LETTER *Special Section on Cryptography and Information Security*

A Meet in the Middle Attack on Reduced Round Kuznyechik

Riham ALTAWY[†a)], ***Member and*** **Amr M. YOUSSEF**[†b)], ***Nonmember***

SUMMARY In this letter, we present a meet-in-the-middle attack on the 5-round reduced Kuznyechik cipher which has been recently chosen to be standardized by the Russian federation. Our attack is based on the differential enumeration approach. However, the application of the exact approach is not successful on Kuznyechik due to its optimal round diffusion properties. Accordingly, we adopt an equivalent representation for the last round where we can efficiently filter ciphertext pairs and launch the attack in the chosen ciphertext setting. We also utilize partial sequence matching which further reduces the memory and time complexities. For the 5-round reduced cipher, the 256-bit master key is recovered with an online time complexity of $2^{140.3}$, a memory complexity of $2^{153.3}$, and a data complexity of 2^{113}.

key words: *Kuznyechik, Cryptanalysis, Meet-in-the-middle attacks, Differential enumeration, GOST-Grasshopper.*

1. Introduction

The Russian Federation has recently published a project for a new standard for block cipher encryption algorithm [1]. A draft for this new algorithm was presented by its designers at CTCrypt 2014 [2]. The new algorithm, Kuznyechik, (Grasshopper in Russian), is in the process of being standardized [1] to accompany the current Russian encryption standard GOST 28147-89 [3]. Fault analysis attacks on Kuznyechik have been proposed in [4].

The first application of a non standard type of MitM attacks on AES was due to the work of Demirci and Selçuk [5] where they showed that if the input to the distinguisher has only one active byte that takes all the possible values, then each output byte can be evaluated as a function of 25 byte parameters which elevated the memory requirements of the attack. They also showed that the values of each output byte corresponding to the input byte values form an ordered sequence that can be used as a property to identify the right key guess. Afterwards, Dunkelman *et al.* proposed the idea of multisets and differential enumeration [6] to tackle the high memory requirements of the approach of Demirci and Selçuk. Differential enumeration allows the ordered sequence to be generated by the knowledge of 16 byte parameters only instead of 24, which brings down the number of entries of the table from 2^{192} to 2^{128}. Later on, Derbez *et al.* [7] improved the attack of Dunkelman *et al.* by borrowing ideas from the rebound attack [8] which further brought the number of required parameters to 10 bytes, thus the number of entries of the precomputation table is further reduced to 2^{80}. Moreover, a 9-round attack on AES-192 using a 5-round truncated differential distinguisher was presented by Li *et al.* in [9]. Finally, Guo *et al.* generalized the attack on Feistel structures in [10].

In this work, we present a MitM attack on Kuznyechik using the idea of efficient differential enumeration. Unlike AES, Kuznyechik employs an optimal diffusion transformation applied to the whole state, where one byte difference results in a full active state with certainty after one round. Consequently, we construct a three round distinguisher in our attack to recover 16-bytes of the master key of the reduced 5-round cipher. The direct application of the attack on Kuznyechik requires a time complexity that exceeds that of the exhaustive search for the 256-bit key, which is attributed to the optimal round diffusion. Accordingly, we adopt an equivalent representation of the last round which allows us to efficiently select ciphertext pairs that satisfy the lower half of the differential path used in the attack with certainty. Hence, our attack is considered in the chosen ciphertext setting. This modification lowers the time complexity of the online phase by a factor of 2^{120} because we eliminate the probabilistic propagation of the 16 to 1 transition through the linear transformation from the ciphertext side. We also present partial sequence matching, by which we generate, store, and match parts of the ordered sequence while maintaining negligible probability of error. Indeed, not only we decrease the partially encrypted/decrypted data during online matching and thus the overall online time complexity of the attack is lowered, but this approach also reduces the memory requirements of the attack.

2. Specification of Kuznyechik

Kuznyechik [1], [2] is an SPN block cipher that operates on a 128-bit state. The cipher employs a 256-bit key which is used to generate ten 128-bit round keys. The encryption procedure updates the 16-byte state by iterating the round function for nine rounds. The round function consists of:

[†]Concordia Institute for Information Systems Engineering, Concordia University, Montréal, Québec, Canada
a) E-mail: r_altaw@encs.concordia.ca
b) E-mail: youssef@ciise.concordia.ca
DOI: 10.1587/transfun.E0.A.1

- SubBytes (S): A nonlinear byte bijective mapping.
- Linear Transformation (L): An optimal diffusion operation that operates on a 16-byte input and has a branch number = 17.
- Xor layer (X): Mixes round keys with the encryption state.

