DE GRUYTER
OPEN

TATRA
MOUNTAINS
Mathematical Publications
DOI: 10.2478/tmmp-2014-0023
Tatra Mt. Math. Publ. **60** (2014), 25–45

A METHOD FOR SECURITY ESTIMATION OF THE SPN-BASED BLOCK CIPHER AGAINST RELATED-KEY ATTACKS

Dmytro Kaidalov — Roman Oliynykov — Oleksandr Kazymyrov

ABSTRACT. Symmetric block ciphers are the most widely used cryptographic primitives. In addition to providing privacy, block ciphers are used as basic components in the construction of hash functions, message authentication codes, pseudorandom number generators, as a part of various cryptographic protocols, etc. Nowadays the most popular block cipher is AES (Advanced Encryption Standard). It is used as a standard of symmetric encryption in many countries. Several years ago it was found a theoretical attack exploiting the AES key expansion algorithm that allows reducing significantly the complexity comparing to the brute force attack. This article presents an advanced method of finding the number of active substitutions that helps to estimate the security of encryption algorithms against related-key attacks. The method was applied to a prospective block cipher, which is a candidate for the Ukrainian standard.

1. Introduction

In [5]–[7] it is stated that related-key attacks found for AES-192 and AES-256 significantly reduce the theoretical security level of this algorithm. The natural question how to design an SPN-based block cipher resistant to related-key attacks is arisen.

In this article an improved SPN-like cipher based on a prospective algorithm Kalyna [4], [17] is considered. The new cipher has significant improvements in comparison with AES. These include a new key expansion scheme, higher resistance level against algebraic attacks, higher performance, etc. The original encryption algorithm was proposed to the public competition of block cipher selection to be a prototype during the development of Ukrainian national standard [17].

2010 Mathematics Subject Classification: 94A60, 68P25.
Keywords: block cipher, differential cryptanalysis, related-key attack.

To achieve a high level of security the proof against related-key attacks must be conducted. This is a challenging problem since existing methods have high complexities when they are applied to the modern ciphers. Therefore, the main goal of the article is the development of a more efficient method of finding the number of active substitutions that helps to estimate the security of encryption algorithms against related-key attacks. The article contains some necessary parts of the developed algorithm, the estimation of complexities and the application of the proposed method to the described SPN-based cipher.

2. Related-key attacks on modern block ciphers

2.1. Description of the related-key attack on AES

The following brief description of the relate-key attack is taken from [5]. The core of the attack is differential cryptanalysis [16]. The idea of this attack is to inject a difference into the internal state, causing a disturbance, and then to correct it with the next injections. The resulting difference pattern is spread out due to the message schedule causing more disturbances in other rounds. The goal is to have as few disturbances as possible in order to reduce the complexity of the attack [5].

In the related-key scenario it is allowed to inject difference into the key, and not only into the plaintext as in pure differential cryptanalysis. To use the attack the information about the encryption key is unknown, but an attacker can control the difference of key pairs.

Local collisions in AES-256 are best understood on a one-round example (Figure 1).

Here one active S-box is needed and five non-zero byte differences in the two subkeys. These five bytes split into two parts: one-byte disturbance and four-byte correction.

Due to the key schedule the differences spread to other rounds. Most of the AES key schedule operations are linear, so a sequence of several consecutive round key values can be viewed as a codeword of a linear code. This is a particularly case when a trail does not have active S-boxes in the key schedule.

Let figure out how to build an optimal trail for the key recovery attack. Typically, a trail is better if it has fewer active S-boxes. Disturbance differences must form a codeword that has a low weight. Simultaneously, correction differences must also form a codeword that sums into the key schedule codeword. In further trails, the correction codeword is constructed from the former one by just shifting four columns to the right and applying S-box and the MixColumns operations. Synchronization is simple since the injection is made to the first row, which is not rotated by ShiftRows. Otherwise, the task of synchronizing two

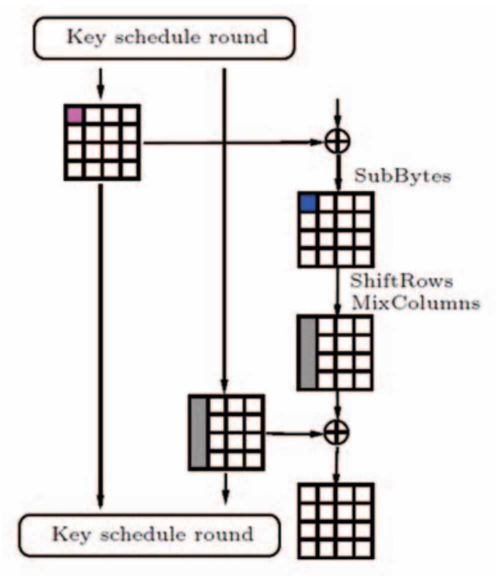

FIGURE 1. Local collisions in AES [5].

codewords would have been much harder and would have led to high-weight codewords [5].

