A Novel Method to Design S-Boxes Based on Key-Dependent Permutation Schemes and its Quality Analysis

Kazys Kazlauskas
Institute of Mathematics and Informatics
Vilnius University
Vilnius, Lithuania

Robertas Smaliukas
Institute of Mathematics and Informatics
Vilnius University
Vilnius, Lithuania

Gytis Vaicekauskas
Institute of Mathematics and Informatics
Vilnius University
Vilnius, Lithuania

***Abstract*—S-boxes are used in block ciphers as the important nonlinear components. The nonlinearity provides important protection against linear and differential cryptanalysis. The S-boxes used in encryption process could be chosen to be key-dependent. In this paper, we have presented four simple algorithms for generation key-dependent S-boxes. For quality analysis of the key-dependent S-boxes, we have proposed eight distance metrics. We have assumed the *Matlab* function "*randperm*" as standard of permutation and compared it with permutation possibilities of the proposed algorithms. In the second section we describe four algorithms, which generate key-dependent S-boxes. In the third section we analyze eight normalized distance metrics which we have used for evaluation of the quality of the key-dependent generation algorithms. Afterwards, we experimentally investigate the quality of the generated key-dependent S-boxes. Comparison results show that the key-dependent S-boxes have good quality and may be applied in cipher systems.**

Keywords—data encryption; substitution boxes; generation algorithms; distance metrics; quality analysis

I. Introduction

Cryptographic objects are private key algorithms, public key algorithms and pseudorandom generators. Block ciphers usually transform the 128 or 256 bits string of the plaintext to a string of the same length cipher text under control of the secret key. The private key cryptography, such as DES [1], 3DES, and Advanced Encryption Standard (AES) [2], uses the same key for the sender and for the receiver to encrypt the plaintext and to decrypt the ciphertext. The private key cryptography is more suitable for the encryption of a large amount of data. The public key cryptography, such as the Rivest-Shamir-Adleman algorithm defined by the National Institute of Standards and Technology of the Unite (RSA) or Elliptic Curve algorithms, uses different keys for encryption and decryption. The AES has been accepted to replace DES. AES overpasses DES in an improved security because of larger key sizes. AES is suitable for 8 bit microprocessor platforms and 32 bit processors [3].

The essential part of every block cipher is an S-box. To secure the cipher against attacks, the nonlinearity of the S-box should have a maximum nonlinearity, and the difference propagation probability should be minimum. Substitution is a nonlinear transformation that performs confusion of bits. A nonlinear transformation is important for every encryption algorithm and it is proved to be a strong cryptographic method against the linear and differential cryptanalysis. Nonlinear transformations are implemented as lookup tables (S-boxes). An S-box with p input bits and q output bits is denoted as $p \to q$. The DES uses eight $6 \to 4$ S-boxes. S-boxes are designed for software implementation on 8-bit processors. The block ciphers with $8 \to 8$ S-boxes are SAFER, SHARK, and AES. For processors with 32-bit or 64-bit words, S-boxes with more output bits provide a high efficiency. The Snefru, Blowfish, CAST, and SQUARE use $8 \to 32$ S-boxes. The S-boxes can be selected at random as it is in Snefru, and can be computed using a chaotic map, or have some mathematical structure over a finite Galois field. Examples of the last approach are SAFER, SHARK, and AES. Key-dependent S-boxes are slower, but more secure than the key independent S-boxes [4]. The use of the key independent chaotic S-boxes are analyzed in [5], in which the S-box is constructed with a transformation $F((X+K) \bmod M)$, where K is the key [6].

There are two ways to fight against the linear and differential cryptanalysis. The first one is to create S-boxes with low linear and differential probabilities. The other is to design the round transformation so that only trails with many active S-boxes would occur. The round transformation must be designed in such a way that differential steps with few active S-boxes would be followed by differential steps with many active S-boxes [6].

Two general principles of block ciphers are confusion and diffusion. Confusion is transformation that changes the dependence of the statistics of the cipher text on the statistics of the plaintext. Diffusion is spreading of the influence of one plaintext bit to many cipher text bits with the purpose to hide the statistical structure of the plaintext. In most cipher systems the confusion and diffusion are achieved by means of round repetition. Repeating a single round contributes to the cipher's simplicity [6]. Modern block ciphers consist of four transformations: substitution, permutation, mixing, and key-adding [7], [8].

