INFORMATICA, 2009, Vol. 20, No. 1, 23–34

Key-Dependent S-Box Generation in AES Block Cipher System

Kazys KAZLAUSKAS, Jaunius KAZLAUSKAS
Institute of Mathematics and Informatics
Akademijos 4, 08663 Vilnius, Lithuania
e-mail: kazlausk@ktl.mii.lt

Received: June 2008; accepted: September 2008

Abstract. Advanced Encryption Standard (AES) block cipher system is widely used in cryptographic applications. A nonlinear substitution operation is the main factor of the AES cipher system strength. The purpose of the proposed approach is to generate the random S-boxes changing for every change of the secret key. The fact that the S-boxes are randomly key-dependent and unknown is the main strength of the new approach, since both linear and differential cryptanalysis require known S-boxes. In the paper, we briefly analyze the AES algorithm, substitution S-boxes, linear and differential cryptanalysis, and describe a randomly key-dependent S-box and inverse S-box generation algorithm. After that, we introduce the independency measure of the S-box elements, and experimentally investigate the quality of the generated S-boxes.

Keywords: advanced encryption standard, key-dependent S-boxes, generation algorithm.

1. Introduction

Cryptography has an important role in the security of data transmission and is the best method of data protection against passive and active fraud. The growing number communication users has led to increasing demand for security measures to protect data transmitted over open channels (Chen *et al*., 2008; Li *et al*., 2007; Sakalauskas, 2005). A cipher system is a set of reversible transformations from the set M of a plaintext into the set C of a ciphertext. Each transformation depends on a secret key and the ciphering algorithm. In the block cipher system, the plaintext is divided into the blocks and the ciphering is carried out for the whole block (El-Ramly *et al*., 2001).

Two general principles of block ciphers are diffusion and confusion. Diffusion is spreading of the influence of a one plaintext bit to many ciphertext bits with intention to hide the statistical structure of the plaintext. Confusion is transformation that change dependence of the statistics of ciphertext on the statistics of plaintext. In most cipher systems the diffusion and confusion is achieved by means of round repetition. Repeating a single round contributes to cipher's simplicity (Masuda *et al*., 2006). Modern block ciphers consist of four transformations: substitution, permutation, mixing, and key-adding (Schneier, 1996; Menezes *et al.*, 1997).

Cryptographic objects are private key algorithms, public key algorithms and pseudorandom generators. Block ciphers transform usually the 128 or 256 bits string to a string

of the same length under control of the secret key. Private key cryptography, such as DES (DES, 1977), 3DES, and Advanced Encryption Standard (AES) (AES, 2001), uses the same key for the sender and receiver to encrypt the plaintext and decrypt the ciphertext. Private key cryptography is more suitable for the encryption of a large amount of data. Public key cryptography, such as the Rivest-Shamir-Adleman (RSA) or Elliptic Curve algorithms, uses different keys for encryption and decryption. The AES algorithm defined by the National Institute of Standards and Technology of the United States has been accepted to replace DES as the new private key encryption algorithm. AES overpass DES in improved security because of larger key sizes. AES is suitable for 8 bit microprocessor platforms and 32 bit processors (Su *et al*., 2003).

Block cipher systems depend on the S-boxes, which are fixed and have no relation with the secret key. So only changeable parameter is the secret key. Since the only nonlinear component of AES is S-boxes, they are an important source of cryptographic strength.

The use of key-dependent S-boxes in block cipher design has not been widely investigated in the literature. Research into 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 (Menezes *et al.*, 1997). Some results have demonstrated that a randomly chosen S-box of sufficient size will have several of these desirable properties with high probability (Keliher, 2003).

This paper outlines the work of the authors' investigation into the design of a new pseudo-randomly generated key-dependent S-boxes. Other systems using key-dependent S-boxes have been proposed in the past, the most well-known is Blowfish (Schneier, 1996) and Khufu (Merkle, 1991). Each of these two systems uses the cryptosystem itself to generate the S-boxes. Preliminary results show, that our proposed algorithm has good cryptographic strength, with the added benefit that is resistant to linear and differential cryptanalysis, which require that the S-boxes be known.