An example of a good key schedule pattern for AES-256 is depicted in Figure 2 as a 4.5-round codeword.

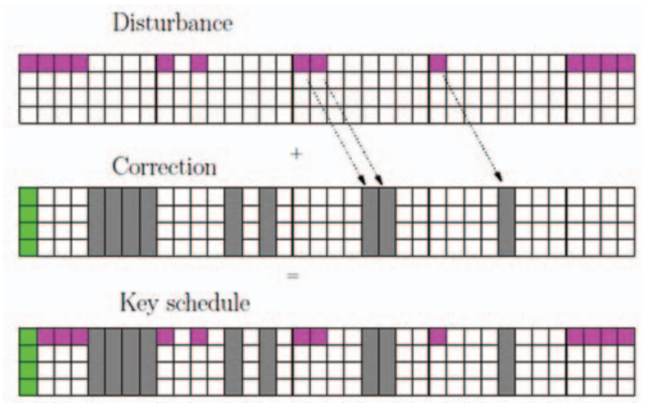

FIGURE 2. AES-256 key schedule codeword (4.5 key-schedule rounds) [5].

In the first four key-schedule rounds the disturbance codeword has only 9 active bytes, which is the lower bound. It is needed to avoid active S-boxes in the key schedule as long as possible. Due to a weak diffusion in the AES key schedule the difference affects only one more byte per key schedule round. The correction column should be positioned four columns to the right, and propagates backwards in the same way. The last column in the first round key is active, so all S-boxes of the first round are active as well, which causes unknown difference in the first column. This "alien" difference should be canceled by the plaintext [5].

So such collisions can considerably reduce the complexity of key-recovery attack.

2.2. Existing methods to search related-key differential characteristics

The first automatic search for the best differential characteristic of DES was performed by Matsui [12]. Further, algorithms for automatic search of differential characteristic were proposed for MD4, MD5, and SHA-1 [13]–[15].

The first automatic tool for finding related-key differential characteristics was presented by Alex Biryukov and Ivica Nikolic in [11]. They described an algorithm that allows to compute differential characteristic for AES, Camellia, Khazad and other cryptographic primitives. The general idea of our searching algorithm is close to the one in [11]. Unlike the algorithm of Biryukov and Nikolic that controls each byte of a state, in the proposed algorithm the entire column is controlled for the amount of active bytes without the exact position of these bytes. It is become possible because of the structure of researched encryption algorithm. Such scheme decreases the complexity of computation and reduces influence of high branching mentioned in [11].

3. A prospective SPN-based block cipher

This section describes some important parts of a prospective SPN-based cipher in details that is necessary for our method. This cipher is based on AES. The main differences compared to AES are the follows: pre- and post-whitening use modulo addition 2^{64}, expanded size of MDS matrix to 8×8, application of several S-boxes optimized with respect to differential, linear and algebraic cryptanalysis, and considerably redesigned key expansion routine.

3.1. General parameters

The encryption algorithm can be used with different types of input data. It supports length blocks of 128, 256 or 512 bits. The length of the key can be also 128, 256 or 512 bits [4], [17]. The internal state of the cipher can be represented as a matrix of bytes. The matrix dimension is 8^*N_b bytes. In Table 1 acceptable combinations of block and key sizes are represented.

Table 1. Acceptable combinations of block and key sizes.

Size of block, bits	Supported key size, bits
128 ($N_b = 2$)	128, 256
256 ($N_b = 4$)	256, 512
512 ($N_b = 8$)	512

Number of rounds depends on the size of block and key that are presented in Table 2.

In this paper the 128-bit version of the cipher is considered. Presented information in the next chapters is concerned only about this version if another is not mentioned.

Table 2. The number rounds for different versions of the cipher.

Size of block, bits	Size of key 128 bits	Size of key 256 bits	Size of key 512 bits
128 ($N_b = 2$)	10	14	–
256 ($N_b = 4$)	–	14	18
512 ($N_b = 8$)	–	–	18

3.2. Basic transformations

In the presented algorithm four basic transformations are used:

- key addition,
- bytes substitution,
- shift rows,
- mix columns.

These transformations are the basis of entire high-level structure [4], [17].

3.3. Key expansion scheme

Let K_M be the main key of encryption and $(K_1, K_2, \ldots, K_m)$ are the round keys that are generated by the key expansion scheme.

The cipher is based on an SP network similar to the AES cipher

$$Cipher_{K_M} = \prod_{i=1}^{N_r} \theta \circ \gamma \circ \sigma_{k_i},$$

where

σ_{k_i} – round key adding,

γ – nonlinear layer (bytes substitution),

θ – linear layer (mix columns, shift rows),

N_r – amount of rounds in block cipher.

The key expansion scheme contains the following two steps:

1. Computing of an intermediate value K_t based on a master key K_M and constants.
2. Computing of round keys $(K_1, K_2, \ldots, K_m)$ based on a master key K_M, an intermediate value K_t and constants.