Block cipher systems depend on the S-boxes, which are fixed and have no relation with the secret key. So only a changeable parameter is the secret key. The only nonlinear component of AES is S-boxes, so they are an important source of cryptographic strength. Research of the S-box design has focused on determination of S-box properties which yield cryptographically strong ciphers, with the aim of selecting a small number of good S-boxes for use in a block cipher DES and CAST [8]. Some results have demonstrated that a randomly chosen S-box of sufficient size will have several of these desirable properties with a high probability [9]. In [10] a dynamic AES-128 with a key-dependent S-box is designed and implemented. The paper of [11] presents a new AES-like design for key-dependent AES using the S-box rotation method. An approach for designing a key-dependent S-box defined over $GF(2^4)$ in AES is presented in [12]. A key-dependent S-box of AES algorithm using a variable mapping technique is analyzed in [13]. A key-dependent S-box generation algorithm in AES block cipher system is proposed in the paper [14]. Hamdy *et al.* [15] have proposed a customized version of the AES block cipher in which the key-dependent S-box generation algorithm is used. In the paper Hosseinkhany *et al*. the dynamic S-box is generated in the AES cipher system using the secret key [16]. Other systems, using key-dependent S-boxes were proposed in the past, the most well-known of which is Blowfish [7] and Khufu [17]. Each of these two systems uses the cryptosystem itself to generate the S-boxes. In [19] for generation S-boxes an algorithm based on chaotic map and composition method is used. In [20] a method for the construction of block ciphers with multi-chaotic systems is proposed. In the paper of D. Lambic the security analysis and improvement of a block cipher with dynamic S-boxes based on tent map is analyzed [21]. In the paper of Ozkaynak *et al*., is done analysis of a novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system [22].

This paper outlines the work of the authors' investigation into the design of a new pseudo-randomly generated key-dependent S-boxes. We have presented four simple algorithms for generation key-dependent S-boxes. For quality analysis of the key-dependent S-boxes, we have proposed to use eight distance metrics. Modeling results show, that the proposed algorithms have a good cryptographic strength, with an additional benefit that the algorithms are resistant to the linear and differential cryptanalysis, which require that the S-boxes be known. In the second section, we analyze four algorithms for generation of key-dependent S-boxes. In the third section we propose eight distance metrics for evaluation of the quality of key-dependent S-boxes. Afterwards, we discuss the experimental results and give conclusions.

II. Algorithms for Generation Key-Dependent S-Boxes

In this section we analyze four algorithms for generation of key-dependent S-boxes *Sboxm*. These algorithms use some key-dependent permutations of the elements of the initial substitution box *Sbox* to get key-dependent substitution box *Sboxm*. Algorithm 1 was proposed in the paper [18].

The initial substitution box *Sbox* may be the AES substitution box (table) or the ordered numbers 0,1,…,255, or these numbers mixed in any order. In all these cases the sender and the receiver must know these initial S-boxes. We assume that the S-boxes are rearranged according to the rows to the 256-size vectors, i.e., the initial substitution box *Sbox*(*i*), *i* = 0,1,…,255 and the key-dependent substitution box *Sboxm*(*i*), *i* = 0,1,…,255 are 256-size vectors of the different integer numbers (bytes) from the interval [0, 255]. The indexes *i* of these vectors are also the integer numbers (bytes) from the interval [0, 255]. In the encryption process, the indexes *i* of the vector *Sbox* (or *Sboxm*) are replaced by the corresponding values *Sbox*(*i*), (or *Sboxm*(*i*)). In the decryption process, the values of the vector *Sbox*(*i*), (or *Sboxm*(*i*)) are replaced by the corresponding indexes of the vector *Sbox* (or *Sboxm*).

A. Algorithm 1 (A1)

Input:

The secret key *key*(*i*), *i* = 1,…, *l* is the vector of *l* integer numbers (bytes) from the interval [0, 255].

The initial substitution box *Sbox*(*i*), *i* = 0,1,…,255 is the vector of different integer numbers (bytes) from the interval [0,255].

Output:

The key-dependent substitution box *Sboxm*(*i*), *i* = 0,1,…,255 is the vector of the integer numbers (bytes) from the interval [0, 255].

The key-dependent inverse substitution box *invSboxm*(*i*), *i* = 0,1,…,255 is the vector of different integer numbers (bytes) from the interval [0, 255].