A new method to generate pseudo-random S-boxes as a function of the secret key will be presented. In the next section, we briefly introduce the AES algorithm. In the following two sections, we analyze AES S-boxes, differential and linear cryptanalysis. A central part of the paper describes the pseudo-randomly key-dependent S-box and inverse S-box generation algorithm. After that, the paper discusses experimental results and gives conclusions.

2. The AES Algorithm

The AES is a private key block cipher that processes data blocks of 128 bits with key length of 128, 192, or 256 bits. The AES algorithm's operations are performed on a 2-D array of 4 $times$ 4 bytes called the State. The initial State is the plaintext and the final State is the ciphertext. The State consists of 4 rows of bytes. As the block length is 128 bits, each row of the State contains 4 bytes. The four bytes in each column form a 32 bit word. After an initial round key addition, a round function consisting of four transformations – SubBytes, ShiftRows, MixColumns, and AddRoundKey is applied to

each data block. The round function is applied 10, 12, or 14 times depending on the key length. AES-128 applies the round function 10 times, AES-192 – 12 times, and AES-256 – 14 times. The transformations are reversible linear and non-linear operations to allow decryption using their inverses. Every transformation affects all bytes of the State. The transformation SubBytes is a nonlinear byte substitution that operates on each byte of the State using a table (S-box). The numbers of the table is computed by a finite field inversion followed by an affine transformation. The resulting table is called an S-box. The ShiftRows transformation is a circular shifting operation, which rotates the rows of the State with different numbers of bytes (offsets). The offset equals to the row index: the second row is shifted one byte to the left, the third row – two bytes to the left, the fourth row – three bytes to the left, and first row – four bytes to the left. MixColumns transformation mixes the bytes in each column by multiplying the State with the polynomial modulo x^4+1. The State bytes are the coefficients of the polynomial. The AddRoundKey transformation is an XOR operation that adds the round key to the State in each round. The round keys are generated during the key expansion process. The initial round key equals to secret key (Zhang and Parhi, 2002; Su *et al*., 2003; Hsiao *et al*., 2005; Feldhofer *et al*., 2005).

3. Substitution S-Boxes

Substitution is a nonlinear transformation which performs confusion of bits. A nonlinear transformation is essential for every modern encryption algorithm and is proved to be a strong cryptographic primitive against linear and differencial cryptanalysis. Nonlinear transformations are implemented as lookup tables (S-boxes). An S-box with p input bits and q output bits is denoted $p \rightarrow q$. The DES uses eight $6 \rightarrow 4$ S_boxes. S-boxes are designed for software implementation on 8-bit processors. The block ciphers with $8 \rightarrow 8$ S-boxes are SAFER, SHARK, and AES. For processors with 32-bit or 64-bit words, S-boxes with more output bits provide high efficiency. The Snefru, Blowfish, CAST, and SQUARE use $8 \rightarrow 32$ S-boxes. The S-boxes can be selected at random as in Snefru, 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. S-boxes that depend on key values are slower but more secure than key independent ones (Schneier, 1996). Use of key independent chaotic S-boxes are analyzed in (Jakimovski and Kocarev, 2001), in which the S-box is constructed with a transformation $F((X + K) \bmod M)$, where K is the key (Masuda *et al*., 2006).

In the AES, the S-box generate two transformations in the Galois fields GF(2) and GF(2^8). S-box is a nonlinear transformation where each byte of the State is replaced by another byte using the substitution table. The first transformation: S-box finds the multiplication inverse of the byte in the field GF(2^8). Since it is a algebraic expression, it is possible to mount algebraic attacks. Hence, it is followed by an affine transformation. The affine transformation is chosen in order to make the SubBytes a complex algebraic

expression while preserving the nonlinearity property. The both S-box transformations can be expressed in a matrix form as (Hsiao *et al*., 2005; Hsiao *et al*., 2006)

$$S' = M \bullet S^{-1} + C, \tag{1}$$

where the sign $\bullet$ is multiplication and the sign + is addition in the field GF(2^8). The 8×1 vector S' denotes the bits of the output byte after the S-box transformations. The inverse S-box transformation can be get by multiplying both sides of equation (1) by M^{-1} and it performs the inverse affine transformation followed by the multiplicative inverse in GF(2^8):

$$S^{-1} = M^{-1} \bullet S' + M^{-1} \bullet C. \tag{2}$$

4. Linear and Differential Cryptanalysis

Linear and differential cryptanalysis uses the input-output correlation and the difference propagations of the cipher in order to extract partial or whole bits of the secret key. Linear cryptanalysis exploits a cipher's weakness expressed in terms of "linear expressions". In Matsui's terminology (Matsui, 1994) a linear expression for one round is an equation for a certain modulo two sum of round input bits and round outputs bits as a sum of round key bits. The expression should be satisfied with probability much more than 0.5 to be useful (Jakimovski and Kocarev, 2001).

