IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 3, Ver. IV (May - June 2017), PP 21-26
www.iosrjournals.org

The New Approach of AES Key Schedule for Lightweight Block Ciphers

Meltem Kurt Pehlivanoğlu[1], M. Tolga Sakalli[2], Nevcihan Duru[1],
Fatma Büyüksaraçoğlu Sakalli[2]
[1](Computer Engineering Department, Kocaeli University, Turkey)
[2](Computer Engineering Department, Trakya University, Turkey)

Abstract : *This paper considers block ciphers and key schedule algorithm that is one of the crucial components of a block cipher. It computes round keys/subkeys for relevant round from a short key. The presented experiments show that proposed key schedule algorithm which inspired by Advanced Encryption Standard's (AES) key schedule has desirable properties: Avalanche Effect and Strict Avalanche Criterion (SAC). It satisfies good bit confusion and diffusion. The average success rate of the proposed key schedule algorithm for the SAC test is 95%. As a side result it was found that while testing SAC effect computed values that lie between confidence lower and upper bounds, greater than upper bounds and less than lower bound all of them reach normal distribution. Also based on example given experimental result, proposed structure exhibits a very strong Avalanche Effect because almost at the first round approximately half the bits are changed in the key.*
Keywords: *AES, Avalanche Effect, Key Expansion Algorithm, Key Schedule, SAC*

I. Introduction

In 1883 Kerckhoffs emphasized six axioms of cryptography in [1, 2] these are [3]; indecipherable system, not trust to secret algorithms and secret parameters (that axiom known as Kerckhoffs' Principle), effectual key transmission, practicable to telecommunication technology, lightweight, simple and user friendly. Kerckhoffs' Principle means that trust only secrecy of the key underlie most important design principle of the Shannon's model [4]. In this paper we focus on the second axiom because we present new key schedule algorithm whose preliminary version given in [5]. A block cipher is composed of three deterministic algorithms which are encryption, decryption and key schedule. Encryption algorithm computes ciphertext using plaintext and user-selected key is sent to key schedule algorithm to compute round keys. Decryption algorithm computes plaintext by inverting encryption function.

This is an important question: what are the requirements of a good key schedule design? There are three extremely important components: to ensure non-linearity, has Avalanche Effect property, which means a small change in the key or plaintext should increase change dramatically in the ciphertext, and the third is has SAC property, which means changing a single bit (only one-bit at a time) in the key or plaintext should change any bit in the ciphertext with probability 1/2. A weakness in the key schedule can easily break down a cipher which has a good design or make the cipher vulnerable to attacks related especially keys. Two design principles suggested by Shannon for ciphers are diffusion and confusion. Using these principles linear relation between plaintext and ciphertext is broken down. [6] shows that ciphers with well-designed, complex key schedules resist attacks better than poorly or simple designed key schedules. To reach desired security level, the hope is that good confusion, diffusion and providing strength against linear and differential cryptanalysis, on designing complex key schedules some ciphers uses pseudo-random number generators (PRNGs) with master key. Blowfish [7], RC5 [8], RC6 [9] and KHAZAD [10] ciphers' key schedules are such examples. [3] introduces some general strategies to construct a key schedule as follows: linear key schedule with linear operations like bit permutations, extractions (DES [11], IDEA [12], Skipjack [13]). In addition to previous strategy, mask with fixed constants (SAFER [14], Square [15], SIMON [16]). Add non-linear component to the previous two strategies (AES [17], PRESENT [18]). The last strategy is use different block or stream cipher to encrypt master key, finally encrypted master key is used as subkey (Camellia [19], NOEKEON [20]). In this paper we introduce a key schedule, including some improvements on the key schedule of AES and aim to be useful for other block ciphers. As a side result it was found that this key schedule algorithm ensures Avalanche Effect and SAC properties contrary to many key schedule algorithms.

This paper is organized as follows. In Section 2 the key schedule of AES briefly given and also previous studies in the literature which conclude modifications of this schedule presented. In Section 3 improved key schedule algorithm is introduced. In Section 4 we present the experimental results that measure non-linearity of the given key schedule in previous section. In the last Section 5 described extracted conclusion and future research items.

