-0002").

References

1. Pouya Kamalinejad, Chinmaya Mahapatra, Zhengguo Sheng, Shahriar Mirabbasi, Victor CM Leung, and Yong Liang Guan. Wireless energy harvesting for the internet of things. *IEEE Communications Magazine*, 53(6):102–108, 2015.
2. Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park. Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18, 2017.
3. DG INFSO et al. Internet of things in 2020: Roadmap for the future. *INFSO D*, 4, 2008.
4. Alex S Weddell and Michele Magno. Energy harvesting for smart city applications. In *2018 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*, pages 111–117. IEEE, 2018.
5. Zhaogang Shu, Jiafu Wan, Di Li, Jiaxiang Lin, Athanasios V Vasilakos, and Muhammad Imran. Security in software-defined networking: Threats and countermeasures. *Mobile Networks and Applications*, 21(5):764–776, 2016.
6. Ioannis Andrea, Chrysostomos Chrysostomou, and George Hadjichristofi. Internet of things: Security vulnerabilities and challenges. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 180–187. IEEE, 2015.
7. NSA. Lightweight cryptography, 2019. [Online; 2019].
8. Alex Biryukov and Léo Paul Perrin. State of the art in lightweight symmetric cryptography. 2017.
9. Bassam J Mohd, Thaier Hayajneh, and Athanasios V Vasilakos. A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues. *Journal of Network and Computer Applications*, 58:73–93, 2015.
10. Joan Daemen and Vincent Rijmen. *The design of Rijndael*, volume 2. Springer, 2002.
11. Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of aes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 69–88. Springer, 2011.
12. Dag Arne Osvik, Joppe W Bos, Deian Stefan, and David Canright. Fast software aes encryption. In *International Workshop on Fast Software Encryption*, pages 75–93. Springer, 2010.
13. Benjamin Buhrow, Paul Riemer, Mike Shea, Barry Gilbert, and Erik Daniel. Block cipher speed and energy efficiency records on the msp430: System design trade-offs for 16-bit embedded applications. In *International Conference on Cryptology and Information Security in Latin America*, pages 104–123. Springer, 2014.
14. Stefan Tillich and Johann Großschädl. Instruction set extensions for efficient aes implementation on 32-bit processors. In *International workshop on cryptographic hardware and embedded systems*, pages 270–284. Springer, 2006.
15. Shay Gueron. Intel's new aes instructions for enhanced performance and security. In *International Workshop on Fast Software Encryption*, pages 51–66. Springer, 2009.
16. Adam J Elbirt. Fast and efficient implementation of aes via instruction set extensions. In *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, volume 1, pages 396–403. IEEE, 2007.
17. M Taufik, DE Amin, and MA Saifuddin. Hardware implementation and optimization of advanced encryption standard (aes) algorithm based on ccsds. In *AIP Conference Proceedings*, volume 2226, page 060004. AIP Publishing LLC, 2020.
18. Kerry McKay, Larry Bassham, Meltem Sönmez Turan, and Nicky Mouha. Report on lightweight cryptography (nistir8114). *National Institute of Standards and Technology (NIST)*, 2017.

19. Hyubgun Lee, Kyounghwa Lee, and Yongtae Shin. AES implementation and performance evaluation on 8-bit microcontrollers. *CoRR*, abs/0911.0482, 2009.
20. 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.
21. R Nithya and Deepa S Kumar. Where aes is for internet, simon could be for iot. *Procedia Technology*, 25:302–309, 2016.
22. David J Wheeler and Roger M Needham. Tea, a tiny encryption algorithm. In *International Workshop on Fast Software Encryption*, pages 363–366. Springer, 1994.
23. Roger M Needham and David J Wheeler. Tea extensions. *Report, Cambridge University*, 1997.
24. Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, et al. Hight: A new block cipher suitable for low-resource device. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 46–59. Springer, 2006.
25. Manoj Kumar, Saibal K Pal, and Anupama Panigrahi. Few: A lightweight block cipher. *IACR Cryptol. ePrint Arch.*, 2014:326, 2014.
26. Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. Present: An ultra-lightweight block cipher. In *International workshop on cryptographic hardware and embedded systems*, pages 450–466. Springer, 2007.
27. Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences*, 58(12):1–15, 2015.
28. Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Dong-Geon Lee. Lea: A 128-bit block cipher for fast encryption on common processors. In *International Workshop on Information Security Applications*, pages 3–27. Springer, 2013.
29. Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, et al. Prince–a low-latency block cipher for pervasive computing applications. In *International conference on the theory and application of cryptology and information security*, pages 208–225. Springer, 2012.
30. Ronald L Rivest. The rc5 encryption algorithm. In *International Workshop on Fast Software Encryption*, pages 86–96. Springer, 1994.
31. George Hatzivasilis, Konstantinos Fysarakis, Ioannis Papaefstathiou, and Charalampos Manifavas. A review of lightweight block ciphers. *Journal of Cryptographic Engineering*, 8(2):141–184, 2018.
32. Tomer Ashur and Daniël Bodden. Linear cryptanalysis of reduced-round speck. In *Proceedings of the 37th Symposium on Information Theory in the Benelux*, 2016.
33. Itai Dinur. Improved differential cryptanalysis of round-reduced speck. In *International Conference on Selected Areas in Cryptography*, pages 147–164. Springer, 2014.
34. Harshal Tupsamudre, Shikha Bisht, and Debdeep Mukhopadhyay. Differential fault analysis on the families of simon and speck ciphers. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 40–48. IEEE, 2014.
35. Ashutosh Dhar Dwivedi, Pawel Morawiecki, and Gautam Srivastava. Differential cryptanalysis of round-reduced speck suitable for internet of things devices. *IEEE Access*, 7:16476–16486, 2019.
36. Daniel Engels, Markku-Juhani O Saarinen, Peter Schweitzer, and Eric M Smith. The hummingbird-2 lightweight authenticated encryption algorithm. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pages 19–31. Springer, 2011.
37. Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.
38. Zeinab Fawaz, Hassan Noura, and Ahmed Mostefaoui. An efficient and secure cipher scheme for images confidentiality preservation. *Signal Processing: Image Communication*, 42:90–108, 2016.
39. Hassan Noura, Lama Sleem, Mohamad Noura, Mohammad M Mansour, Ali Chehab, and Raphaël Couturier. A new efficient lightweight and secure image cipher scheme. *Multimedia Tools and Applications*, 77(12):15457–15484, 2018.
40. Salim Muhsin Wadi and Nasharuddin Zainal. High definition image encryption algorithm based on aes modification. *Wireless personal communications*, 79(2):811–829, 2014.
41. Xingyuan Wang, Lin Teng, and Xue Qin. A novel colour image encryption algorithm based on chaos. *Signal Processing*, 92(4):1101–1108, 2012.
42. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. Notes on the design and analysis of simon and speck. *IACR Cryptology ePrint Archive*, 2017:560, 2017.
43. R Beaulieu, D Shors, J Smith, S Treatman-Clark, B Weeks, and L Wingers. The simon and speck families of lightweight block ciphers cryptology eprint archive, 2013.
44. Guy L Steele Jr, Doug Lea, and Christine H Flood. Fast splittable pseudorandom number generators. In *ACM SIGPLAN Notices*, volume 49, pages 453–472. ACM, 2014.
45. Craig G Eisler and G Eric Engstrom. Method and system for managing color specification using attachable palettes and palettes that refer to other palettes, December 28 1999. US Patent 6,008,816.
46. Ronald L Rivest. The rc4 encryption algorithm. rsa data security. *Inc., March*, 12:9–2, 1992.
47. Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72, 1991.
48. Mitsuru Matsui. Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer, 1993.
49. A. F. Webster and Stafford E. Tavares. On the design of s-boxes. In *Advances in Cryptology*, CRYPTO '85, pages 523–534, Berlin, Heidelberg, 1986. Springer-Verlag.
50. Carlisle Adams and Stafford Tavares. The structured design of cryptographically good s-boxes. *Journal of Cryptology*, 3(1):27–41, Jan 1990.
51. Zheng Gong, Svetla Nikova, and Yee Wei Law. Klein: a new family of lightweight block ciphers. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pages 1–18. Springer, 2011.

52. Frédéric Lafitte. The boolfun package: Cryptographic properties of boolean functions. 2012.
53. Fatih Özkaynak. An analysis and generation toolbox for chaotic substitution boxes: a case study based on chaotic labyrinth rene thomas system. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, pages 1–10, 2019.
54. R Core Team et al. R: A language and environment for statistical computing. 2013.
55. William Stein et al. Sage: Open source mathematical software. *7 December 2009*, 2008.
56. José Antonio Alvarez-Cubero and Pedro J Zufiria. A c++ class for analysing vector boolean functions from a cryptographic perspective. In *2010 International Conference on Security and Cryptography (SECRYPT)*, pages 1–9. IEEE, 2010.