1. Compute the initial value *j*, which depends on all the secret key's values *key*(*i*), *i* = 1,2,…,*l* from the interval [0, 255]:

$$j \leftarrow \sum_{i=1}^{l} key(i) \bmod 256$$

$$\textit{for all } i = 0,1,\ldots,255 \textit{ do}$$

2. Compute the index *j* which depends on the values of the initial substitution box *Sbox* and on the values of the secret key *key*:

$$k \leftarrow (Sbox(i) + Sbox(j)) \bmod l$$
$$j \leftarrow (j + key(k)) \bmod 256$$

3. Replace the values *Sbox*(*i*) by the values *Sbox*(*j*), and the values *Sbox*(*j*) by the values *Sbox*(*i*):

$$p \leftarrow Sbox(i)$$
$$Sbox(i) \leftarrow Sbox(j)$$
$$Sbox(j) \leftarrow p$$
$$\textit{end for}$$

4. Write the key-dependent substitution box values to *Sboxm*:

$$Sboxm \leftarrow Sbox$$

5. Compute the key-dependent inverse substitution box values *invSboxm*:

```
for all $i = 0,1,...,255$ do
$invSboxm\,(Sboxm(i)) \leftarrow i$
end for
```

B. Algorithm 2 (A2)

Input:

The secret key *key*(*i*), *i* = 1,…, *l* is the vector of *l* integer numbers (bytes) from the interval [0, 255].

The initial substitution box *Sbox* is (16x16)-size matrix of the different integer numbers (bytes) from the interval [0, 255].

Output:

The key-dependent substitution box *Sboxm*(*i*), *i* = 0,1,…,255 is the vector of the different integer numbers (bytes) from the interval [0, 255].

The key-dependent inverse substitution box *invSboxm*(*i*), *i* = 0,1,…,255 is the vector of different integer numbers (bytes) from the interval [0, 255].

```
1) 128 bits of the secret key key divide to the left ($key_1$) and right ($key_2$) equal parts.
2) The left part of the key $key_1$ divide into 16 equal parts $k_1(i)$, i = 1,2,...,16. $k_1(i)$ are the integer numbers from the interval [0,15].
3) for all i = 1,2,...,16 do
      cyclically rotate bytes of the rows of the initial substitution box Sbox to the left according to $k_1(i)$
   end for
   The result is the intermediate substitution box $Sbox_1$.
4) The right part of the key $key_2$ divide to the 16 equal parts $k_2(j)$, j = 1,2,...,16. $k_2(j)$ are the integer numbers from the interval [0,15].
5) for all j = 1,2,...,16 do
      cyclically rotate bytes of the columns of the intermediate substitution box $Sbox_1$ to the left according to $k_2(j)$.
   end for
   The result is the key dependent substitution box Sboxm.
6) Rearrange key-dependent substitution box ((16x16)-size matrix) Sboxm to the vector.
7) Compute the key-dependent inverse substitution box invSboxm:
   for all $i = 0,1,...,255$ do
      $invSboxm(Sboxm(i)) \leftarrow i$
   end for
```

C. Algorithm 3 (A3)

Algorithm 3 is a mixed version of Algorithm 1 and Algorithm 2. In that case, the initial substitution box *Sbox* is the input of the Algorithm 1. The output of the Algorithm 1 is the intermediate substitution box *Sboxm*1. The input of the Algorithm 2 is the intermediate substitution box *Sboxm*1. Finally, the output of the Algorithm 2 is the output of Algorithm 3, i.e., *Sboxm*.

D. Algorithm 4 (A4)

Algorithm 4 is a mixed version of Algorithm 2 and Algorithm 1. In that case, the initial substitution box *Sbox* is the input of the Algorithm 2. The output of the Algorithm 2 is the intermediate substitution box *Sboxm*2. The input of the Algorithm 1 is the intermediate substitution box *Sboxm*2. Finally, the output of the Algorithm 1 is the output of Algorithm 4, i.e., *Sboxm*.

III. Distance Metrics for Evaluation of the Quality of S-Boxes

We introduce several distance metrics that are able to calculate a distance between the given initial S-box *Sbox* and the key-dependent S-box *Sboxm*. We assume that the S-boxes are rearranged according to the rows to the *N*-size vectors. The key-dependent S-box *Sboxm* is key-dependent permutations without repetition of a given set of *N* integer elements of the initial box *Sbox*. The initial S-box *Sbox* and the key-dependent S-box *Sboxm* have the same length. For AES S-box *N* = 256. The *i*-th integer element of S-box is represented as *Sbox*(*i*). We have normalized all distances between the S-boxes. The smaller the value of the normalized distance, the more similar are the *Sbox* and *Sboxm*, and *vice versa*. For example, the normalized distance between *Sbox* and *Sbox* is equal to 0. For convenience, in this chapter we use indexes of S-boxes from 1 to *N* instead of from 0 until *N*–1.