The computation of intermediate value K_t is depicted in Figure 3. The algorithm contains all those transformations that are used in the encryption cipher. This reduces implementation complexities because all basic operations are the same.

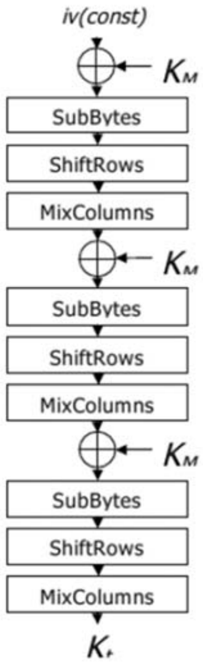

FIGURE 3. Computing of K_t.

The intermediate value can be computed by the following formulas:

$$IM_{K_M} = \prod_{i=1}^{N_r} \theta \circ \gamma \circ \sigma_{k_i},$$

$$K_t = IM_{K_M}(i\nu),$$

where $i\nu$ is a constant which reduces symmetry in the round keys.

Generation of round keys can be formalized with next expression:

$$RK_{K_t}[K_M] = \sigma_{K_t+tm\nu_0} \circ \prod_{i=1}^{2} \theta \circ \gamma \circ \sigma_{K_t+tm\nu_i},$$

where $tm\nu_i$ are constants. For each round this value must be unique and must have the low computational complexity (e.g., a simple shift by a few positions).

General algorithm for the round keys generation is represented in Figure 4 [4], [17].

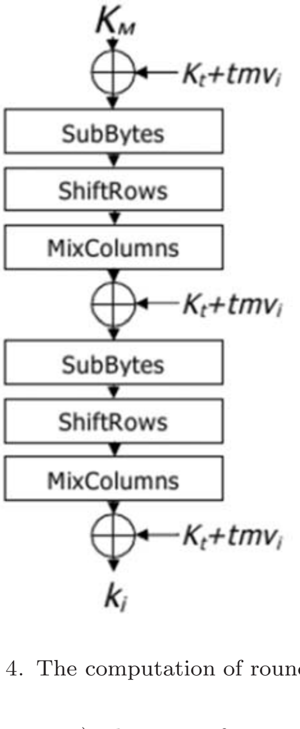

FIGURE 4. The computation of round keys.

In the above scheme (Figure 4) the transformations similar to the scheme of computation K_t and the encryption scheme are used. This reduces the complexity of implementation. According to [10] the scheme has several properties, such as:

1. One-way mapping: having an encryption key it is very easy to generate round keys, but having one or more round keys it is computationally very difficult to retrieve the encryption key or another round key.

2. Non-linear dependence between each bit of encryption key.

3. Statistical properties of the key schedule were verified by the NIST statistical test suite.

4. Simple implementation (based on the cipher round transformations only), good key agility and possibility to generate round keys in direct and reverse order with the same computational complexity.

3.4. The encryption scheme

The encryption transformation is almost the same as in the AES algorithm. A general scheme for 128-bit version of the cipher is represented in Figure 5. Addition of keys K_0 and K_{10} is performed by modulo 2^{64}. In other cases the XOR operation is used [4], [17].

It is easily to see that the cipher is based on a classic SP network. The same structure is used in the key expansion algorithm for simplification of the implementation.

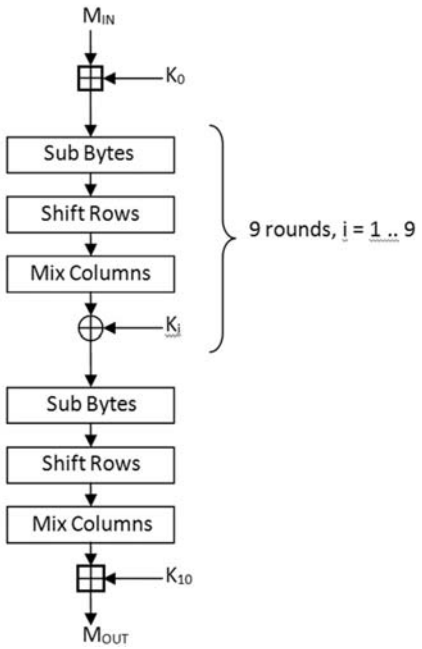

FIGURE 5. A high-level structure of the 128-bit block cipher.

4. The algorithm for estimation of the number of active bytes for related-key attacks

4.1. The method for cipher security estimation

To prove the security of an encryption algorithm against related-key attacks it is needed to find the best differential characteristic. The best characteristic has as few active bytes as possible. So, to find the best differential characteristic the amount of active bytes, which will be used during its construction, should be counted. An active byte is a non-zero byte difference that passes through the substitution table. As it was mentioned above, a substitution of bytes is a non-linear mapping. It means that the output difference is not possible to predict with probability 1 and an attacker has to select a right pair satisfying each active substitution. If the amount of active bytes exceeds some boundary value the cipher can be considered as secure with respect to key-related attack, because the complexity becomes higher than the complexity of the brute force attack on the cipher. The boundary value depends on the size of cipher block and the size of encryption key.

