

Keyed S-boxes from sponge functions

Arthur van der Merwe *

David Paul †

Jelena Schmalz §

Timothy M. Schaerf ‡

School of Science and Technology

University of New England

Armidale

NSW 2351

Australia

Abstract

This paper examines the generation of key-dependent S-boxes using sponge functions. In this study, we render 8-bit key-dependent S-boxes employing a novel approach that contrasts previous techniques using a mixing strategy. We test the efficiency and security of the resultant S-boxes by performing a sequence of experiments. We consider the integration of the keyed S-boxes into a symmetric cipher and determine the probability of differential cryptanalysis. We found that the maximum differential distribution is approximately $12/256$ or $\approx 2^{-6}$, with a mean of roughly $11/256$ and a standard deviation of $1.2/256$. Our review of other methods concludes that they have the same maximum differential distribution, while our generation method is significantly faster. We conclude by characterising future research on keyed S-boxes in a white-box context.

Subject Classification: (2010) 94A60, 20C05.

Keywords: *Sponge, Symmetric, Encryption, S-box, Dynamic, White-box encryption.*

This research is supported by an Australian Government Research Training Program (RTP) Scholarship.

* ORCID-ID: 0000-0001-9146-5433

† ORCID-ID: 0000-0002-2428-5667

§ ORCID-ID: 0000-0001-5486-7679

‡ ORCID-ID: 0000-0001-6642-8374

* E-mail: avande22@myune.edu.au (Corresponding Author)

† E-mail: dpaul4@une.edu.au

§ E-mail: jschmalz@une.edu.au

‡ E-mail: tschaerf@une.edu.au

1. Introduction

The design of block ciphers has been an elusive art for decades. The initial design of DES [19] and the introduction of cipher structures based on bitstreams and binary combinations [18] were drastically different from the Enigma machine [20] and Caesar ciphers' lexical notions. Horst Feistel's work [18] was a turning point in block cipher design, shifting away from linear shift ciphers and incorporating non-linear layers and structural diffusion. Since the creation of DES [19], advanced ciphers like AES [27], MISTY [25], and others have embodied linear and non-linear layers to circumvent attacks that were successful on their predecessors, employing the long-tail approach [14]. Recent developments in white-box cryptography [8, 9, 12], which embeds secret keys within application code by combining techniques for encryption and obfuscation, similarly transformed the strategy for symmetric ciphers. Cipher designers looked towards mechanisms to conceal secret encryption keys while the cipher is wholly accessible to an adversary [15]. The initial interest of our research focused on the design of white-box ciphers, consequently discovering a novel technique to combine secret keys with standard S-boxes.

2. Contribution

During this study, our analysis encountered simplistic ways to incorporate encryption keys into the structure of a symmetric cipher. To our surprise, our experiments showed that our keyed S-box generation method has a low probability of statistical attacks while being efficient. Previous approaches to incorporating keys into S-boxes were complex and had computational restraints, which we discuss in Section 5. Our methodology of rendering keyed S-box layers focuses on 128-bit block ciphers with 8-bit input and 8-bit output S-boxes that cipher designers can extend or generalise to various block sizes. However, smaller S-boxes appear to have adverse properties. Realistically, we intended to create a mechanism to generate unique S-boxes efficiently, such that a white-box cipher can easily rotate static keys. Current white-box ciphers incorporate table-based methodologies and lack key rotation capability [8]. Randomised encryption schemes are also well-desired [28], especially in the presence of adaptive adversaries. With this reasoning, we determined that a simplistic and efficient method for dynamic S-box generation is a well-desired property in future white-box ciphers. In later research, we

aim to extend the generation method by designing a white-box cipher incorporating the S-box generation method.

3. The structure of the paper

We commence in Section 4 by reviewing the notation and definitions utilised in this paper, then discussing key-dependent S-box generation in Section 5. We detail the SPONGE construction in Section 5.1 and then define our methodology to create key-dependent S-boxes from SPONGES in Section 5.1, followed by analysis in Section 5.3. We examine smaller S-boxes in Section 5.4 and recommend ‘Golden S-boxes’, finishing by detailing improvements and future work in Section 6.