II. The Key Schedule of AES

We focus on the AES key schedule (expansion) algorithm that takes as input k word key and it produces k·(r+1) words linear array (r is round of the cipher). Expanded key provide k word key for the initial of AddRoundKey stage for each r round. Fig. 1 describes key schedule pseudocode for the 16-bytes (128-bit, or four words (a word is a 32-bit)).

```
KeyExpansion(byte key[16], word w[44])
{
       word temp;
       for (i = 0; i < 4; i++)
  w[i] = (key[4*i], key[4*i+1],key[4*i+3], key[4*i+3]);

   for (i = 4; i < 44; i++){
              temp = w[i-1];
              if ( i mod 4 = 0 )
       temp = SubWord(RotWord(temp)) ⊕ Rcon[i/4];
              w[i] = w[i-4] ⊕ temp;
                           }
}
```

Fig. 1. Key expansion pseudocode for the 16-bytes key input [21].

Fig. 1 shows, the first k words of expanded key is 16 byte-key array taken as input. Remainder bytes of the expanded key are filled with k words at a time. RotWord function circular left shift one byte on a word. SubWord function is also substitution on each byte of input word by using AES S-box. Rcon is round constant arrays indicating related round. Detailed structure was described in [21]. In the literature previous studies on AES key schedule reviewed as follows. In 2000, Ferguson et al. [22] approached to partial key guessing and key splitting properties. According to authors' results using partial key guessing property on AES key schedule attacker can be learn amount of key bytes, which will not be neglected, by guessing some bytes. With key splitting property guessing bottommost rows leaves the cipher vulnerable meet-in-the-middle attack. In [23] the authors made some experiments on the original key schedule of the AES to measure bit confusion by using frequency test and bit diffusion by using SAC test. However results were pointing that AES key schedule has bit leakage problem: majority of subkeys did not reach complete bit diffusion and all of them failed SAC test. They also proposed new AES key schedule which has much better performance than original AES. However in [24] authors found equivalent keys, which produce the same encryption result, given in [23] key schedule structure. They also designed two new AES schedule algorithms: one of them a new approach of [23] that eliminates the equivalent keys, the other is a new on-the fly key schedule for AES which executes with at that time encryption process. In 2009, for AES-256 chosen-key and related-key attacks are given in [25]. The authors analyzed that slow diffusion that causes related-key attack. Moreover, in the same year, in [26] were introduced related-key attacks for all keys of AES-192 and AES-256. The first key recovery attacks on the full keys AES-128, AES-192, AES-256 given in [27]. In 2011, Nikolić presented a new and improved AES called xAES [28]. The author presented a few approaches to raise resistant of AES key schedule against related-key attack these are increasing the number of rounds, designing key schedule full of S-box, the last one is while keeping the number of rounds constant alter the number of S-box or some operations like ANDs, XORs (exclusive-or), rotations, ORs. Author also proved that xAES is resistant against related-key attack. Huang and Lai proposed new AES key schedule to increase security level, the authors also pointed out new weakness of the AES key schedule [29].

III. A New Key Schedule Proposal

We propose a new key schedule algorithm having some major modification on AES key schedule. We aim to use it security of extremely constrained environments such as sensor networks, RFID tags etc., especially for lightweight ciphers. For all these reasons, initially input bits are reduced as 64-bit for 10 rounds to ensure more compact. Moreover the properties of many lightweight block ciphers in the literature were given in [30]. In the round function using word-wise operation helps attacker to get easily some free bytes therefore in [29] proposed to use byte-wise operation. In this schedule we use byte-wise operation. Also to avoid side-channel attack asymmetry such as round constant is necessary for that reason we used not only different constants named RCon representing as $r_0 r_1 \ldots r_7$ which randomly generated for each round, but also added RCon_R that the reverse of the RCon constant representing as $r_7 r_6 \ldots r_0$. Proposed key schedule can take keys of 64-bits as already mentioned before. An algorithmic description code of the key schedule is given in the Fig. 2.

```
// newk[i] is the i-th byte of newkey
// r is the round number r = 1, 2, …, 10

       newkey = Master Key ⊕  Rcon[r];
       for (i = 0; i < 8; i++)
       k[i] = newk [i];
       for (j = 8; j < 32; j++){
              temp = k[j-1];
              if ( j mod 8 = 0 )
              temp = SubFunc(MixFunc(SubFunc(temp)));
              k[j] = k[j-8] ⊕ temp;
              l = j;                    }
       sub temp = (k[l-7], k[l-6], k[l-5], k[l-4],
k[l-3], k[l-2], k[l-1], k[l]);
       sub key = newkey ⊕ RconR[r] ⊕ sub temp;

return sub key
```

Fig. 2. Key expansion pseudocode for the New Key Schedule 64-bit (8-bytes) key input.