1) The Hamming distance. The Hamming (H) distance between two S-boxes of equal length is defined as a number of not equal elements in the same positions in the initial S-box Sbox and in the key-dependent S-box Sboxm

$$d_H(Sbox, Sboxm) = \sum_{i=1}^{N} d_i,\ where\ d_i = \begin{cases} 1, if\ \ Sbox(i) \neq Sboxm(i) \\ 0, otherwise \end{cases} \tag{1}$$

The mean of the *H* distance is equal to (*N*+1)/2. The maximal *H* distance is equal to *N*. The normalized *H* distance $\bar{d}_H$ is obtained by dividing the distance d_H by the maximal *H* distance

$$\bar{d}_H = \frac{1}{N} d_H. \tag{2}$$

2) Spearman's distance. Spearman's (S) distance is an absolute distance between two rank vectors

$$d_S(Sbox, Sboxm) = \sum_{i=1}^{N} \sum_{j=1, i \neq j}^{N} \mid i - j \mid,$$

for such (*i*, *j*) that *Sbox* (*i*) = *Sboxm*(*j*) (3)

S distance is the summation of the absolute differences between two ranks of all equal elements from *Sbox* and *Sboxm*. *S* distance is similar to the Manhattan distance that used for quantitative variables. The mean of the *S* distance is equal to $N^2/3$. The *S* distance is maximal between *Sbox* and inverted *Sbox*, and is equal to $N^2/2$ for even *N* and to $(N^2-1)/2$ for odd *N*. The normalized *S* distance for even *N* is defined as

$$\bar{d}_S = \frac{2}{N^2} d_S \text{, for even } N. \tag{4}$$

3) Squared Spearman's distance. The squared Spearman's (SS) distance assigns a larger distance when deviations between equal elements of two S-boxes are larger. It is defined as follows:

$$d_{SS}(Sbox, Sboxm) = \sum_{i=1}^{N} \sum_{j=1, i \neq j}^{N} (i-j)^2,$$

$$for\ such\ (i,j)\ that\ Sbox(i) = Sboxm(j). \tag{5}$$

The mean of the *SS* distance is equal to $N^3/3$. The maximal *SS* distance is equal to $(N^3 - N)/3$. The normalized *SS* distance is defined as

$$\bar{d}_{SS} = \frac{3}{N^3 - N} d_{SS}. \tag{6}$$

The *SS* distance is similar to Spearman's rank correlation coefficient, a metric often used in statistics to calculate the correlation between two rankings.

4) The T distance. The T distance is the number of transpositions required to transform Sbox into *Sboxm*.

$$d_T(Sbox, Sboxm) = \sum_{i=1}^{N-1} \sum_{j=1}^{N-1} d_{ij}\text{, } where\ d_{ij} =$$

$$= \begin{cases} 1\ if\ \ Sbox(i) = Sboxm(j)\ \&\ Sbox(i+1) = Sboxm(j+1) \\ 0\ otherwise \end{cases} \tag{7}$$

The maximal value of d_T is *N*–1. The normalized *T* distance is defined as

$$\bar{d}_T = 1 - \frac{1}{N-1} d_T. \tag{8}$$

5) Kendall distance. The Kendall (K) distance is given by

$$d_K(Sbox, Sboxm) = \sum_{i=1}^{N} \sum_{j=1, i<j}^{N} d_{ij}\text{, } where\ d_{ij} =$$

$$= \begin{cases} 1\ \ if\ \ Sbox(i) < Sbox(j)\ \&\ Sboxm(j) < Sboxm(i) \\ 0\ \ otherwise \end{cases} \tag{9}$$

This distance is equal to the number of pair-wise adjacent permutations required to transform *Sbox* into *Sboxm*. The mean of the *K* distance is equal to $N^2/4$. The maximal value of *K* distance is $(N^2 - N)/2$. The normalized *K* distance is defined as

$$\bar{d}_K = \frac{2}{N^2 - N} d_K. \tag{10}$$

The normalized *K* distance lies in the interval [0,1]. For example, the normalized *K* distance 0.3 indicates that 30 % of the pairs of S-boxes elements differ in ordering between *Sbox* and *Sboxm*.