In 1991 was introduced a cryptoanalytic technique known as differential cryptanalysis (Biham and Shamir, 1991). It was successfully applied to attack a variety of SPNs, including DES. Differential cryptanalysis requires knowledge of the XOR tables of S-boxes. For an $n \times n$ S-box, S, the XOR table has rows and columns indexed by $0, 1, \ldots, 2^n - 1$, and the table entries are defined as follows: if $i, j \in \{0, 1, \ldots, 2^n - 1\}$, position (i, j) in the XOR table contains value $|\{X \in \{0,1\}^n\colon\ S(X) \oplus S(X \oplus i) = j\}|$, where i and j are n bits strings. The essential part of every block cipher is an S-box. To secure the cipher against these attacks, the nonlinearity of the S-box should satisfy: the maximum input-output correlation and the difference propagation probability should be minimum (Keliher and Meijer, 1997).

There are two ways to fight against linear and differential cryptanalysis. One is built 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 occur. The round transformation must be designed in such a way that differential steps with few active S-boxes are followed by differential steps with many active S-boxes (Masuda *et al.*, 2006).

The object of this proposal is an AES cipher using key-dependent S-boxes. The fact that the S-boxes are unknown is one of the main strength of our cipher system, since both linear and differential cryptanalysis require known S-boxes. If the S-boxes are generated from the key in sufficiently random fashion, each S-box has a high probability of being complete, possessing fairly high nonlinearity. It is not apparent that the pseudorandom nature of the S-boxes introduces any weakness into the system. Ideal randomness

of S-box cannot be achieved. Ideal randomness is not mathematically possible for the following reasons: the value of all elements in the S-box difference table should be even, since $a \oplus b = b \oplus a$. Since the S-box is bijective, the input difference of 0 will lead to an output difference of 0. So the element corresponding to row $= 0$ and column $= 0$ at the difference table will be 2^n and all other elements in row $= 0$ and column $= 0$ will be 0 (Sakthivel, 2001).

5. A Randomly Key-Dependent S-Box and Inverse S-Box Generation Algorithm

A randomly key-dependent S-box and inverse S-box is constructed by composing two transformations: key pseudo-expansion and key-dependent S-box and inverse S-box generation.

5.1. *Key Pseudo-Expansion*

The key pseudo-expansion transformation takes the 16, 24, or 32 bytes secret key, use round constant *rcon* and generates a 176 byte long key pseudo-schedule b, which we use in the second transformation. The bytes of the secret key are row-wise rearranged into a 4×4 size key matrix. The method of the key pseudo-expansion is an element-wise XOR of the row of key matrix and the row four rows up. In every fourth row, before applying the XOR, the initial row is cyclically rotated and XOR-ed with constant *rcon* (AES, 2001).

Algorithm 1. Key pseudo-expansion
Input: key, rcon.
Output: key-dependent 176 bytes b.

1: $b = (\text{reshape}(\text{key}, 4, 4))'$
2: **for** $i = 5, \ldots, 44$ **do**
3: $laik = b(i-1, :)$
4: **if** $i \bmod 4 = 1$
5: $laik = laik([2\ 3\ 4\ 1])$
6: $v = rcon((i-1)/4, :)$
7: $laik = \text{XOR}(laik, v)$
8: **end if**
9: $b(i, :) = \text{XOR}(b(i-4, :), laik)$
10: **end for**