Additionally, an initial XOR layer is applied prior to the first round. The full encryption function where the ciphertext C is evaluated from the plaintext P is given by:

$$C = (X[K_{10}] \circ L \circ S) \circ \cdots \circ (X[K_2] \circ L \circ S) \circ X[K_1](P)$$

In our attack, we use an equivalent representation of the last round function. The representation exploits the fact that both the linear transformation, L, and the Xor operation, X, are linear and thus, their order can be swapped. One has to first Xor the data with an equivalent round key, then apply the linear transformation, L, to the result. We evaluate the equivalent round key after the last round r by $EK_{r+1} = L^{-1}(K_{r+1})$. We also use the following property of the Sbox:

Property 1. For a given Sbox differential $(\delta x, \delta y)$, the average number of solutions to $S(x) \oplus S(x \oplus \delta x) = \delta y$ is 1.

For further details regarding the employed Sboxes and linear transformation and key scheduling process, the reader is referred to [2]. The following notation is used throughout the letter:

- x_i, y_i, z_i: The 16-byte state after the X, S, L operation, respectively, at round i.
- x_i^j: The state at round i whose position within a set or a sequence is given by j.
- $x_i[j]$: The j^{th} byte of the state x_i, where $j = 0, 1, \cdots, 15$, and the bytes are indexed from left to right.
- Δx_i, $\Delta x_i[j]$: The difference at state x_i, and byte $x_i[j]$, respectively.

The memory complexity of our attack is given in 16-byte states and the time complexity is evaluated in reduced round Kuznyechik encryptions. In the following sections, we give a preliminary security analysis of Kuznyechik and present the details of our MitM attack.

3. Security Analysis of Kuznyechik

The designers of Kuznyechik did not provide any security analysis of the cipher. Accordingly, we give our analysis of the cipher against some well known attacks.

3.1 Differential and Linear Cryptanalysis

Since Kuznyechik employs the same Sbox as the Russian hash function standard Streebog [2], we use the Sbox properties presented in [11]. The linear transformation of the cipher has an optimal branch number of 17. The maximum Sbox differential probability $= 8/256 = 2^{-5}$. Accordingly, the maximum probability of a differential characteristic over two rounds is $(2^{-5})^{17} = 2^{-85}$. Also, with an Sbox nonlinearity of 100, there is no linear approximation over two rounds with an input-output correlation larger than $((128-100)/128)^{17} \approx 2^{-37.27}$. Furthermore, let N_r denote the minimum number of differentially or linearly active Sboxes for r rounds, $r = 1, \cdots, 9$. Then, using the mixed integer linear programming approach proposed in [12], one can show that $N_r = 1, 17, 18, 34, 35, 51, 52, 68, 69$, for $r = 1, 2, \cdots, 9$, respectively. Consequently, given the block length of 128-bits and by noting the data complexity of differential and linear cryptanalysis, there is no useful differential or linear characteristic of length more than three rounds.

3.2 Related-key Cryptanalysis

This attack exploits either the slow diffusion or the symmetry in the key schedule. The Kuznyechik key schedule employs a Feistel structure with the same round function used in the encryption rounds. This round function is designed to cause fast and nonlinear diffusion between round keys. Also, eight rounds of processing are used between round keys extraction which makes it infeasible to propagate and maintain a given relation between keys from two successive extractions.

3.3 Integral Cryptanalysis

Using a set of 2^8 chosen plaintexts which differ in one byte that takes all the 2^8 values and the remaining fifteen bytes are equal, results in a zero sum of all 256 cipher states at every byte position after two rounds. Accordingly, one can recover the third round key by guessing the key bytes independently, decrypting the corresponding 256 ciphers, and checking for the zero sum at the respective position. Once the last round key is recovered, one can peal this round off and repeat the attack to recover the key in the round before the last. The time complexity for recovering the last round key of the three rounds reduced cipher is $16 \times 2^8 \times 2^8 \approx 2^{20}$ and for recovering the whole master key is 2^{21}. Extending the attack to four rounds can be done by guessing the last round key and then launching the previous attack on the first three rounds to recover the third round key which increases the time complexity to $2^{20+128} = 2^{148}$.

3.4 Higher Order Differential Cryptanalysis

Since the algebraic degree of the Sbox is 7, then the degree of two consecutive rounds of the cipher is at most $7 \times 7 = 49$ and any 50-th (or higher) order derivative must be 0. One can append an additional round before these two rounds by choosing 2^{56} plaintext inputs with seven active Sboxes. The fourth round subkey bytes are then recovered independently with a time complexity of $16 \times 2^{56+8} = 2^{68}$.