4. Notation and definitions

Throughout this paper, we denote elements in hexadecimal notation. For example, 0x04 is equivalent to the decimal four and the bit string 100. Additionally, if two operations are functionally equivalent, $a \triangleq b$, then the output of $a(x)$ is equal to $b(x)$ for the same value x . We define the truncation of values using $\lfloor a \rfloor_x$ to reflect the rightmost x bits of a . Similarly, $\lfloor a \rfloor_x$ represents the leftmost x bits of a . Lastly, we denote a random key k sampled from a uniformly random key space $\mathcal{K}$ as $k \xleftarrow{\$} \mathcal{K}$ and specify the $\oplus$ operation as binary addition modulo two.

Given a pair of distinct binary strings of length n , $v \leftarrow \{0,1\}^n$ and $u \leftarrow \{0,1\}^n$, $u \oplus v$ is the value of the difference. When discussing difference propagation given a function f , we refer to the difference between two values $(v \oplus u)$ as the input difference and the output difference as $f(v) \oplus f(u)$. A differential function f consists of an input difference a and an output difference b and is denoted by $f(a, b)$. The differential probability (DP) of the differential (a, b) is given by the number of pairs that have input difference a and output difference b , divided by the total number of pairs with input difference a :

Definition 1: Given an input difference a and an output difference b , for $u, v \in \{0,1\}^n$ the differential probability of a function f is defined as:

$$DP(a, b) = \#\{(v, u) \mid u \oplus v = a \text{ and } f(v) \oplus f(u) = b\} / 2^{n-1}$$

The differential probability (DP) is either zero or a multiple of $2^{-(n-1)}$. There are 2^{n-1} non-zero input differences for any given n . There are 2^{n-1} different pairs for every non-zero input difference. The highest differential

probability value for all inputs is the *maximum differential probability*. Where the differential probability is zero, it is named the *impossible differential*.

Negligible Functions

A negligible function decreases faster than $1/p(n)$ for any polynomial p . Real-world paradigms function on polynomial-time algorithms; events that transpire with a negligible probability look to polynomial-time algorithms like they never materialise.

Definition 2: A function $\mu : \mathbb{N} \rightarrow \mathbb{R}$ is negligible for every polynomial p , if there exists an n_0 such that for all $n > n_0$, $\mu(n) \leq 1/p(n)$.

Random permutations

A random permutation is a random arrangement of the elements of a set whose values have an equiprobable likelihood of occurring.

$\mathcal{O}(n)$ Notation describes the upper bound of the complexity of executing an algorithm.

Definition 3: Let $T(n)$ and $f(n)$ be two positive functions. We write $T(n) \in \mathcal{O}(f(n))$, and say that $T(n)$ has order of $f(n)$, if there are positive constants M and n_0 such that $T(n) \leq M \cdot f(n)$ for all $n > n_0$.

1. An algorithm with $T(n) \in \mathcal{O}(1)$ is said to have constant time complexity.
2. An algorithm with $T(n) \in \mathcal{O}(n)$ is said to have linear time complexity.
3. An algorithm with $T(n) \in \mathcal{O}(n^2)$ has quadratic time complexity.

5. Keyed dependent S-box generation

Incorporating a cipher's key with the non-linear layer is a widely deployed strategy for white-box ciphers, e.g., in *SPNBox* [8]. Mixing S-boxes has been examined since 1991 [26], where designers used several intricate techniques to obtain low differential probabilities while maintaining a non-linear structure. The work of [21] proposed generating a permutation and mixing the columns and rows of the AES S-box. Incorporating the columns and rows might sound simplistic but demonstrated statistical attack resistance [21]. Several S-box generation techniques use chaotic tent maps, which have been studied for approximately ten years [10, 22, 23]. We revisit the approaches proposed

from 1991 to the current day and then present a novel method for generating key-dependent S-boxes using a keyed hash function with the Richard Durstenfeld algorithm [16].

History of key-dependent S-boxes

One of the early ciphers designed with key-dependent S-boxes was Khufu, designed by Merkle et al. in 1991 [26], a 64-bit block cipher with 16 rounds using a 512-bit key. Khufu combined half of the 512-bit key with an 8×13 -bit S-box, then connected the output of the S-box with the other 256-bit half. Khufu was predominantly used to encrypt a large quantity of data. In 1993 the Blowfish algorithm [30] was developed by Schneier et al., followed by the Twofish block cipher [31]. The Twofish cipher used a pre-computed key-dependent S-box with an elaborate key schedule. The Twofish cipher used half of the n -bit key for key-dependent S-boxes, and the other half XOR'ed with the state.