The user-supplied key is in other word master-key is stored in a key array as *Master Key*, *r* is the number of rounds and *Rcon* is the round constant array that takes different values every indices *r*. According to key schedule *Master Key* is XORed with *Rcon* value which is stored as *newkey* array representing as $k_0k_1…k_7$. Each bytes *k* [i] depends on *k* [i-1] and the eight position back *k* [i-8]; they XORed each other. For the byte *k* [i] whose position in the *k* array is multiple of eight, complex function that substitution-diffusion-substitution design architecture used. Fig. 3 illustrates that complex function have which sub-functions. The complex function consists of the following a pair of sub-functions: *SubFunc* performs four-bit substitution on each four-bit. In this function we used PRESENT cipher's S-Box [18] because of 4-bit S-box more compact than 8-bit AES S-Box. It is not only well-suited to the hardware implementation but also it is resistant differential and linear attacks because of the conditions given in [18]. The other function is *MixFunc* that used for bit permutation to provide diffusion via multiplication with constant matrix A given in (1). This matrix is MDS (Maximum Distance Separable) matrix that ensures perfect diffusion and maximum branch number [31]. Multiplication with matrix elements is in GF (2^4) using the irreducible polynomial x^4+x+1.

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} \tag{1}$$

temp (t_i) value is the result of these sub-functions. *SubFunc* step is totally applied twice.

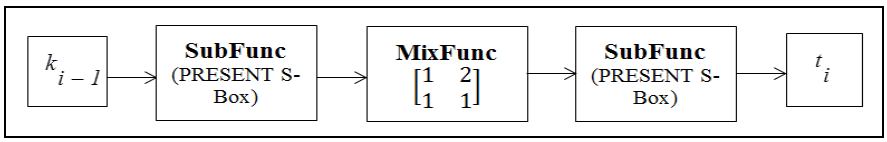

Fig. 3. Sub-functions of the complex function i=8k (for i=1 to 3).

Figure 4 illustrates the generation of the expanded subkey using the complex function represented in the Fig. 3. Considering of the solution two weaknesses of the AES slow diffusion and bit leakage, two different processes are added to the key schedule algorithm. To overcome slow diffusion weakness while getting temporary t_i values, added a diffusion element *MixFunc* which is given Fig. 3. To avoid bit leakage problem $RCon_R$ constant and *newkey* array are added to the process of the key schedule that can be seen Fig. 4.

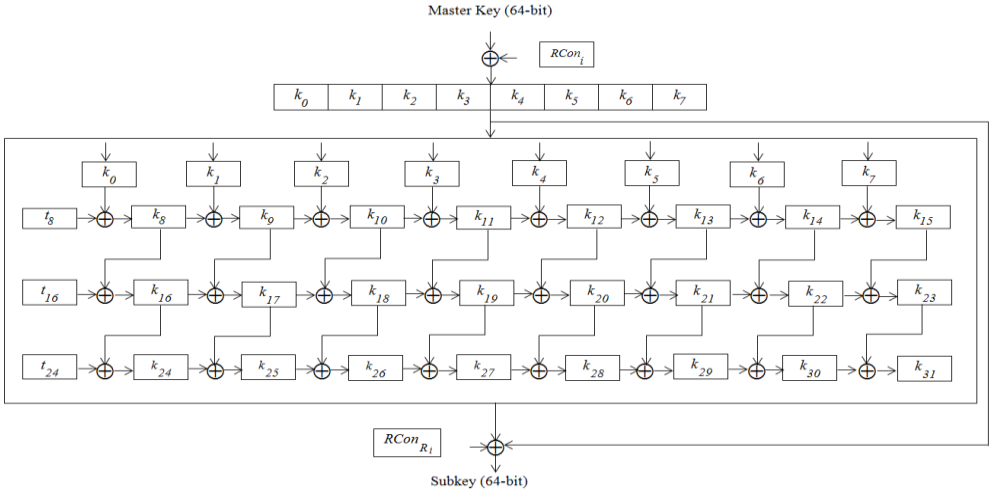

Fig. 4. Proposed key schedule for single round

IV. Experimental Results

To measure diffusion properties of the proposed new key schedule we use one of the basic statistical tests SAC which was performed to measure of bit diffusion that is the Shannon's diffusion property. SAC algorithm's pseudo code is given Fig. 5.

```
// counter [Master Key_block_length] [round_number] [round key_block_length]
and the first value of this array is set to 0
// r is the round number
// test_number is the test number to measure the SAC value

        for (t = 1; t <= test_number; t++){
        Master Key = round(); // generate random 64 bit Master Key
        for (i = 1; i <= 64; i++){
        K1 = Change_bit(Master Key,i);
        SubKey_Master = New_Key_Schedule(Master Key);
        SubKey_K1 = New_Key_Schedule(K1);
        for (r = 1; r <= 10; r++){
             for (j = 1; j <= 64; j++){
             if SubKey_Master [j] != SubKey_K1 [j]
              counter [i] [r] [j]++ ;
                                        }
                                    }
                                   }
                                            }
return counter
```