4. A MitM Attack using Differential Enumeration on Kuznyechik

In our analysis we do not exploit specific properties of the key scheduling algorithm or the employed linear transformation. Thus our attack is applicable to SPN ciphers which employ a round function that consists of a substitution layer followed by an optimal diffusion linear transformation. The presented complexities are valid for the 128-bit block length and 256-bit key length version. Generally, our attack divides the reduced Kuznyechik block cipher, C_K, into three parts, such that $C_K = C_{k_2} \circ C^m \circ C_{k_1}$, where C^m is the middle part of the cipher which exhibits a distinguishing property. The employed property is evaluated without the knowledge of the key bits used in these middle rounds. Hence, correct round key candidates for k_1 and k_2 are checked if they verify this distinguishing property or not. Our middle distinguisher is a truncated differential characteristic such that, when a set of input states from a *δ-set* [13] is presented as its input, the set of each byte of the output state forms an ordered sequence.

Definition 1. (*δ-set* of Kuznyechick) *is a set of 256 states where one byte at a particular state takes all the* 2^8 *possible values and the rest of the 15 bytes are constants.*

In our MitM attack, we employ a distinguisher where the *δ-set* is presented at their input from the ciphertext side, thus, after partially decrypting it, we acquire the corresponding ordered sequence. We denote the *δ-set* at state x_i resulting from changing the byte at position j by δs^j, $j = 0, 1, \cdots, 15$, where $\delta s^j = \{x_i^0, x_i^1, \cdots, x_i^{255}\}$. We also denote the set of 255 differences at byte $x_{i-r}[k]$ which form the ordered sequence for an r round distinguisher by os^k, $k = 0, 1, \cdots, 15$, where

$$os^k = \{\Delta^1 x_{i-r}[k], \Delta^2 x_{i-r}[k], \cdots, \Delta^{255} x_{i-r}[k]\},$$

and $\Delta^l x_{i-r}[k] = x_{i-r}^0[k] \oplus x_{i-r}^l[k]$, for $l = 1, 2, \cdots, 255$. The correct ordered sequence os^k is evaluated by partially decrypting the 256 bytes which are different in the *δ-set* for r rounds. However, we compute all the possible sequences so that one does not need to know the key bits involved in this encryption process. because we simply compute it using all the possible values of the involved parameters. Our proposed 5-round MitM attack employs a three round distinguisher. Figure 1 depicts the differential path used in the attack in which we embed a $1 \rightarrow 16 \rightarrow 16 \rightarrow 1$ distinguisher that starts at x_5 and ends at x_2. The length of the distinguisher is restricted by the properties of optimal linear transformation used in the Kuznyechik round which guaranties full diffusion in one round. In what follows, we give the details of the attack steps.

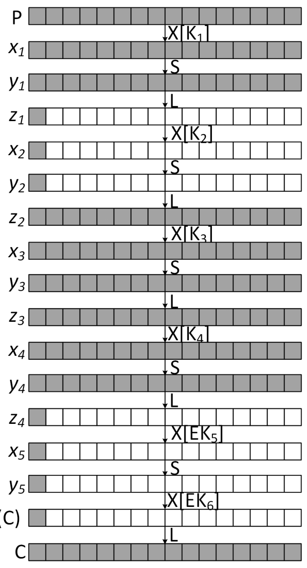

Fig. 1 Differential path used in the 5-round attack.

4.1 Attack Procedure:

The attack recovers the 128-bit first round key K_1 and one byte of $EK_6 = L^{-1}(K_6)$. The attack is composed of precomputation and online phases. In the precomputation phase, we iterate on all the values of the parameters required for evaluating the ordered sequence, and for each value, we deduce its corresponding 33 bytes values which are then used to generate the ordered sequence table. The online phase is further divided into data collection and filtration, and key recovery phases. In the data collection phase, we collect many pairs such that one of them satisfies the 5-round differential characteristics given in Figure 1. Finally, In the key recovery phase, for each of the obtained pairs, we guess some key bits, compute the ordered sequence, and match it with the ones stored in the precomputed table.

Precomputaion phase: In this phase, we construct a lookup table that contains 2^{152} ordered sequences of the resulting 255 difference,

$$os^0 = \{\Delta^1 x_2[0], \Delta^2 x_2[0], \cdots, \Delta^{255} x_2[0]\},$$

from the $\delta s^0 = \{x_5^0, x_5^1, \cdots, x_5^{255}\}$. As depicted in Figure 1, our *δ-set* is the set of states resulting from changing the first byte at state x_5 and is given by: $\delta s^0 = \{x_5^0, x_5^1, \cdots, x_5^{255}\}$. The corresponding ordered sequence:

$$os^0 = \{\Delta^1 x_2[0], \Delta^2 x_2[0], \cdot, \Delta^{255} x_2[0]\}$$

is evaluated by the knowledge of the values of 33 bytes. More precisely, in addition to δs^0, given the values of 16 bytes at y_4, 16 bytes at y_3 and 1 byte at $y_2[0]$, the i^{th} element, $\Delta^i x_2[0]$ of the ordered sequence is computed as follows:

- Compute $\Delta^i x_5[0] = x_5^0[0] \oplus x_5^i[0]$ from δs^0 and compute the value of $\Delta^i y_4$.
- Using the value of y_4 and $\Delta^i y_4$, pass the substitution layer with certainty, evaluate $\Delta^i x_4$ and compute the value of $\Delta^i y_3$.
- Using the value of y_3 and $\Delta^i y_3$, evaluate $\Delta^i x_3$, and compute $\Delta^i y_2[0]$.
- Using $\Delta^i y_2[0]$ and $y_2[0]$, compute $\Delta^i x_2[0]$.