The best differential characteristic can be found by searching among all possible input differences. This can be done automatically by special software. In the next sections the search in reasonable time for the 128-bit version of the cipher is shown.

4.2. The algorithm for counting active bytes

As it was mentioned before, the best differential characteristic (with the fewest amount of active bytes) can be found by searching among all possible input differences. For each characteristic the amount of active bytes after the key expansion scheme and all rounds of encryption should be counted. The proposed algorithm is considered for the 128-bit version of the cipher, but it can be extended for larger blocks as well as other ciphers.

Let us consider an example to compute a differential characteristic for the evaluation of K_t. As it was mentioned before, K_t is an intermediate key that is computed in the key expansion scheme. A general scheme for computation of a differential trail is represented in Figure 6.

The following notions were used in Figure 6:

$i\nu$	– the difference of initial vectors (always equal to zero);
K_M	– master key of encryption (which is needed to be expanded);
SB	– sub bytes (bytes substitution using substitution tables);
SR	– shift rows;
MC	– mix columns.

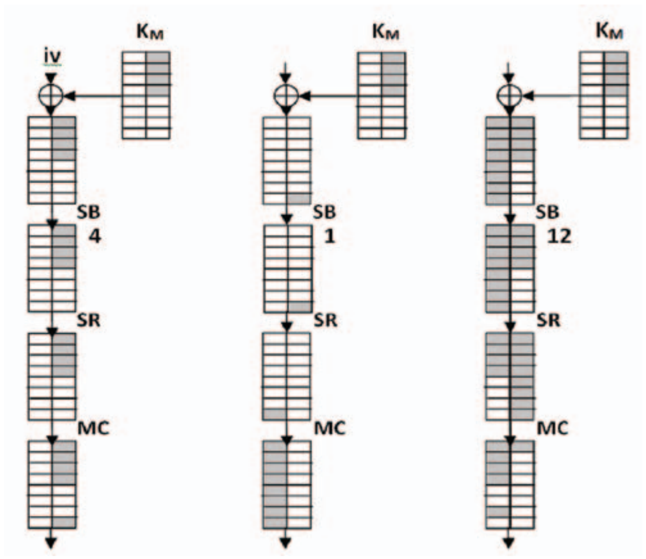

FIGURE 6. A differential trail for K_t computation.

In the example (Figure 6) three rounds (from left to right) of K_t computation are shown. Input data is a difference of the encryption key K_M. Then it is shown how this difference changes by different transformations. The amount of active bytes is the most interesting information. In the above example this value is equals to 4 for the first round, 1 for the second round and 12 for the third round. In general there are 17 active bytes.

It should be mentioned that in linear transformations the made decision was based on the best case for the cryptanalyst. For example, the difference after MC has such value that addition with the key K_M gives collision in the second round.

4.2.1. A general description

For the 128-bit version case an automatic algorithm for counting active bytes can be implemented. In the proposed method the entire column is under control instead of separate bytes. As it was mentioned in Section 3, the state of the cipher is represented as a byte matrix. Each column of the matrix consists of 8 bytes. Thus the 128-bit cipher has two columns. In the automatic searching algorithm the amount of active bytes in columns is counted without their exact positions. This algorithm considerably reduces the complexity. The accuracy of the result decreases, because the found trails may not really exist. Such trails cut down the amount of active bytes, which leads to the minimum value. If the amount of active bytes does not exceed a theoretical bound, then the cipher is resistant against related-key attacks.

Figure 7 presents an example of a differential trail for K_t computation scheme. The non-zero numbers (i.e., active bytes) are recalculated after each transformation. This example gives 3 active bytes in each round and 9 active bytes in total.

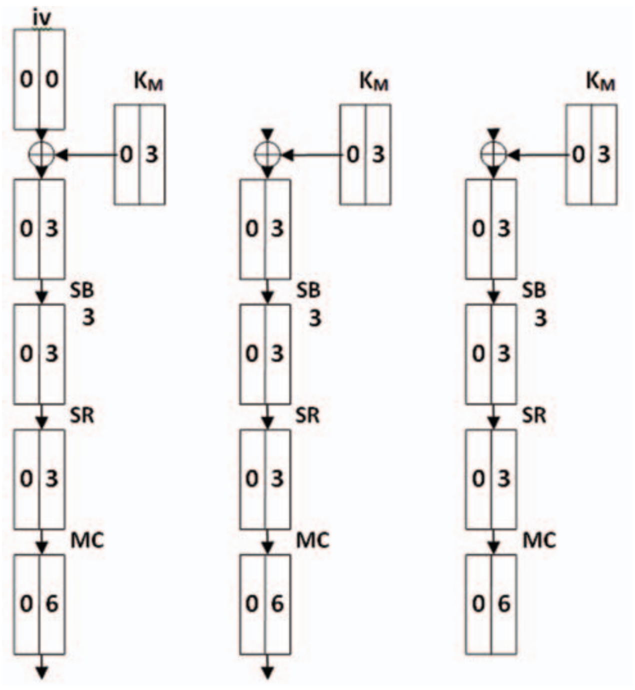

FIGURE 7. An example of the active bytes computation.