57. Henri Gilbert and Helena Handschuh. *Fast Software Encryption (12 conf.)*. Springer, 2005.
58. Ann Braeken. *Cryptographic properties of Boolean functions and S-boxes*. PhD thesis, phd thesis-2006, 2006.
59. Claude Carlet. On highly nonlinear s-boxes and their inability to thwart dpa attacks. In *International Conference on Cryptology in India*, pages 49–62. Springer, 2005.
60. Yves Crama and Peter L Hammer. *Boolean functions: Theory, algorithms, and applications*. Cambridge University Press, 2011.
61. Deepak Kumar Dalai, Kishan Chand Gupta, and Subhamoy Maitra. Results on algebraic immunity for cryptographically significant boolean functions. In Anne Canteaut and Kapaleeswaran Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004*, pages 92–106, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
62. Cunsheng Ding, Guozhen Xiao, and Weijuan Shan. *The stability theory of stream ciphers*, volume 561. Springer Science & Business Media, 1991.
63. Limin Fan, Yongbin Zhou, and Dengguo Feng. A fast implementation of computing the transparency order of s-boxes. In *2008 The 9th International Conference for Young Computer Scientists*, pages 206–211. IEEE, 2008.
64. Jennifer Seberry, Xian-Mo Zhang, and Yuliang Zheng. Systematic generation of cryptographically robust s-boxes. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 171–182. ACM, 1993.
65. Sylvain Guilley, Philippe Hoogvorst, and Renaud Pacalet. Differential power analysis model and some results. In *Smart Card Research and Advanced Applications Vi*, pages 127–142. Springer, 2004.
66. Lama Sleem and Raphaël Couturier. Testu01 and practrand: Tools for a randomness evaluation for famous multimedia ciphers. *Multimedia Tools and Applications*, pages 1–14, 2020.
67. Pierre L'Ecuyer and Richard Simard. Testu01: Ac library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):22, 2007.
68. C Doty-Humphrey. Practrand. *URL: https://goo. gl/HwU9g5*, 2014.
69. Daniel Lemire. testingRNG, 2018. [Online; 2018].
70. Jung-Sik Cho, Sang-Soo Yeo, and Sung Kwon Kim. Securing against brute-force attack: A hash-based rfid mutual authentication protocol using a secret value. *Computer communications*, 34(3):391–397, 2011.
71. Shujiang Xu, Yinglong Wang, Jizhi Wang, and Min Tian. Cryptanalysis of two chaotic image encryption schemes based on permutation and xor operations. In *2008 International Conference on Computational Intelligence and Security*, volume 2, pages 433–437. IEEE, 2008.
72. Jean-Baptist du Prel, Gerhard Hommel, Bernd Röhrig, and Maria Blettner. Confidence interval or p-value?: part 4 of a series on evaluation of scientific publications. *Deutsches Ärzteblatt International*, 106(19):335, 2009.
73. CR Wilson VanVoorhis and Betsy L Morgan. Understanding power and rules of thumb for determining sample sizes. *Tutorials in quantitative methods for psychology*, 3(2):43–50, 2007.
74. Jun-xin Chen, Zhi-liang Zhu, Chong Fu, Li-bo Zhang, and Yushu Zhang. An efficient image encryption scheme using lookup table-based confusion and diffusion. *Nonlinear Dynamics*, 81(3):1151–1166, 2015.
75. Guoji Zhang and Qing Liu. A novel image encryption method based on total shuffling scheme. *Optics Communications*, 284(12):2775–2780, 2011.
76. Rhouma Rhouma and Safya Belghith. Cryptanalysis of a new image encryption algorithm based on hyper-chaos. *Physics Letters A*, 372(38):5973–5978, 2008.
77. Benyamin Norouzi, Seyed Mohammad Seyedzadeh, Sattar Mirzakuchaki, and Mohammad Reza Mosavi. A novel image encryption based on hash function with only two-round diffusion process. *Multimedia systems*, 20(1):45–64, 2014.
78. Quan Huynh-Thu and Mohammed Ghanbari. Scope of validity of PSNR in image/video quality assessment. *Electronics letters*, 44(13):800–801, 2008.
79. Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004.
80. Shujun Li and Xuan Zheng. Cryptanalysis of a chaotic image encryption method. In *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, volume 2, pages II–708. IEEE, 2002.