In 2008, Zhang et al. developed an efficient Feistel DSDP cipher [11] with a key-dependent S-box and key-dependent P-boxes to resist shortcut attacks such as linear and differential cryptanalysis. Wang et al., in 2009, produced dynamic S-boxes with iterating chaotic tent maps [34], based on the work of [22]. Wang et al. improved the cipher's security by alternating the state value of the tent map. In 2010 Zaibi et al. [35] proposed constructing dynamic S-boxes using chaotic maps. They showed that combining one and three-dimensional piecewise linear maps has the lowest linear approximation probability and an equiprobable input/output distribution. In 2011 Szaban et al. proposed using dynamic cellular automata (CA)-based S-boxes [32] proving high non-linearity and balance, low auto-correlation and distance to fulfil the strict avalanche criterion. In 2012, Hosseinkhani et al. used a novel method to generate dynamic S-boxes in two steps [21], starting with the AES S-box and then swapping the columns and rows. Mahmoud et al. [24] 2013 used an equivalent methodology, using a Linear Feedback Shift Register (LFSR) to generate random sequences and reorder the columns and rows of the AES S-box based on the LFSR output. Alabaichi et al. 2014 then enhanced the Blowfish cipher by designing a new function (F-Function) into a Cylindrical Coordinate System (CCS), producing a dynamic 3D S-box [3]. Zhang et al., in 2014, used image encryption to generate dynamic S-boxes [36]. Al-Wattar et al. [2] and Liu et al. [23] proposed a dynamic encryption method based on DNA. They introduced dynamic S-boxes and chaotic systems to develop a new image encryption scheme. Belazi et al. in 2017 generated

dynamic S-boxes based on a permutationâ€‘substitution using chaotic systems [5]. Similarly, Ahmad et al. in 2017 also proposed a block cipher based on chaotic maps [1].

Although the approaches and advancements discussed above are significant discoveries, they cannot rapidly generate keyed S-boxes based on minor variations in the multiple inputs and reusing the random keys. Using a mechanism similar to hash functions enables an implementation to vary the hash function inputs rather than interchanging the key.

5.1 Key dependent S-boxes from hash functions

We generate key-dependent S-boxes using an arbitrary key k and message m as input to a duplex SPONGE [6]. We then use the sponge to squeeze a randomised permutation as input to the Durstenfeld algorithm [16] to shuffle the standard AES S-box.

Background on duplex SPONGE structures

The full-state keyed duplex sponge generalises the duplex defined by Bertoni et al.[6]. The full-state keyed duplex uses a public permutation, represented as a function, $p(\cdot)$ from b bits to b bits, $p: \{0,1\}^b \rightarrow \{0,1\}^b$, and a random key $k \leftarrow \mathcal{K}$. The internal state S has a width b , such that $b \leftarrow |S|$ and is broken up into a capacity c and rate r with the conditions $b = r + c$, $r < b$, and $|k| \leq c$.

A single instance of a full-state keyed duplex has two interfaces, namely `D.initialise` and `D.duplexing`. In the `D.initialise` interface, a random key k is set, with the condition that $|k| \leq c$ and the remainder of S is padded with zeroes such that $S = S || 0^{b-|k|}$, denoted as $\text{pad}_b(m)$. The `D.duplexing` procedure is parameterised with a message $m \in \{0,1\}^{b-|k|}$ such that $|m| < b$ and the number of output bits is z . Each `D.duplexing` call XORs the input message m and applies the public permutation $p(\cdot)$ to the state S . `D.duplexing` then outputs a string $\text{left}_z(b)$ which is a state S truncated to the leftmost z -bits. The full-state keyed duplex sponge repeats the `D.duplexing` interface i times until the desired number of output bits are received. In Algorithm 1, we describe the two procedures that make up the full-state keyed duplex sponge as implemented in our key-dependent S-box generation algorithm.