Input of the key pseudo-expansion algorithm is the secret key and round constant *rcon*. Output of the key pseudo-expansion algorithm is the key pseudo-expanded schedule b (176 bytes). Command *reshape* in line 1 rearrange the bytes of the secret key into a 4×4 size matrix. Sign $'$ means transpose operation. The remaining 160 bytes of the key pseudo-schedule are generated in the **for . . . end for** cycle (lines 2–10). In line 3 *laik* is the previous row, which in line 9 is XOR-ed with the row four rows before. If the row index is equal to $5, 9, \ldots$, then *laik* is cyclically rotated and XOR-ed in line 7 with the

round constant *rcon*. Rotating is done in line 5, where the *laik* is a row of four bytes and is cyclically permuted by using the new index vector [2 3 4 1]. The hexadecimal numbers of the first column of the 10×4 matrix *rcon* are {01, 02, 04, 08, 10, 20, 40, 80, 1b, 36}. Remaining three columns of the matrix *rcon* are zero columns.

Note 1. If the secret key length is 192 or 256 bytes, we use only the first 128 bytes of the key.

Note 2. The key pseudo-expansion algorithm don't use the SubByte function and S-box, while key expansion function in AES uses S-box and SubByte function (AES, 2001).

5.2. *Key-dependent S-Box and Inverse S-Box Generation*

The output of the key pseudo-expansion algorithm is 176 byte long key pseudo-schedule b, which is the function of the key. The key pseudo-schedule b is used to generate a randomly key-dependent S-box and inverse S-box.

Algorithm 2. Generation a randomly key-dependent S-box and inverse S-box.

Input: key-dependent 176 bytes b from the output of the key pseudo-expansion algorithm.

Output: integer numbers from the interval [0, 255] of the key-dependent S-box (*Sbox*) and the inverse S-box ($invSbox$)

1: Initialization:
$i = 0$
$k = 1$
$l = 1$
2: Compute the first subtotal modulo 256:
$S(1) = (b(1) + b(2)) \bmod 256$
$Sbox(1) = S(1)$
3: **while** $k < 256$
$i = i + 1$
$m = 1 + (k + i * l) \bmod 176$
$S(i + 1) = (S(i) + b(m)) \bmod 256$
$l = 0$
4: **for** $j = 1, \ldots, k$ **do**
Compare subtotal $S(i+1)$ with the elements *Sbox* (j) and count the number l of the S-box elements which are not equal to $S(i + 1)$
end for
5: **if** $l = j$
$Sbox(k + 1) = S(i + 1)$
$k = k + 1$
end if
end while
6: **for** $k = 1, \ldots, 256$ **do**
$invSbox(Sbox(k) + 1) = k - 1$
end for

Note 3. Randomness of the key-dependent S-box and inverse S-box is achieved by choosing the index m of the bytes b, which depends on the variables i, k, and l.

6. Experimental Results

We introduce the independency measure of the S-box elements:

1) Normalize the S-box elements

$$y = \frac{x - mean(x)}{std(x)}, \tag{3}$$

where $mean(x) = (\max(x) + \min(x))/2$, $std(x) = sqrt((\max(x) - \min(x))^2/12)$.

2) Compute the correlation function *corr* of the normalized S-box.

3) Find the maximal value *MAX* of the correlation function and form a new function *CORR*, in which the maximal value *MAX* of the correlation function *corr* is changed to zero.

4) Calculate the independency measure of the S-box elements

$$ratio = \frac{std(CORR)}{MAX}. \tag{4}$$

Experiment 1. We experimentally checked how the independency measure *ratio* of the S-box elements depends on the interval length. For the generation of the random integer numbers from some interval we applied Matlab function *randperm*. We repeated the experiment 500 times and calculated mean and standard deviation of the *ratio*. The results are given in the Table 1.

Table 1

The mean of the *ratio* of the dependent and independent integer numbers via the interval length