In order to make it more understandable in the next sections each transformation is described in details.

4.2.2. Key addition

The key addition operation is performed as follows: a value of a state column is subtracted from a value of the corresponding column of a key state. Then an absolute value of the result is taken. Some examples are given in Figure 8.

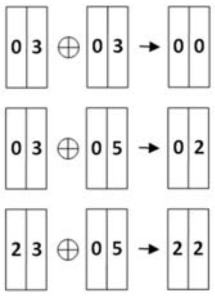

Figure 8. The calculation of active bytes after key addition.

It is assumed that non-zero bytes have such positions and values that collisions occur. It means that it is the best case for the cryptanalyst. That is why the amount of active bytes can be lower than in real situation.

4.2.3. Byte substitution

The byte substitution transformation does not change the amount of non-zero bytes. The value is always constant before and after the transformation. Nevertheless, this transformation is very important, because the security level of the cipher depends on the amount of non-zero bytes that are passed through the substitution tables. The more secure a cipher is, the more active bytes it has.

4.2.4. Shift rows

The operation of shifting bytes in 128-bit version of the cipher is depicted in Figure 9.

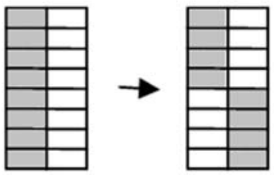

Figure 9. A general calculation of active bytes after shift rows.

The last 4 bytes of the left and right column are replaced. A challenging problem here is how many bytes should be shifted from one column to another. This problem was solved as follows: all possible variants of shifting were taking into account. It means that the differential trail can be branched. It takes a lot of computational resources, because in each round new branches appear and the amount of these independent branches is growing exponentially.

An example of the shift rows transformation with branches is given in Figure 10.

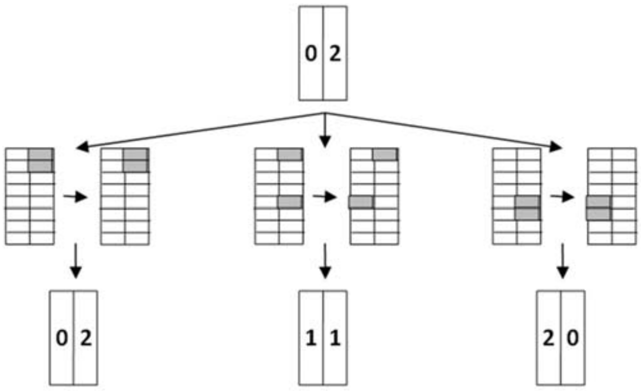

FIGURE 10. The calculation of active bytes after shift rows for the proposed algorithm.

The above example (Figure 10) shows that the state $(0; 2)$ has three independent branches. All of these three branches must be taken into account while finding the best differential trail. Table 3 represents the amount of alternatives for the shift rows operation.

TABLE 3. The amount of branches depends on the amount of non-zero bytes in a column.

Amount of non-zero bytes in a column	Amount of branches
0	1
1	2
2	3
3	4
4	5
5	4
6	3
7	2
8	1

The total amount of branches after the shift rows transformation will not exceed the product of possible shifts of the left column and the right column. For example, if the left column has 3 non-zero bytes and the right column has 4 non-zero bytes, then the amount of branches for this transformation is $4\times 5 = 20$.

4.2.5. Mix columns

Mixing in columns is performed as the product of column and a fixed matrix. Unlike the previous one this transformation does not create new branches. The main property of the mix columns transformation is the sum of non-zero bytes on the input and the output cannot be less than 9:

$$N_{in} + N_{out} \geq 9.$$

There is an exception when the column has no active bytes. In this situation both input and output are without active bytes. In the developed algorithm the best case for the cryptanalyst ($N_{in} + N_{out} = 9$) is taken. Several examples of how Mix Columns works are represented in Figure 11.

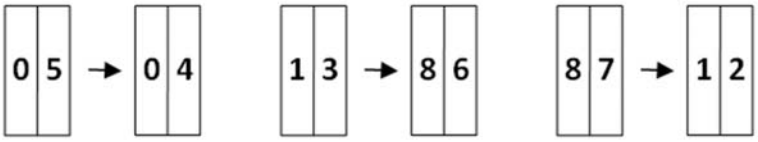

FIGURE 11. The calculation of active bytes after mix columns.