6) Correlation coefficient distance. We introduce the correlation (C) coefficient distance as follows: normalize Sboxm elements x=$\{x_1, ..., x_N\}$

$$y = \frac{x - mean(x)}{std(x)}, \tag{11}$$

and define the correlation coefficient of *Sboxm* elements as

$$d_C(Sboxm) = std(corr(\tau)), \tag{12}$$

where *mean* is the arithmetic mean, *std* is the standard deviation, *corr*(τ) is the correlation function of *y*. We assume, that *corr*(0) = 0. The maximal value of the correlation coefficient is *N*–1. We define the normalized correlation coefficient distance as follows:

$$\bar{d}_C = 1 - \frac{d_C}{N-1}. \tag{13}$$

7) Pearson's correlation coefficient distance. For the initial S-box Sbox with elements $\{x_1, ..., x_N\}$ and for the key-dependent S-box Sboxm with elements $\{y_1, ..., y_N\}$ the formula of the Pearson (P) correlation coefficient is [23]

$$r = \frac{N\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{N\sum x_i^2 - (\sum x_i^2)}\sqrt{N\sum y_i^2 - (\sum y_i)^2}}, \tag{14}$$

Pearson's correlation coefficient distance between initial S-box *Sbox* and key-dependent S-box *Sboxm* can be rescaled to a distance measure of range [0 – 1] by:

$$\bar{d}_P = 1 - \text{abs}(r). \tag{15}$$

The Pearson's correlation coefficient distance between two S-boxes $\bar{d}_P$ = 1 if correlation coefficient *r* is equal to zero and $\bar{d}_P$ = 0 if correlation coefficient *r* is equal to ± 1.

8) The longest common subsequence distance. The length of the longest common subsequence (LCS) is a measure of the similarity between Sbox and Sboxm. The minimum length of the LCS is equal to one and the maximum is equal to N. We define the LCS distance d_{LCS} (Sbox, Sboxm) as N minus the length of the longest common subsequence. The LCS distance lies between 0 and N–1. The normalized longest common subsequence distance $\bar{d}_{LCS}$ is defined as

$$\bar{d}_{LCS} = \frac{d_{LCS}}{N-1}. \tag{16}$$

IV. Experimental Results and Discussion

A. Experiment 1

Consider the 128 bit length secret key in hexadecimal form:

key = {CA, 6A, C5, 21, 5B, 46, 50, 3D, 98, 19, F0, 72, 6D, 41, 43, C7}. (17)

We have generated the key-dependent S-box *Sboxm* using Algorithm 2 and key (17). The initial S-box *Sbox* is the AES S-box (Table I). The key-dependent S-box *Sboxm* is given in Table II. We have assumed that S-boxes are rearranged according to the rows to the 256-size vectors, i.e., the initial substitution box *Sbox*(*i*), *i* = 0,1,…,255 and the key-dependent substitution box *Sboxm*(*i*), *i* = 0,1,…,255 are 256-size vectors of the different integer numbers from the interval [0, 255]. Thus, AES S-box (Table I) is the 256-size vector in hexadecimal form: {63, 7C,…,76; CA, 82,…,C0; … ; 8C, A1,…,16} and the permuted key-dependent S-box *Sboxm* (Table II) is the 256-size vector {56, D6,…, 6A; 45, DD,…, CC;…; C5, 52,…, 47}. Using eight metrics, we have calculated normalized distances between these two vectors. The normalized distances between initial AES S-box *Sbox* and the key-dependent S-boxes *Sboxm* are given in Table III. In the row „Algorithm2“ of the Table III we have used algorithm A2 for key-dependent permutation of the AES S-box, while in the row „*randperm*“ of the Table III we have used Matlab function „*randperm*“ for permutation of the AES S-box. From Table III we can see that all distances between the *Sbox* and *Sboxm* for algorithm "*randperm"* and for algorithm A2 are similar. It follows that the proposed algorithm A2 permutes the bytes of the AES S-box no worse than the Matlab function „*randperm*“. It confirms the good quality of the proposed key-dependent permutation algorithm A2.