Interval length	Mean of the *ratio* of the linearly dependent numbers	Mean of the *ratio* of the independent numbers
8	$0.362427 \pm 1.6737 \cdot 10^{-16}$	0.230258 ± 0.0627
16	$0.388781 \pm 5.0212 \cdot 10^{-16}$	0.166652 ± 0.0381
32	$0.401525 \pm 4.4632 \cdot 10^{-16}$	0.125118 ± 0.0206
64	$0.407811 \pm 5.0211 \cdot 10^{-16}$	0.086057 ± 0.0118
128	$0.410932 \pm 5.5791 \cdot 10^{-16}$	0.062037 ± 0.0069
256	$\mathbf{0.412488 \pm 7.2528} \cdot 10^{-16}$	$\mathbf{0.044390 \pm 0.0032}$
512	$0.413264 \pm 7.2528 \cdot 10^{-16}$	0.031398 ± 0.0015
1024	$0.413652 \pm 6.6949 \cdot 10^{-16}$	$0.022096 \pm 8.3026 \cdot 10^{-4}$
2048	$0.413846 \pm 2.7895 \cdot 10^{-16}$	$0.015646 \pm 4.4133 \cdot 10^{-4}$
4096	$0.413942 \pm 5.5791 \cdot 10^{-16}$	$0.011044 \pm 1.9339 \cdot 10^{-4}$
40960	$0.414029 \pm 1.6737 \cdot 10^{-16}$	$0.003497 \pm 2.0673 \cdot 10^{-5}$
409600	$0.414038 \pm 8.9265 \cdot 10^{-16}$	$0.001105 \pm 2.0026 \cdot 10^{-6}$

From Table 1 we can see, that *ratio* depends on the interval length of the integer numbers. For the AES, the S-box elements are integers from the interval [0, 255]. In that case, the measure *ratio* for randomly generated elements has value 0.044390 ± 0.0032, and *ratio* has the maximal value 0.4124876 $\pm 7.2528 \cdot 10^{-16}$ for linearly dependent elements. If we analyze only the interval [0,255] (16×16 S-box), and suppose that function *randperm* permutes the integer numbers "ideally", the *ratio* = 0.0443904 ± 0.0032 can be used as an "ideal" *ratio* for the random 16×16 S-box. So, the *ratio* can be used as the independency measure of the integer numbers from the finite interval. Theoretically, in case the interval of the integer numbers is infinitely long, for the independent integer numbers the *ratio* is equal to zero.

Experiment 2. The purpose is to verify the proposed S-box and inverse S-box generation algorithm. Consider the 128 bit length secret key in the hexadecimal form:

$$key\_hex = \{2f, e3, c2, d3, bf, e5, 56, c7, f8, 79, dc, cb, 8c, fd, 6e, 5f\}. \tag{5}$$

1) We generate the key-dependent S-box using the transformations described in the paper. The key-dependent S-box is given in the Table 2. The independency measure *ratio* of the S-box elements is equal to 0.0432548 and is approximately equal to "ideal" *ratio* = 0.044390 for independent numbers generated using function *randperm*. The correlation function of the normalized S-box is in the Fig. 1.

2) We change two bits in the secret key (5), for example, $2f \to 1f$:

$$key\_hex = \{1f, e3, c2, d3, bf, e5, 56, c7, f8, 79, dc, cb, 8c, fd, 6e, 5f\}, \tag{6}$$

Table 2
Key-dependent S-box. Key is as in (5)

238	111	194	228	165	51	54	157	216	18	202	92	74	129	188	250
230	154	0	72	208	151	125	24	83	146	114	218	121	227	21	2
185	37	17	8	214	134	220	80	39	181	6	128	207	77	191	231
10	118	96	79	187	226	50	247	110	15	245	171	86	81	126	224
234	3	53	108	65	192	35	26	150	14	57	144	167	38	106	91
236	89	205	88	155	55	104	178	85	215	123	193	76	229	112	56
75	44	163	97	174	52	199	243	30	133	143	232	137	61	87	211
179	235	241	223	4	59	201	42	90	225	162	47	22	139	253	13
136	120	82	170	101	180	102	189	23	78	45	64	63	131	198	153
183	210	169	251	43	115	173	46	244	164	168	116	20	212	9	33
200	93	100	36	49	130	109	132	66	248	246	41	124	221	254	141
28	95	204	252	69	27	209	1	71	242	122	103	213	196	190	67
182	34	135	219	140	48	147	98	60	177	142	148	237	195	175	99
5	186	70	166	184	117	7	107	239	217	62	176	159	138	11	113
68	249	105	149	127	58	73	145	32	19	84	197	94	156	161	160
172	40	25	158	222	31	16	206	12	255	152	233	29	240	119	203

and generate another S-box (Table 3). In that case the *ratio* is equal to 0.043231 and is approximately equal to "ideal" *ratio*. The correlation function of the normalized S-box (Table 3) is similar to correlation function of the key (5) (see Fig. 1).