4.3. The description of results

The minimal secure threshold for estimated cipher is 26 active bytes. This value is obtained from the next equation:

$$\left(2^{-5}\right)^{x} < 2^{-128},$$

where 2^{-128} is the probability to break the cipher with the brute force attack, 2^{-5} is the maximum probability that an input difference maps to the output difference. Therefore, for 26 active bytes the entire complexity to break the cipher is 2^{-130}, which is more than the complexity of the brute force attack.

The practical experiments gave the best differential trail for the 128-bit version of the cipher. This trail has 27 active bytes. The results for each round are represented in Table 4.

TABLE 4. The description of the best differential trail.

Part of cipher	Round	Amount of active bytes	Accumulated amount of active bytes
K_t computation	1	7	7
	2	4	11
	3	2	13
Key expansion scheme	1	2	15
	2	2	17
Main encryption loop	1	1	18
	2	1	19
	3	1	20
	4	1	21
	5	1	22
	6	1	23
	7	1	24
	8	1	25
	9	1	26
	10	1	27
Total			27

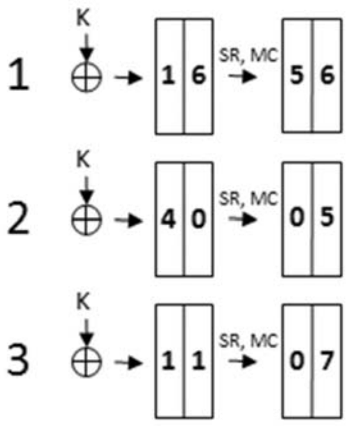

FIGURE 12. K_t computation.

Table 4 shows that the differential trail in the main encryption loop is iterative and only 1 active byte is added to the total number. The more detailed description follows.

The stage of computing K_t is given in Figure 12. This figure depicts the found differential trail for all 3 rounds. The difference of initial states is equal to $(0;0)$ and the difference of encryption keys is equal to $(1;6)$.

In Figure 13 all rounds of the key expansion scheme are represented. The key difference is equal to the output difference of the K_t computation (i.e., $(0;7)$). To receive $(1;1)$ the difference of the master key (i.e., $(1;6)$) was subtracted from the $(0;7)$ as defined in Section 4.4.2.

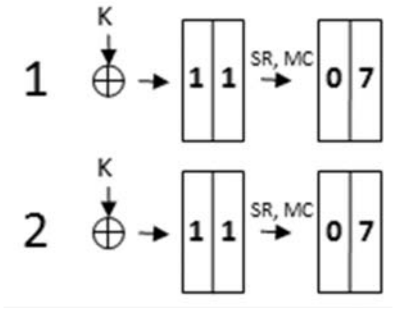

Figure 13. Round key computation.

In Figure 14 ten rounds of the main encryption loop are represented. At this stage the input state can be chosen randomly. To decrease the number of active bytes the input state $(0;6)$ was taken. According to the previous stage all round keys are equal to $(0;7)$. The final result is the differential trail with 27 active bytes. The theoretical threshold value for this version of the cipher is equal to 26 active bytes. As it was mentioned before the obtained value is the minimum amount of the active bytes that can be achieved. Practically the number of non-zero bytes in the best differential trail can be higher.

The main property of the searching algorithm is that the exact trail cannot be found, but one can prove that the encryption algorithm is resistant against related-key attacks. In particular, if the number of active bytes is greater than or equal to the threshold then it is impossible to find a differential trail that can be used in an differential attack.

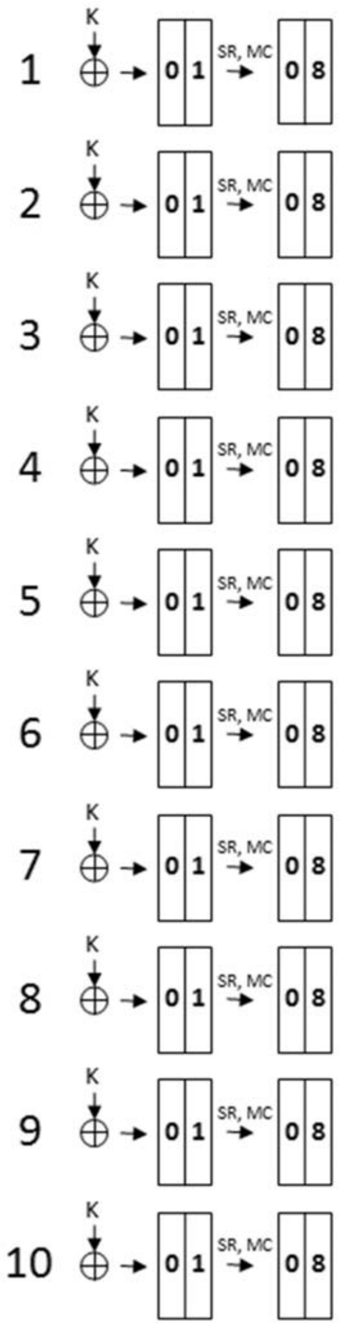

FIGURE 14. The main encryption loop.