B. Experiment 2

In order to evaluate the performance of our four algorithms, we have generated initial S-box *Sbox* – ordered integer numbers {0,1,…,255}. Then, using Matlab function "*randperm*" and Algorithms A1 – A4, we have calculated randomly permuted integer numbers (bytes) without repetition, i.e. key-dependent S-boxes *Sboxm.* After, we have evaluated eight normalized distances between initial *Sbox* and generated key-dependent S-boxes *Sboxm* for function "*randperm*" and for A1 – A4 algorithms. Such experiments we have repeated 1000 times with different randomly generated keys and have calculated the means and standard deviations of these normalized distances. We have used 128-bit length 1000 random keys, which we have generated using *Matlab* function "*randperm*". We have assumed the *Matlab* function "*randperm*" as standard of permutation of the integer numbers. Hence, we could evaluate the performance of our four algorithms comparing the averaged normalized distances $\bar{d}_i$ (i =1,…,8) of the function "*randperm*" with the averaged normalized distances $\bar{d}_i^{(j)}$ (i = 1,…,8; j = 1,…,4) of the proposed four algorithms using the measure (18)

$$\hat{d}^{(j)} = \frac{1}{8}\sum_{i=1}^{8}\frac{\left|\bar{d}_i - \bar{d}_i^{(j)}\right|}{\bar{d}_i}100\,\%, \qquad j = 1,2,3,4 \tag{18}$$

in which $\bar{d}_i$ is the normalized mean of *i*-th distance between initial *Sbox* and *Sboxm* in case we have used for permutation of the initial S-box "*randperm*" function; $\bar{d}_i^{(j)}$ is the normalized mean of *i*-th distance between initial *Sbox* and key-dependent *Sboxm* in case we have used for permutation of the initial S-box j-th algorithm.

TABLE I. AES S-BOX IN HEXADECIMAL FORM

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

TABLE II. KEY-DEPENDENT S-BOX *SBOXM*. GENERATION ALGORITHM IS A2. INITIAL S-BOX IS AES S-BOX. SECRET KEY IS AS IN (17)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	56	D6	7F	4A	D7	B2	9E	BA	32	AA	0A	1C	FC	F7	F0	6A
1	45	DD	BE	21	50	A4	2F	DF	70	C8	40	6D	48	4D	05	CC
2	78	13	2E	69	9D	5A	9A	BC	35	02	3D	10	D8	58	16	E1
3	1E	DA	14	64	27	9F	76	8C	3E	4F	66	BF	97	B1	A0	C4
4	C0	63	F8	3A	11	F2	2A	33	5B	46	99	7E	62	DE	E3	F3
5	CA	A1	37	0D	FA	06	38	85	C2	01	B8	EA	91	4C	19	15
6	F5	6C	D4	AC	1F	65	3C	0B	75	B7	7C	25	7B	36	D5	44
7	B5	7D	18	A6	90	17	E8	A5	F4	B9	4B	FF	E4	84	04	82
8	24	88	61	12	74	E9	86	5D	AE	CF	09	FD	98	26	1B	03
9	5E	8B	A8	53	C7	89	C3	20	D9	4E	5C	9B	3B	57	0F	CE
A	43	E6	B4	A9	41	CB	87	2B	B0	95	1D	D2	D0	83	77	1A
B	D1	C9	ED	92	6B	F6	C6	30	F9	2D	AF	FE	7A	28	73	51
C	5F	59	8E	0E	AD	B6	67	F1	9C	BD	BB	DB	CD	EF	93	FB
D	8F	22	3F	42	94	34	A7	A2	E2	71	C1	AB	79	60	A3	23
E	EB	55	72	08	E0	0C	2C	EC	49	96	6F	68	07	EE	E5	B3
F	C5	52	D3	80	39	29	54	31	8A	E7	81	00	DC	8D	6E	47

TABLE III. NORMALIZED DISTANCES BETWEEN AES S-BOX AND KEY-DEPENDENT S-BOX *S-BOXM*. GENERATION ALGORITHM IS A2. SECRET KEY IS AS IN (17)

Distance	$\bar{d}_H$	$\bar{d}_S$	$\bar{d}_{SS}$	$\bar{d}_T$	$\bar{d}_K$	$\bar{d}_C$	$\bar{d}_P$	$\bar{d}_{LCS}$
Algorithm2	0.9922	0.6282	0.4770	0.9961	0.4811	0.9605	0.9447	0.8980
"*randperm*"	0.9960	0.6368	0.4805	0.9960	0.4908	0.9560	0.9496	0.8939

In equation (18) and Table IV $\bar{d}_1$, $\bar{d}_1^{(j)}$ are Hamming distances $\bar{d}_H$; $\bar{d}_2$, $\bar{d}_2^{(j)}$ are Spearman's distances $\bar{d}_S$; $\bar{d}_3$, $\bar{d}_3^{(j)}$ are squared Spearman's distances $\bar{d}_{SS}$ and so on.