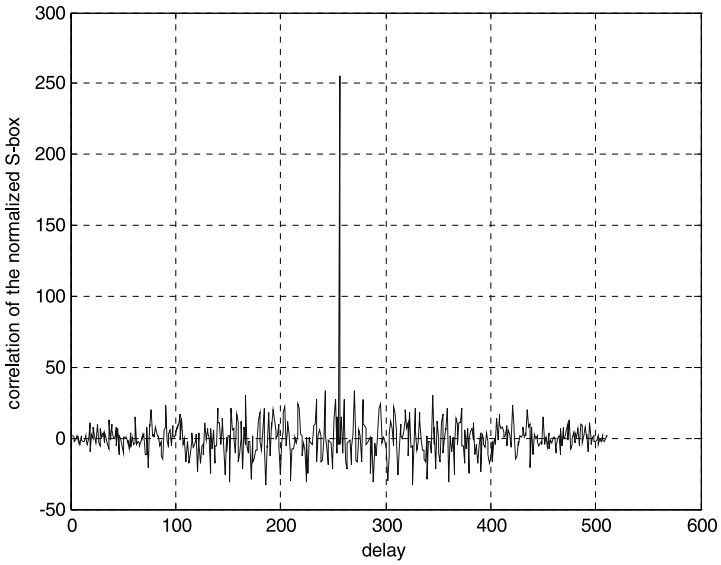

Fig. 1. Correlation function of the normalized S-box (Table 2).

Table 3
Key-dependent S-box. Key is as in (6)

222	84	187	195	174	14	78	45	197	181	121	242	129	142	186	90
214	184	21	172	109	83	63	92	244	252	95	238	51	0	249	70
185	246	18	183	29	196	6	81	162	223	73	221	23	55	126	149
10	44	139	103	19	67	31	164	251	225	71	76	178	148	191	87
218	247	156	151	36	153	117	143	171	110	88	182	146	165	39	27
220	144	175	204	37	239	115	235	79	202	245	135	208	42	253	22
75	161	188	52	100	17	123	96	133	65	233	82	40	212	48	113
179	134	206	125	35	25	50	207	190	46	89	154	38	118	102	111
152	157	227	180	2	114	193	147	250	54	119	105	145	9	32	192
199	226	229	16	200	49	41	243	13	173	176	4	255	234	205	159
232	213	241	108	254	47	136	91	15	216	28	7	120	69	94	80
60	177	248	168	201	170	198	141	194	20	26	85	56	12	231	30
230	124	3	59	132	203	1	215	66	43	33	127	57	106	237	86
53	101	62	140	236	99	58	128	240	112	68	189	211	150	77	158
116	155	166	104	228	130	5	217	167	137	163	72	131	97	98	107
219	138	209	93	64	24	61	160	11	169	210	8	74	122	224	34

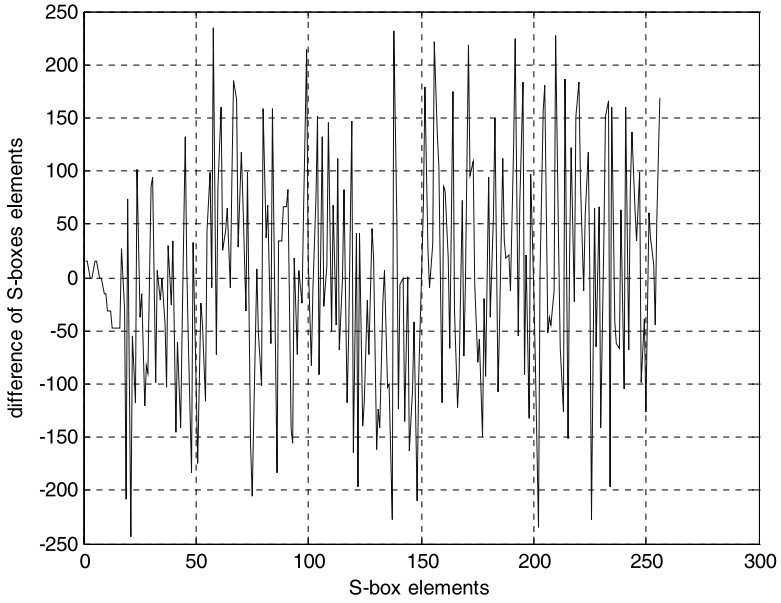

Fig. 2. Plot of the difference of the S-boxes elements (Table 2 and Table 3).