Algorithm 1 Full-state keyed duplex sponge $[k, m, S, p(\cdot)]$

```

procedure D.INITIALISE( $k$ )                                ▷ initialise the procedure with a random key  $k$ 
   $S \leftarrow k || 0^{b-|k|}$                                     ▷ output  $b$ -bits
end procedure

procedure D.DUPLEXING( $m, S$ )                                ▷ Initialise the procedure with a message  $m$  and state  $S$ .
  if  $b \geq r$  or  $|m| \geq b$  then
    return  $\perp$ 
  else
     $S \leftarrow S \oplus \text{pad}_b(m)$                                 ▷ XOR the padded message.
     $S \leftarrow p(b)$                                             ▷ Apply the permutation to  $S$ .
  end if
  return  $\text{left}_z(S)$                                             ▷ Truncate  $z$  resultant bytes as the output.
end procedure

```

Key dependent S-boxes from SPONGES

We define a strategy by discussing the general idea of generating key-dependent S-boxes from SPONGES. We follow up the strategy by performing experiments and assessing the potential security aspects of using SPONGES to generate key-dependent S-boxes.

In the D.initialise procedure

The procedure is initialised with a uniform random key from a key space $\mathcal{K}$ as $k \xleftarrow{\$} \mathcal{K}$ such that $|k| = b$ and setting the sponge capacity c . The rate r is then padded with zeros $0^{b-|k|}$ satisfying $b = r + c$, $r < b$, and $k \leq c$, given $S = S || 0^{b-|k|}$.

D.duplexing procedure

Key observations when using the direct output of the D.duplexing procedure for the generation of an S-box are the following:

1. Given an input m , and $|m| < b$, the maximum output will be $\frac{|m|}{2}$ -bits, which may be truncated.
2. Building 256 8-bit entries for S-box values (2048-bits) will require at least 32 iterations of the D.duplexing procedure.
3. The D.duplexing output will contain collisions concerning previous results since we are attempting to generate a unique set of 256 values from 256 possibilities.
4. The output of D.duplexing may be represented as 32-bit signed integers, where a sorting algorithm can reorder S-box values.

Because of the significant collisions and duplications in S-box values, using the output directly is not practicable. However, the output of

D.duplexing may be used to reorder the entries of a well-known S-box, like the AES [27] or Skipjack [7] S-box. Using a key directly to reorder an S-box would trivially disclose the key. The **D.duplexing** procedure is unique in compressing the key bits, hiding the original key in a one-way manner in the output permutation. Determining the actual key bits amounts to inverting the SPONGE function. Additionally, the input message m may contain environment-specific information like a counter and device identifier, enabling the generation of a unique S-box for each encryption cycle or transaction. We defer the input message m to the implementer but stress that m must be unique for every generated S-box.

5.2 Keyed S-box generation

The standard AES S-box operates on an 8-bit input and 8-bit output, having 2^8 (256) distinct S-box values. Similarly, other S-boxes may operate on 4 bits and have 2^4 (16) values. We start the keyed S-box generation of 256 values, operating on 8-bit input and 8-bit output termed $\mathcal{S}_\sigma$.

To generate $\mathcal{S}_\sigma$ we initialise the sponge with a 64-bit random key and sample 64-bit output from **D.duplexing** over 128 rounds, producing 8192-bits. We use **D.duplexing**($m, 64$) with the message space m of the S-box input pairs, obtaining 128 outputs $q = \{c_1, c_2, \dots, c_{127}, c_{128}\}$. The outputs $c_1 \dots c_{128}$ are concatenated and represented as 256 signed integers.

We then use the Richard Durstenfeld algorithm [16] with the SPONGE output to shuffle the standard AES S-box in a key-dependent manner. The Durstenfeld algorithm reduces the shuffling algorithm's time complexity to $\mathcal{O}(n)$ compared to $\mathcal{O}(n^2)$ for the Fisher and Yates implementation [17]. We take each c_i in $q = \{c_1, c_2, \dots, c_{255}, c_{256}\}$ as a set of integers, executing Algorithm 2, to create a key-dependent S-box SB' from an empty S-box, iterating SB once. We show the high-level process in Figure 5.1.