4.4. The complexity of algorithm

The input values to the algorithm are various values of differences of master keys (it is a stage of the K_t computation). The algorithm was built in such a way that it controls 2 columns. Each column can have values between 0 and 8 (the amount of active bytes). It means that each column have 9 different variants of input values. The values of each column can be taken independently. So the total amount of input values is 81

$$N_{MK} = N_{Col}^{c} = 9^2 = 81,$$

where N_{MK} is the amount of possible combinations for the differences of master keys, N_{Col} is the number of possible states for one column, c is the number of columns in the cipher state.

With the same technique open texts can be chosen for the input to the main loop of the encryption. The number of possible combinations of input differences for open texts is equal to

$$N_{PT} = N_{Col}^{c} = 9^2 = 81.$$

As it was mentioned in Section 4.2.4, the Shift Rows transformation can create new branches. It means that additional paths appear while differential trails are searched. To find the total amount of branches after the Shift Rows transformation the multiplication of each column must be done

$$N_{DC} = \prod_{i=0}^{c} N_i,$$

N_{DC} – the amount of branches after the Shift Rows transformation;
c – the number of columns in the cipher state;
N_i – the number of branches for each column (according to Table 3).

For example, if the current state of the cipher is $(3; 4)$, then there are no more that $4 \times 5 = 20$ independent branches after Shift Rows.

The algorithm searches among all possible input text differences. On average, each column makes 3 different branches. It means that there are 9 different branches for one round. This transformation contributes most to the total complexity, because it is performed in each round. The amount of branches can be computed by the following formula

$$N_{DC} = \prod_{i=0}^{r} \prod_{j=0}^{c} N_{ij},$$

N_{DC} – the total amount of branches;
c – the number of columns in the cipher state;
r – the number rounds;
N_{ij} – the amount of branches for each column for one round (according to Table 3).

Filtering of bad characteristics reduces the complexity of the algorithm. A bad characteristic is a characteristic, which exceeds a threshold value of active bytes. So if the number of active bytes in a differential trails reach the theoretical value (or a predefined constant), then it is dropped. It considerably simplifies the computation.

The general formula to compute the complexity of the proposed algorithm is stated below:

$$O = N_{PT} N_{MK} N_{DC} = N_{Col}^{c} N_{Col}^{c} \prod_{i=0}^{r} \prod_{j=0}^{c} N_{ij} = \left(N_{Col}^{c}\right)^{2} \prod_{i=0}^{r} \prod_{j=0}^{c} N_{ij}$$

Using this formula the total complexity for the 128-bit version of the cipher can be computed. There are 15 rounds (5 rounds of the key expansion scheme and 10 rounds of the main encryption loop). The calculation for the 128-bit version of the cipher is represented below:

$$O_{128} = N_{PT} N_{MK} N_{DC} = 9^{2} 9^{2} 9^{15} = 3^{57} \approx 2^{90}.$$

The complexity impossible to reached in practice. But it can be considerably reduced by the filtering idea. The minimal value for the 128-bit version of the cipher is 27—this is minimal amount of active bytes, which was set as a threshold.

On a quad core system with an AMD Phenom 3.2 GHz processor the proposed algorithm worked for about 10 minutes to find the differential trail described in Section 4.3.

5. Conclusions

The method for security estimation of a prospective SPN-based cipher against related-key attacks is presented in this paper. The new cipher is based on the AES construction with a redesigned key schedule to provide protection against such type of attack. The cipher was proposed to the public competition of block cipher selection to be a prototype during development of the Ukrainian standard.

The calculation of the amount of active bytes is the core of the proposed method. Unlike the algorithm of Biryukov and Nikolic, which controls each byte of a state, the presented algorithm tracks the entire column without the knowledge of exact positions of these bytes. This approach decreases the complexity of computations and reduces the influence of high branching mentioned in [11].

The developed method can show the existence of differential characteristics that can lead to the key recovery attack with the complexity lower than the complexity of brute force attack. The calculation of the complexity for the proposed method is presented. This method with the direct application has a high computational complexity, which is about $O = 2^{90}$ for the 128-bit version of the cipher. The proposed optimization to reduce the complexity is based on the dynamic elimination of differential trails that reach a threshold value of active bytes. The optimized algorithm that was implemented in C++ takes about 10 minutes to find the best known differential trail for the 128-bit version of the cipher.

Thus, the proposed method gives an analytic proof of the security of block ciphers regarding related-key attacks. It was shown that the 128-bit version of the cipher does not have a differential characteristic that can lead to the key recovery attack with complexity lower than the complexity of the brute force attack.

This method can be extended to different variants of block size and key lengths of the proposed cipher as well as to other block ciphers with the similar structure.

REFERENCES

[1] SCHNEIER, B.: *Applied Cryptography. Protocols, Algorithms, and Source, Code in C* (2nd ed.), John Wiley and Sons, New York, 1994.

[2] STALLINGS, W.: *Cryptogtraphy and Network Security: Principles and Practice*. Prentice Hall, New York, 2006.