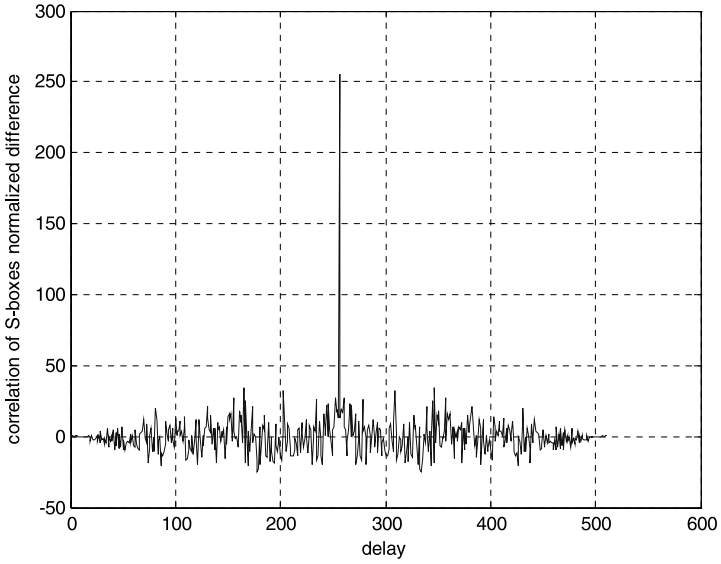

Fig. 3. Correlation function of the normalized difference between the elements of two S-boxes (Table 2 and Table 3).

We calculate the difference of these two S-boxes (Fig. 2) and mutual independency *ratio* = 0.040642, which is approximately equal to "ideal" independency *ratio* = 0.044390, i.e., the mutual dependence of these two S-boxes is insignificant. Correlation of the normalized difference between the elements of S-boxes (Table 2 and Table 3) are given in the Fig. 3.

3) We change only one bit of the secret key (5) ($2f \rightarrow 3f$), generate the S-box and calculate difference among the elements of this S-box and the elements of Table 2. The mutual independency *ratio* of these two S-boxes is 0.0445, i.e., approximately is equal to "ideal" *ratio*, so the mutual dependence of these two S-boxes is insignificant.

7. Conclusions

We presented a new approach to generate the AES randomly key-dependent S-boxes. The quality of this approach is tested by changing only one bit of the secret key to generate new S-boxes. For that purpose the independency measure *ratio* based on correlation method was introduced. It was established that for any change of the secret key, the structure of the S-box will be changed essentially. The randomly key-dependent S-boxes make our approach resistant to linear and differential cryptanalysis. This approach will lead to generate more secure block ciphers, solve the problem of the fixed structure S-boxes, and will increase the security level of the AES block cipher system. The main advantage of such approach is that an enormous number of S-boxes can be generated by changing secret key.

Acknowledgements

The authors thank anonymous reviewer for the comments and suggestions that contributed to improve the quality of the paper. Research in part was supported by the Lithuanian State Science and Studies Foundation.

References

Advanced Encryption Standard (AES) (2001). *Federal Information Processing Standards*. Publication 197, November 26.

Biham, E., and A. Shamir (1991). Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, **4**(1), 3–72.

Chen, T.-H., G. Horng and Ch.-S. Yang (2008). Public key autentification schemes for local area networks. *Informatica*, **19**(1), 3–16.

Data Encryption Standard (DES) (1977). *National Bureau of Standards*. FIPS Publication 46.

El-Ramly, S.H., T. El-Garf and A.H. Soliman (2001). Dynamic generation of S-boxes in block cipher systems. In *Eighteen National Radio Science Conference*. Mansoura Univ., Egypt. pp. 389–397.