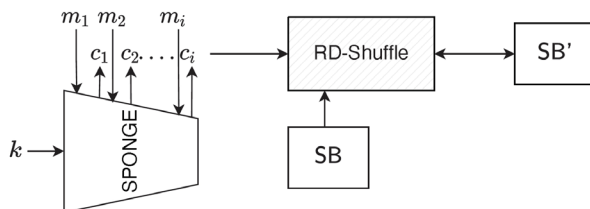


Figure 5.1

Keyed S-box generation where SB denotes the standard AES S-box and SB' is the generated S-box.

Algorithm 2

```

procedure KEYED S-BOX( $q, SB$ )
  for  $i$  from  $c_1$  to  $c_i \in q$  do
     $z = c_1 || \dots || c_i$ 
  end for
  RD-Shuffle( $z, SB$ )
end procedure

procedure RD-SHUFFLE( $z, SB$ )
  Set  $SB' = 0$ 
  Set  $SB$ 
  for  $i$  from  $|SB'|$  to 0 do
     $j \leftarrow z[i]$ 
    Set  $j = j \bmod |SB|$ 
    Set  $SB'[i] = SB[j]$ 
    Remove  $SB[j]$ 
  end for
  return  $SB'$ 
end procedure
```

▷ Concatenate the SPONGE output values.

▷ The Richard Durstenfeld algorithm.

▷ Set an empty S-box.

▷ Set AES S-box.

▷ Iterate through empty S-box setting values.

▷ Select a random element, indexed by z .

▷ Reduce j to be in the range $|SB|$.

▷ Set SB' S-box value indexed by z .

▷ Delete the element and reduce the size of SB .

5.3 Analysis of the keyed S-boxes

During this study, we implemented the keyed S-box construction to estimate the effectiveness of utilising a randomised S-box. In the case of an S-box with 256 entries, we implemented the algorithm in 2, acquiring the outcomes in Table 5.1. We sampled a series of 10 to 10000 S-boxes. Our experiments showed that the prevalent maximum differential probability was 12, with a mean of 11 and a standard deviation of 1.2. Considering the average security case, a single input-output difference has an average maximum differential probability of $12 / 256$ or $\approx 2^{-6}$. The time to generate a single S-box was approximately 0.014 seconds, with an insignificant deviation. Experiments were conducted on a MacBook Pro using macOS Monterey version 12.3.1 with 16 GB 3733 MHz RAM, a 2.3 GHz Quad-Core Intel Core i7 and an Intel Iris Plus Graphics 1536 MB GPU.

Table 5.1
Summary of our 8-bit Keyed S-box metrics.

Samples	Generation Time		Maximum differential probability			
	Mean seconds	Standard Deviation seconds	Mean	Standard Deviation	Distinct Values	Mode
10	0.0154	0.0011	10	2.449	10, 4, 12	12
100	0.0139	0.0010	11.373	1.529	10, 4, 12, 14	12
1000	0.0129	0.0006	11.339	1.274	4, 10, 12, 14, 16, 18	12
10000	0.0135	0.0010	11.338	1.217	4, 10, 12, 14, 16, 18	12

Experiments indicate that the generation of keyed S-boxes using our method is secure on average and efficient.

Our method is as secure as [35], [33], [4], [22] and [10], which all have a maximum differential probability of 12. The work of [24] generates dynamic S-boxes taking 2.963207 seconds to process 500 data blocks. Our method, in contrast, takes 0.0154 seconds for an entire S-box. However, it is unclear what hardware the designers of [24] used to obtain their metrics.

Potential differential probability

Our study considered combining the S-box generation method with a potential 128-bit block cipher with a fully activated state for a single input difference and an average differential probability of 2^{-6} . Assuming a fully active state of a 128-bit cipher, the differential probability could be $2^{-6 \times 16}$ or 2^{-96} . If extended over 4 rounds, the maximum differential probability could reach $2^{-96 \times 4} = 2^{-384}$. The potential maximum differential probability depends on the cipher design, such as the mixing and diffusion layers.