The proposed normalized correlation distance $\bar{d}_C$ and Pearson correlation distance $\bar{d}_P$ are similar, but standard deviation of the proposed distance $\bar{d}_C$ is about eight times less as compared with $\bar{d}_P$. From Table IV, and according with introduced quality measure (18), we can see that the best is Algorithm 3 – 0.3650 %, after follows Algorithm 4 – 0.4400 %, Algorithm 2 – 2.2849 % and Algorithm 1 – 2.4240 %. From Table V, it follows that Algorithm 1 generates 1000 S-boxes during 0.0940 sec., Algorithm 2 – during 0.2340 sec. and, finally, Algorithm 3 and Algorithm 4 – during 0.3280 sec.

TABLE IV. MEANS AND STANDARD DEVIATIONS OF 8 NORMALIZED DISTANCES BETWEEN INITIAL S-BOX *SBOX* (ORDERED INTEGER NUMBERS {0,1,…,255}) AND KEY-DEPENDENT S-BOXES *SBOXM*

i	**Algorithm** **Distance**	**„*randperm*“** $\bar{d}_i$	**A1** $\bar{d}_i^{(1)}$	**A2** $\bar{d}_i^{(2)}$	**A3** $\bar{d}_i^{(3)}$	**A4** $\bar{d}_i^{(4)}$
1	$\bar{d}_H$	0.9960 ± 0.0040	0.9936 ± 0.0326	0.9961 ± 0.0066	0.9960 ± 0.0039	0.9961 ± 0.0039
2	$\bar{d}_S$	0.6668 ± 0.0253	0.6381 ± 0.0347	0.6605 ± 0.0762	0.6586 ± 0.0274	0.6585 ± 0.0269
3	$\bar{d}_{SS}$	0.5005 ± 0.0308	0.4669 ± 0.0372	0.4969 ± 0.0562	0.4933 ± 0.0319	0.4899 ± 0.0314
4	$\bar{d}_T$	0.9961 ± 0.0038	0.9914 ± 0.0186	0.9393 ± 0.0553	0.9959 ± 0.0040	0.9959 ± 0.0480
5	$\bar{d}_K$	0.5008 ± 0.0204	0.4803 ± 0.0226	0.4927 ± 0.0376	0.4998 ± 0.0206	0.5008 ± 0.0211
6	$\bar{d}_C$	0.9560 ± 0.0031	0.9554 ± 0.0115	0.9495 ± 0.0078	0.9559 ± 0.0032	0.9559 ± 0.0031
7	$\bar{d}_P$	0.9496 ± 0.0368	0.9246 ± 0.0599	0.9125 ± 0.0660	0.9495 ± 0.0376	0.9487 ± 0.0382
8	$\bar{d}_{LCS}$	0.8939 ± 0.0081	0.8861 ± 0.0280	0.8519 ± 0.0313	0.8940 ± 0.0082	0.8949 ± 0.0080
	$\hat{d}^{(j)}$		2.4240 %	2.2849 %	0.3650 %	0.4400 %

TABLE V. GENERATION TIME OF 1000 KEY-DEPENDENT S-BOXES *S-BOXM* WITH COMPUTER AMD ATHLON-X2, 2.59 MHZ

Algorithm	A1	A2	A3	A4	*“randperm”*
Time, sec.	0.0940	0.2340	0.3280	0.3280	0.0359

V. CONCLUSIONS

This paper proposes a new simple algorithms to generate key-dependent S-boxes. The generated key-dependent S-boxes can be used in block ciphers such as AES cipher. We suggest for testing key-dependent S-boxes to apply eight distance metrics. The results of the experiments and tests show that the new generated S-boxes are truly random. Using our algorithms, we can get 256! different substitution values instead of 256 values as it is in AES S-box. It increases the encryption complexity and aggravate the cryptanalysis process. It was established that for any changing secret key, the structure of the key-dependent S-boxes are changing essentially. Also it was shown that this is achieved with negligible time delay. For example, if we use algorithm A1, 1000 S-boxes were generated during 0.0940 sec. These algorithms will increase the security of the cipher systems. As compared with algorithm in the paper [14], these algorithms generate S-boxes about eight times faster. For measure of the quality of the permuted S-boxes we have proposed to use eight distance metrics. The distances between initial S-boxes and key-dependent S-boxes of our algorithms we have compared with appropriate distances of the *Matlab* function “*randperm*”. We have assumed this function as standard of permutation of the integer numbers (bytes) of S-boxes. Also it was found that the proposed new correlation distance metric $\bar{d}_C$ as compared with Pearson distance metric $\bar{d}_P$ is about eight times more accurate.