Fig. 5. SAC algorithm pseudo code that organized according to the proposed key schedule algorithm. (64-bit and 10 rounds)

According to Fig. 5 firstly *test_number* value is chosen and new master key values that are produced as many as *test_number*, called *Master Key* which takes different values each time. For each 64-bit of the related *Master Key*, one bit changed using the *Change_bit* function and the result is stored as *K1*. *SubKey_Master* is obtained using proposed key schedule function *New_Key_Schedule* which takes original *Master Key* as parameter. On the other hand *SubKey_K1* is produced from the K1 using the same *New_Key_Schedule* function. For each 10 rounds, each of the *SubKey_Master* indices and *SubKey_K1* indices are compared then for the corresponding indices; if the values are different *counter* [i] [j] [r] value increased.

Depending on *test_number* value, minimum and maximum confidence boundaries are calculated as (2).

$$\text{confidence bounds} \approx test\_number/2 \pm \sqrt{test\_number} \tag{2}$$

All values of the *counter* [i] [j] [r] are between 0 and *test_number*, on average these values approach to half of the *test*_number. Furthermore, if the value of the *counter* [i] [j] [r] is not among confidence bounds, this value will interpreted as deviation from the randomness.

In this paper we give not only SAC test of the proposed key schedule algorithm for the different *test_number* values but also given Avalanche Effect in the algorithm changing one bit key for 10 rounds.

Table 1. SAC test of the different Master Keys for the *test_number* values

test_number (with Different Master Keys)	*Confidence Bounds (X: lower bound, Y:upper bound) (X ≤ value ≤ Y)*	*Count of values between X and Y*	*Count of values greater than Y*	*Count of values less than X*	*Detailed information about the values less than X*			
					for Minimum Value		*for Maximum value*	
					Value	*Count*	*Value*	*Count*
25	7.5≤value≤17.5	38964	990	1006	3	6	7	665
50	18≤value≤32	39548	682	730	10	4	17	397
100	40≤value≤60	39502	659	799	30	8	39	310
250	109≤value≤141	39295	842	823	92	5	108	246
500	228≤value≤272	39265	780	915	201	1	227	149
750	348≤value≤402	39196	822	942	318	3	347	153
1000	468≤value≤532	39169	887	904	440	4	467	136
1500	711≤value≤789	39239	886	835	677	1	710	128
2000	955≤value≤1045	39133	878	949	917	1	954	104
3000	1445≤value≤1555	39254	814	892	1388	1	1444	96
5000	2429≤value≤2571	39169	900	891	2332	1	2428	41
10000	4900≤value≤5100	38929	895	1136	4771	1	4899	29

To measure the SAC property for the proposed key schedule algorithm used different *test_number* values given Table 1. For each *test_number* values, different master keys are randomly generated and using SAC function *counter* variables are calculated. Additionally some analysis were presented related to how many of the values lies between confidence lower and upper bounds, how many of them greater than upper bound or less than lower bound. Moreover, more detailed information about deviation values from the confidence bounds

that are less than lower bound given in the same table. For example for the *test_number* 25, 1006 of them are less than 7.5 and 6 of them is the minimum value that is 3. Similarly the maximum value that is computed between these 1006 values is 7 and there is counted 665 maximum values. Count distribution of the residual values (except values between confidence bounds and greater than upper bound) is given Fig. 6.

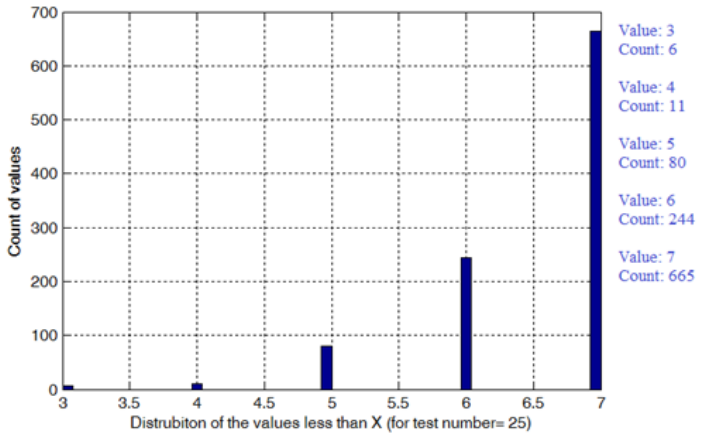

Fig. 6. Count distribution of the values that are less than lower bound for the *test_number* 25