Feldhofer, M., J. Wolkerstorfer and V. Rijmen (2005). AES implementation on a grain of sand. *IEEE Proc. Inf. Secur.*, **152**(1), 13–20.

Hsiao, S.-F., M.-C. Chen, M.-Y. Tsai and C.-C. Lin (2005). System on chip implementation of the whole advanced encryption standard processor using reduced XOR-based sum-of-product operations. *IEEE Proc. Inf. Secur.*, **152**(1), 21–30.

Hsiao, S.-F., M.-C. Chen and C.-S. Tu (2006). Memory-free low-cost designs of advanced encryption standard using common subexpression elimination for subfunctions in transformations. *IEEE Trans. on Circuits and Systems* – I: *Regular Papers*, **53**(3), 615–626.

Jakimovski, G., and L. Kocarev (2001). Chaos and cryptography: block encryption ciphers based on chaotic maps. *IEEE Trans. on Circuits and Systems* – I: *Fundamental Theory and Applications*, **48**(2), 163–169.

Keliher, L. (2003). *Linear Cryptanalysis of Substitution-Permutation Networks*. PhD thesis, Queen's University, Kingston, Canada.

Li, Ch.-M., T. Hwang and N.-Y. Lee (2007). Security flow in simple generalized group-oriented cryptosystem using ElGamal cryptosystem. *Informatica*, **18**(1), 61–66.

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

Matsui, M. (1994). Linear cryptanalysis method for DES ciphers. In *Advances in Cryptology – EUROCRYPT'93, Berlin, Germany*. Springer-Verlag. pp. 386–397.

Menezes, A.J., P.C. van Oorschot and S.A. Vanstone (1997). *Handbook of Applied Cryptography*. Boca Raton, FL: CRC.

Sakalauskas, E. (2005). One digital signature scheme in semomodule over semiring. *Informatica*, **16**(3), 383–394.

Merkle, R. (1991). Fast software encryption functions. In *Advances in Cryptology*: *Proceedings of CRYPTO'90*. Springer-Verlag, Berlin. pp. 476–501.

Sakthivel, G. (2001). *Differential Cryptanalysis of Substitution Permutation Networks and Rijndael-Like Ciphers*. Master's project report. Rochester Institute of Technology.

Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley, New York.

Schneier, B. (1996). Description of a new variable-length 64-bit block cipher. *Fast Software Encryption*, 191–204.

Su, C.-P., T.-F. Lin, C.-T. Huang and C.-W. Wu (2003). A high-throughput low-cost AES processor. *IEEE Communications Magazine*, **41**(12), 86–91.

Zhang, X., and K.K. Parhi (2002). Implementation approaches for the advanced encryption standard algorithm. *IEEE Circuits Syst. Mag*., **2**(4), 24–46.

K. Kazlauskas received a PhD degree from Kaunas Polytechnic Institute and a doctor habilius degree from Institute of Mathematics and Informatics and Vytautas Magnus University. He is a senior researcher of the Recognition Processes Department at the Institute of Mathematics and Informatics and a professor at the Informatics Department of Vilnius Pedagogical University. His research interests include signal processing, parameter estimation, and digital system design.

J. Kazlauskas received a MSc degree from Vilnius University in 2000. Now he is a junior researcher of the Process Recognition Department at the Institute of Mathematics and Informatics. His scientific interests include digital signal processing and parameter estimation.

AES blokinės šifravimo sistemos nuo rakto priklausomų S lentelių generavimas

Kazys KAZLAUSKAS, Jaunius KAZLAUSKAS

AES blokinė šifravimo sistema plačiai naudojama praktikoje. Šios sistemos atsparumas labai priklauso nuo netiesinės baitų pakeitimo operacijos. Straipsnio tikslas – pasiūlyti algoritmą, kuris generuotų atsitiktines priklausomas nuo rakto baitų pakeitimo S lenteles. Trumpai analizuojamas AES algoritmas, jo pakeitimo S lentelės, tiesinė ir diferencialinė kriptoanalizė bei aprašomas naujas nuo rakto priklausomų atsitiktinių S lentelių ir inversinių S lentelių generavimo algoritmas. Pasiūlytas S lentelių elementų nepriklausomumo patikrinimo matas bei tiriama sugeneruotų S lentelių kokybė.