REFERENCIES

[1] Data Encryption Standard (DES) National Bureau of Standards. FIPS Publication, 1977.

[2] Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001.

[3] P. Su, T. F. Lin, C. T. Huang, and C. W. Wu, “A high-throughput low-cost AES processor,” IEEE Communications Magazine, vol. 41, pp. 86-91, 2003.

[4] B. Schneier, ”Description of a new variable-length 64-bit block cipher,” Fast Software Encryption, pp.191-204, 1997.

[5] G. Jakimovski and L. Kocarev, “Chaos and cryptography: block encryption ciphers based on chaotic maps,” IEEE Transaction on Circuits and Systems, Part I, vol. 48, pp. 163-169, 2001.

[6] N. Masuda, G. Jakimovski, K. Aihara, and L. Kocarev, “Chaotic block ciphers: from theory to practical algorithms,” IEEE Trans. on Circuits and Systems – I: Regular Papers, vol. 53, pp. 1341-1352, 2006.

[7] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley, 1996.

[8] J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. CRC, 1997.

[9] L. Keliher, Linear cryptanalysis of substitution-permutation networks. PhD Thesis, Queen’s University, Kingston, Canada, 2003.

[10] M. Mahmoud, A. B. El Hafez, T. A. Elgarf, and A. Zekry, “Dynamic AES-128 with key-dependent S-box,” Int. J. of Engineering Research and Applications, vol.3, pp.1662-1670, 2013.

[11] J. Juremi, R. Mahmod, S. Sulaiman, and J. Ramli, “Enhancing Advanced Encryption Standard S-box generation based on round key,” Int. J. of Cyber-Security and Digital Forensics, vol.1, pp.183-188, 2012.

[12] M. El-Sheikh, O. A. El-Mohsen, and A. Zekry, “A new approach for designing key-dependent S-box defined over GF in AES,” Int. J. of Computer Theory and Engineering, vol. 4, pp.158-164, 2012.

[13] Y. Mohammad, A. E. Rohiem, and A. D. Elbayoumy, “A novel S-box of AES algorithm using variable mapping technique,” 13 Int. Conference on Aerospace Sciences & Aviation Technology, May 2009, Cairo, Egypt, 1/9-10/9, 2009.

[14] K. Kazlauskas and J. Kazlauskas, “Key-dependent S-box generation in AES block cipher system,” Informatica, vol. 20, pp. 23-34, 2009.

[15] N. Hamdy, K. Shehata, and H. Eldemerdash, “Design and implementation of encryption unit based on customized AES algorithm,” Int. J. of Video & Image Processing and Network Security, vol. 11, pp.33-40, 2011.

[16] R. Hosseinkhani and H. S. Javadi, “Using cipher key to generate dynamic S-box in AES cipher system,” Int. J. of Computer Science and Security, vol. 6, pp. 19-28, 2012.

[17] R. Merkle, “Fast software encryption functions,” Advances in Cryptology: Proceedings of CRYPTO’90, Springer-Verlag, Berlin, pp. 476-501, 1991.

[18] K. Kazlauskas, G. Vaicekauskas, and R. Smaliukas, “An algorithm for key-dependent S-box generation in block cipher system,” Informatica, vol. 26, pp. 51-65, 2015.

[19] D. Lambic, „A novel method of S-box design based on chaotic map and composition method“Chaos, Solitons & Fractals, vol. 58, pp.16-21, 2014.

[20] M. Khan, T. Shah, H. Mahmood, and M. A. Gondal, „An efficient method for the construction of block cipher with multi-chaotic systems,“ Nonlinear Dynamics, vol. 71, pp. 489-492, 2013.

[21] D. Lambic, „Security analysis and improvement of a block cipher with dynamic S-boxes based on tent map,“ Nonlinear Dynamics, vol. 79, pp. 2531-2539, 2015.

[22] F. Ozkaynak and S. Yavuz, „Analysis and improvement of a novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system,” Nonlinear Dynamics, vol.78, pp.1311-1320, 2014.

[23] N. A. Rahman, A Course in Theoretical Statistics. Charles Griffin and Company, 1968.