5.4 Analysis of smaller S-boxes

Markku-Juhani et.al. conducted a study [29] on all possible 4×4 -bit S-box bijective combinations and produced properties for optimal S-box linear equivalent classes. In the study, an exhaustive search through all possible $16!$ values of a 4×4 -bit S-box revealed that 54.72% had a maximum differential probability of $3/8$. Additionally, 27% of the S-boxes have a maximum differential probability of $1/2$. Generating 4×4 -bit S-boxes in a key-dependent way has a significant probability of obtaining a maximum differential probability close to $1/2$. The Markku-Juhani et al. study [29] also identified a series of ‘Golden S-boxes’. These ‘Golden S-boxes’ and their inverses exhibit ideal properties with a maximum differential probability of $1/4$. All of the output bits of the S-box have an algebraic degree of 3 and are dependent on all input bits in a linear fashion. Instead of generating smaller S-boxes, e.g., 4×4 – bits in a key-dependent way, it is more feasible to use the ‘Golden S-boxes’. It is, therefore, reasonable to use ‘Golden S-boxes’ in Table 5.2 for small S-boxes, as opposed to our unique method that relies on the random output of a SPONGE construction.

Table 5.2
‘Golden S-boxes’ defined by Markku-Juhani et.al. [29]

035869C7DAE41FB2
03586CB79EADF214
03586AF4ED9217CB
03586CB7A49EF12D

6. Conclusion and future work

Our study demonstrates a unique mechanism to generate key-dependent S-boxes utilising the *SPONGE* construction. We experimented with demonstrating that the generation time is efficient and offers the core properties that prevent differential cryptanalysis by providing maximum resistance. To our knowledge, this paper details the first keyed S-box generation using a hash function. White-box secure ciphers like *SPNBox* [13] use round-reducing ciphers and keyed tables in their non-linear layers. Additional work is required to determine if our S-box generation method might be considered white-box secure and an alternative for a keyed table. Adaptive adversaries are problematic for white-box systems due to the static nature of the embedded keys. Due to the simplistic nature of the generation method in this paper, the keyed S-boxes could be generated for each encryption cycle where the input message can use environmental state information. Generating dynamic S-boxes will resolve the ongoing issue of defeating adaptive adversaries in white-box cipher designs. Future work on our S-box generation method includes designing a block cipher and demonstrating the cipher’s security in a white-box context, replacing keyed tables and proving resistance against adaptive adversaries.

References

- [1] M. Ahmad and A. Chopra, “Chaotic dynamic S-boxes based substitution approach for digital images,” arXiv preprint arXiv:1709.07620 (2017).
- [2] A. S. Al-Wattar, R. Mahmod, Z. A. Zukarnain, and N. I. Udzir, “Generating a new S-box inspired by biological DNA,” *International Journal of Computer Science and Application*, vol. 4, no. 1, pp. 32–42, 2015.

- [3] A. ALabaichi, R. Mahmood, and F. Ahmad, "Randomness analysis of 128 bits Blowfish block cipher on ECB mode," *International Journal of Computer Science and Information Security*, vol. 11, no. 10, p. 8 (2013).
- [4] M. Asim and V. Jeoti, "Efficient and simple method for designing chaotic S-boxes," *ETRI Journal*, vol. 30, no. 1, pp. 170–172 (2008).
- [5] A. Belazi and A. A. Abd El-Latif, "A simple yet efficient S-box method based on chaotic sine map," *Optik*, vol. 130, pp. 1438–1444 (2017).
- [6] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Duplexing the sponge: single-pass authenticated encryption and other applications," in *International Workshop on Selected Areas in Cryptography*, pp. 320–337, Springer (2011).
- [7] E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials," *Journal of Cryptology*, vol. 18, no. 4, pp. 291–311 (2005).
- [8] A. Bogdanov and T. Isobe, "White-box cryptography revisited: Space-hard ciphers," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1058–1069 (2015).
- [9] J. Bringer, H. Chabanne, E. Dottax, et al., "White-box cryptography: Another attempt," *IACR Cryptology ePrint Archive*, vol. 2006, p. 468 (2006).
- [10] G. Chen, Y. Chen, and X. Liao, "An extended method for obtaining S-boxes based on three-dimensional chaotic baker maps," *Chaos, Solitons & Fractals*, vol. 31, no. 3, pp. 571–579 (2007).
- [11] L. Chen and R. Zhang, "A key-dependent cipher DSdp," in *2008 International Symposium on Electronic Commerce and Security*, pp. 310–313, IEEE (2008).
- [12] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "White-box cryptography and an AES implementation," in *International Workshop on Selected Areas in Cryptography*, pp. 250–270, Springer (2002).
- [13] F. Cioschi, N. Fornari, and A. Visconti, "White-box cryptography: A time-security trade-off for the SPNbox family," in *International Conference on Wireless Intelligent and Distributed Environment for Communication*, pp. 153–166, Springer (2018).
- [14] J. Daemen and V. Rijmen, *The Design of Rijndael*, vol. 2, Springer (2002).