[3] (NIST), (ITL): *Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197 (FIPS PUB 197)* November, 26, 2001, `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[4] OLIYNYKOV, R. V.—GORBENKO, I. D.—DOLGOV, V. I.— RUZHENTSEV, V. I.: *Prospective symmetric block cipher: "Kalina"—basic terms and specification*, Applied Radioelectronic **6** (2007), special issue—devoted to the problems of information security, Kharkiv. (In Ukrainian)

[5] BIRYUKOV, A.—KHOVRATOVICH, D.: *Related-key cryptanalysis of the full ES-192 and AES-256*, in: Advances in Cryptology—ASIACRYPT '09 (M. Matsui, ed.), 15th Internat. Conf. on the Theory and Appl. of Cryptology and Information Security, Tokyo, Japan, 2009, Lecture Notes in Comput. Sci., Vol. 5912, Springer-Verlag, Berlin, 2009, pp. 1–18, `http://impic.org/papers/Aes-192-256.pdf/`.

[6] BIRYUKOV, A.—DUNKELMAN, O.—KELLER, N.—KHOVRATOVICH, D.—SHAMIR, A.: *Key recovery attacks of practical complexity on AES variants with up to 10 rounds*, `http://eprint.iacr.org/2009/374.pdf/`.

[7] BIRYUKOV, A.—KHOVRATOVICH, D.—NIKOLI, I.: *Distinguisher and related-key attack on the full AES-256*, University of Luxemburg, August 10, 2009, `http://www.iacr.org/archive/crypto2009/56770229/56770229.pdf/`.

[8] GORBENKO, I. D.: *Information Security in Information and Telecommunication Systems*, Textbook, Part 1, Cryptographic protection of information, Kharkiv, KNURE, 2004. (In Ukrainian)

[9] GOST 28147-89: Information processing systems, Cryptographic protection, Cryptographic transformation algorithm. (In Russian) `http://protect.gost.ru/document.aspx?control=7&id=139177/`.

[10] OLIYNYKOV, R. V.—RUZHENTSEV, V. I.: *A new approach of key schedule construction for symmetric block ciphers*, in: Proc. of the SFU, Engineering, Information Security, Taganrog, Russia, TTISFU (Taganrog Technological Institute of Southern Federal University), 2010, No. 11 (112), pp. 156–161.

[11] BIRYUKOV, A.—NIKOLI, I.: *Automatic search for related-key differential characteristic in byte-oriented block ciphers: Application to AES, Camellia, Khazad and others*, in: Adv. in Cryptology—EUROCRYPT '10, 29th Annual Internat. Conf. on the Theory and Appl. of Cryptographic Techniques, French Riviera, 2010 (H. Gilbert, ed.), Lecture Notes in Comput. Sci., Vol. 6110, Springer, Berlin, 2010, pp. 322–344, `http://link.springer.com/chapter/10.1007/978-3-642-13190-5`.

[12] MATSUI,M.: *On correlation between the order of S-boxes and the strength of DES*, in: Workshop on the Theory and Appl. of Cryptogr. Techniques—EUROCRYPT '94, Perugia, Italy, 1994 (A. D. Santis, ed.), Lecture Notes in Comput. Sci. Vol. 950, Springer-Verlag, Berlin, 1995, pp. 366–377.

[13] FOUQUE, P.-A.—LEURENT, G.—NGUYEN, P.: *Automatic search of differential path in MD4*, Cryptology ePrint Archive, Report 2007/206.

[14] STEVENS, M.: *Fast collision attack on MD5*, Cryptology ePrint Archive, Report 2006/104.

[15] CANNIERE, C. D.—RECHBERGER, C.: *Finding SHA-1 characteristics: General results and applications*, in: Advances in Cryptology—ASIACRYPT '06 (X. Lai et al., eds.), 12th Internat. Conf. on the Theory and Appl. of Cryptology and Inform. Security, Shanghai, China, 2006, Lecture Notes in Comput. Sci., Vol. 4284, Springer-Verlag, Berlin, 2006, pp. 1–20.

[16] HEYS, H. M.: *A tutorial on linear and differential cryptanalysis*, Cryptologia **26** (2002), 189–221.

[17] OLIYNYKOV, R.—GORBENKO, I.—DOLGOV, V.—RUZHENTSEV, V.: *Results of Ukrainian national public cryptographic competition*, Tatra Mt. Math. Publ. **47** (2010), 99–113.

Received September 2, 2014

Dmytro Kaidalov
Roman Oliynykov
Department of Information Technologies Security
Faculty of Computer Engineering and Control
Kharkiv National University of Radioelectronics
Lenin's av., 14
Kharkiv
UKRAINE
E-mail: dmtr.kd@gmail.com
roliynykov@gmail.com

Oleksandr Kazymyrov
Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Bergen
Postboks 7803, N-5020
Bergen
NORWAY
E-mail: oleksandr.kazymyrov@ii.uib.no