For the all *test_number counter* variable takes 40960 (= 64·10·64) values and for the *test_number* 25, 38964 of them are between confidence bounds namely 95.13 % of them pass the SAC test. It is clear that the number of the bits passing from the SAC test almost is the same for all *test_number*. However according to previous studies, while difference between lower and upper confidence bounds decreasing, passing values from the SAC test reduce at the same time. This results show the proposed algorithm obtained a good result for the SAC test. For the *test_number* 10000 it just decreases to 95.04 %. In general it can be said that the success of the algorithm approximately is %95 for the SAC test. Additionally for the new key schedule algorithm, Avalanche Effect test results are given in Table 2.

Table 2. Avalanche Effect test in the '0001020304050607' Master Key (for changing of i^{th} bit and i=1, 2.., 10)

A Bit Changed Master Key (Changed Bit Given Bold)	***Number of Bits that Differ for the Related Round***										***Average Number of Bits that differ at the end of 10 rounds***
	1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	
8001020304050607	35	41	43	39	41	35	35	25	43	31	36,8
4001020304050607	31	33	35	29	37	37	27	25	41	49	34,4
2001020304050607	37	25	21	25	29	25	31	41	29	35	29,8
1001020304050607	35	33	27	23	43	47	35	19	27	43	33,2
0**8**01020304050607	23	19	21	45	31	33	43	19	47	27	30,8
0**4**01020304050607	35	23	35	31	21	27	17	39	31	39	29,8
0**2**01020304050607	35	37	29	29	33	45	37	23	39	21	32,8
0**1**01020304050607	41	35	49	47	41	29	33	35	33	37	38
00**8**1020304050607	43	23	43	31	31	37	35	41	35	39	35,8
00**4**1020304050607	33	39	35	27	29	35	27	31	41	23	32

Table 2 shows the change result when the i^{th} bit of the "0001020304050607" master key is changed (i=1, 2.., 10) to measure avalanche effect. First column of the table shows the value of the changed master key. The second column is divided to ten sub columns which are represent to round numbers. These sub columns indicate the number of bits that differ for the related round when used the two keys differ in the i^{th} bit. Finally, the third column shows the average changing amount of key bits that differ at the end of 10 rounds. Using Matlab we also changed each 64 bit of the original master key (0001020304050607) then calculated the change for two keys which are original master key and changed key. Nevertheless due to the size limit of the table, in the Table 2 only 10 of them given. A bit difference in approximately half the positions in the most desirable outcome [21] hence based on this example it is clear that proposed algorithm exhibits very strong Avalanche Effect from the almost first round. AES key expansion requires three rounds to reach the point which approximately half of the bits are changed. DES is worse than AES [21].

V. Conclusion

There are still open questions about the designing criteria of the key schedules for the block ciphers. In this paper we have designed and presented a new key schedule algorithm which brings new approaches to AES key schedule and contrary to AES it has good bit diffusion. To measure bit diffusion property Avalanche Effect

and SAC criteria have used and according to results subkeys pass the SAC test and attain complete bit diffusion. Moreover usage of round constants breaks the symmetry. In the future, also to measure bit mixing property frequency test will be used.