- [15] C. Delerablée, T. Lepoint, P. Paillier, and M. Rivain, "White-box security notions for symmetric encryption schemes," in *International Conference on Selected Areas in Cryptography*, pp. 247–264, Springer (2013).
- [16] R. Durstenfeld, "Algorithm 235: random permutation," *Communications of the ACM*, vol. 7, no. 7, p. 420 (1964).
- [17] M. Eberl, "Fisher-Yates shuffle," *Arch. Formal Proofs* (2016).
- [18] H. Feistel, W. A. Notz, and J. L. Smith, "Some cryptographic techniques for machine-to-machine data communications," *Proceedings of the IEEE*, vol. 63, no. 11, pp. 1545–1554 (1975).
- [19] P. FIPS, "Data encryption standard—federal information processing standards publication 46," U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service (1977).
- [20] D. H. Hamer, G. Sullivan, and F. Weierud, "Enigma variations: An extended family of machines," in *Proceedings-A*, vol. 140, Citeseer (1993).
- [21] R. Hosseinkhani and H. H. S. Javadi, "Using cipher key to generate dynamic S-box in AES cipher system," *International Journal of Computer Science and Security (IJCSS)*, vol. 6, no. 1, pp. 19–28 (2012).
- [22] G. Jakimoski and L. Kocarev, "Chaos and cryptography: block encryption ciphers based on chaotic maps," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 2, pp. 163–169 (2001).
- [23] H. Liu, B. Zhao, and L. Huang, "Quantum image encryption scheme using Arnold transform and S-box scrambling," *Entropy*, vol. 21, no. 4, p. 343 (2019).
- [24] E. M. Mahmoud, A. Abd El Hafez, and T. A. El Hafez, "Dynamic AES-128 with key-dependent S-box," (2013).
- [25] M. Matsui, "New block encryption algorithm Misty," in *International Workshop on Fast Software Encryption*, pp. 54–68, Springer (1997).
- [26] R. C. Merkle, "Fast software encryption functions," in *Conference on the Theory and Application of Cryptography*, pp. 477–501, Springer (1990).
- [27] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications*, National Institute of Standards and Technology, pp. 19–22 (2001).
- [28] R. L. Rivest and A. T. Sherman, "Randomized encryption techniques," in *Advances in Cryptology*, pp. 145–163, Springer (1983).

- [29] M. J. O. Saarinen, "Cryptographic analysis of all 4×4 -bit S-boxes," in International Workshop on Selected Areas in Cryptography, pp. 118–133, Springer (2011).
- [30] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," in International Workshop on Fast Software Encryption, pp. 191–204, Springer (1993).
- [31] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, The Twofish encryption algorithm: a 128-bit block cipher, John Wiley & Sons, Inc. (1999).
- [32] M. Szaban and F. Serebinski, "Cryptographically strong S-boxes based on cellular automata," in International Conference on Cellular Automata, pp. 478–485, Springer (2008).
- [33] G. Tang, X. Liao, and Y. Chen, "A novel method for designing S-boxes based on chaotic maps," Chaos, Solitons & Fractals, vol. 23, no. 2, pp. 413–419 (2005).
- [34] Y. Wang, K. W. Wong, X. Liao, and T. Xiang, "A block cipher with dynamic S-boxes based on tent map," Communications in Nonlinear Science and Numerical Simulation, vol. 14, no. 7, pp. 3089–3099 (2009).
- [35] G. Zaibi, F. Peyrard, A. Kachouri, D. Fournier-Prunaret, and M. Samet, "A new design of dynamic S-box based on two chaotic maps," in ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2010), pp. 1–6, IEEE (2010).
- [36] X. Zhang, Z. Zhao, and J. Wang, "Chaotic image encryption based on circular substitution box and key stream buffer," Signal Processing: Image Communication, vol. 29, no. 8, pp. 902–913 (2014).

Received August, 2022

Revised November, 2022