References

[1] K. Auguste, *La cryptographie militaire I-III* (Military cryptography, Parts I-III, Journal des Sciences Militaires, IX:5–38,1883).
[2] K. Auguste, *La cryptographie militaire IV* (Military cryptography – Part IV, Journal des Sciences Militaires, IX:161–191, 1833).
[3] R. Avanzi. 2016, *A salad of block ciphers-the state of the art in block ciphers and their analysis* (http://eprint.iacr.org/2016/1171.pdf, 2016).
[4] C.E. Shannon, *Communication theory of secrecy systems* (Bell System Technical Journal, 28:656–715, 1949).
[5] F. Büyüksaraçoğlu Sakallı, E. Buluş, T. Sakallı, H. Vural, AES blok şifresinin anahtar genişletme rutininin geliştirilmesi ve bir blok şifreden bağımsız anahtar genişletme rutininin tasarımı, *6. Uluslararası Bilgi Güvenliği ve Kriptoloji Konferansı*, Ankara, Türkiye, 2013, 347-353
[6] L. R. Knudsen, J. E. Mathiassen, On the role of key schedules in attacks on iterated ciphers, *ESORICS 2004*, French Riviera, France, 2004, 322-334
[7] B. Schneier, Description of a new variable-length key, 64-bit block cipher (blowfish), *FSE 1993*, Cambridge, U. K., 191–204.
[8] R. L. Rivest, The RC5 encryption algorithm, *FSE 1994*, 1994, Leuven, Belgium, 1994, 86–96.
[9] R. L. Rivest, M. Robshaw, R. Sidney and Y. L. Yin. RC6 as the AES, *In Proceedings of the Third AES Candidate Conference*, New York, USA, 2000, 337–342
[10] A. Biryukov, Analysis of involutional ciphers: Khazad and Anubis, *FSE 2003*, Lund, Sweden, 2003, 45–53.
[11] Federal Information Processing Standards, *Data encryption standard* (Publication No. 46, National Bureau of Standards, January 15, 1977).
[12] X. Lai and J. L. Massey, A proposal for a new block encryption standard, *EUROCRYPT 1990*, Denmark, 1990, 389–404.
[13] National Institute of Standards and Technology, *SKIPJACK and KEA algorithm specifications*, Technical report, 1998.
[14] J. L. Massey, SAFER K-64: a byte-oriented block-ciphering algorithm, *FSE 1993*, Cambridge, U. K., 1–17.
[15] J. Daemen, L. R. Knudsen, and V. Rijmen, The block cipher square, *FSE 1997*, Haifa, Israel, 149–165.
[16] R. Beaulieu, D. Shors, J. Smith, S. Treatman Clark, B. Weeks, and L. Wingers, The SIMON and SPECK families of lightweight block ciphers, *Cryptology ePrint Archive*, Report 2013/404, 2013.
[17] J. Daemen and V. Rijmen, *The design of rijndael: AES - the advanced encryption standard information security and cryptography* (Springer, 2002).
[18] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, PRESENT: an ultra-lightweight block cipher, *CHES 2007*, Vienna, Austria, 2007, 450–466.
[19] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, Camellia: a 128-bit block cipher suitable for multiple platforms - design and analysis, *SAC 2000*, Ontario, Canada, 2000, 39–56.
[20] J. Daemen, M. Peeters, G. V. Assche, and V. Rijmen, Nessie proposal: NOEKEON, *First Open NESSIE Workshop*, 2000, 213-230.
[21] W. Stalling, Cryptography and Network Security: Principle and Practice (Sixth Edition, 2014).
[22] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, Improved cryptanalysis of Rijndael, *FSE 2000*, New York, USA, 2000, 213-230.
[23] L. May, M. Henricksen, W. Millan, G. Carter, E. Dawson, Strengthening the key schedule of the AES, *ACISP 2002*, Melbourne, Australia, 2002, 226-240.
[24] J. Choy, A. Zhang, K. Khoo, M. Henricksen, and A. Poschmann, AES variants secure against related-key differential and boomerang attacks, *Information Security Theory and Practice: Security and Privacy of Mobile Devices in Wireless Communication*, 6633, 2011, 191-207.
[25] A. Biryukov, D. Khovratovich, and I. Nikolić, Distinguisher and related-key attack on the full AES-256, *CRYPTO 2009*, Santa Barbara, USA, 2009, 231–249.
[26] A. Biryukov and D. Khovratovich, Related-key cryptanalysis of the full AES-192 and AES- 256, *ASIACRYPT 2009*, Tokyo, Japan, 2009, 1–18.
[27] A. Bogdanov, D. Khovratovich, and C. Rechberger, Biclique cryptanalysis of the full AES, *ASIACRYPT 2011*, Seoul, South Korea, 2011, 334-371.
[28] I. Nikolić, Tweaking AES, *SAC 2010*, Ontario, Canada, 2010, 198-210.
[29] J. Huang, X. Lai, Transposition of AES Key Schedule, *International Conference on Information Security and Cryptology (Inscrypt) 2016*, Beijing, China, 2016, 84-102.
[30] Cryptolux, *Lightweight block ciphers* (https://www.cryptolux.org/index.php/Lightweight_Block_Ciphers).
[31] H. Xu, L. Tan, X. Lai, On the recursive construction of MDS matrices for lightweight cryptography, *ISPEC 2014*, Fuzhou, China, 2014, 552-563.