By employing the rebound based differential enumeration technique [7], we deduce that if x_5^0 of δs^0 belongs to a pair of messages that follows the differential path in Figure 1, then the corresponding ordered sequence os^0 can have only 2^{152} values. Accordingly, a given ordered sequence can be computed by the knowledge of 19 byte parameters only. These parameters are $\Delta x_5[0]$, y_4, $y_2[0]$, and $\Delta y_2[0]$, where $\Delta x_5[0]$ and $\Delta y_2[0]$ denote the differences generated by a conforming message pair.

The procedure for evaluating the 2^{152} sequences from these 19 parameters can be summarized as follows:

1. For each of the 2^{152} possible values of $\Delta x_5[0] \parallel y_4 \parallel y_2[0] \parallel \Delta y_2[0]$, evaluate the values of the 33 bytes required to compute the ordered sequence, which are $y_2[0]$, y_3, and y_4 as follows:
   - Linearly propagate $\Delta x_5[0]$ backwards to evaluate Δy_4.
   - Using Δy_4 and y_4, evaluate Δy_3 and compute Δx_3.
   - Find x_3, such that $S(x_3) \oplus S(x_3 \oplus \Delta x_3) = \Delta y_3$. According to property 1 in Section 2, not all the 2^{152} differentials are possible, but the ones that are possible results in about 2^{16} solutions on average so we get one solution on average.
   - Evaluate $y_3 = S(x_3)$.
2. The additional knowledge of the evaluated value of y_3 provides us with the values of the 33 bytes required to compute the 255 differences $\Delta^l x_2[0]$, $l = 1, 2, \cdots, 255$, of the ordered sequence as described in the previous procedure.
3. Store all the generated sequences in a hash table.

Note that two differences are used at state x_5. The first difference $\Delta^i x_5[0]$ is the difference in the first byte of the i^{th} x_5 state within the delta set, i.e., $\Delta^i x_5[0] = x_5^0[0] \oplus x_5^i[0]$, where $i = 1, \cdots, 255$. The second difference $\Delta x_5[0]$ is the difference in the first byte of state x_5 that belongs to an input pair which conforms to the differential path. In other words, given a pair of conforming inputs with (x_5, x_5') states, $\Delta x_5[0] = x_5[0] \oplus x_5'[0]$.

Data collection and filtration: In this phase, we find enough pairs of messages such that one of them conforms to the truncated differential characteristic in Figure 1.

1. To get one ciphertext structure, randomly pick the value of the rightmost fifteen bytes of $L^{-1}(C)$ and let the first byte take all the possible 256 values. This structure generates about $\frac{2^8 \times (2^8-1)}{2} \approx 2^{15}$ pairs. This step guaranties that all the corresponding (C, C') pairs in the structure conform to the $16 \rightarrow 1$ transition in round 5.
2. Query the decryption oracle for the plaintext pairs (P, P') corresponding to the ciphertext pairs generated in step 1. These pairs are not necessarily going to conform to the $16 \rightarrow 1$ transition in round 1, which happens with probability 2^{-120}.
3. Store the 2^8 plaintexts and their corresponding ciphertexts in a hash table.
4. To get one pair of plaintexts (P, P') that satisfy the $16 \rightarrow 1$ probabilistic transition, we need to try approximately 2^{120} pairs. Since, each structure provides 2^{15} pairs, one requires about 2^{105} structures, and hence the above steps are repeated 2^{105} times.

All in all, we ask for the decryption of $2^{105} \times 2^8 = 2^{113}$ chosen ciphertexts to get the required 2^{120} pairs.

Key recovery: For each plaintext pair(P_i, P_i') and their corresponding ciphertext pair (C_i, C_i'), we deduce the possible values of K_1 and guess the value of $EK_6[0]$ to compute a candidate ordered sequence as follows:

1. Guess a value for $\Delta x_2[0]$, and linearly propagate it backwards to get the value of Δy_1.
2. Using the fact that $\Delta x_1 = \Delta P_i$, find the value of x_1 which provides a solution for $\Delta y_1 = S(x_1) \oplus S(x_1 \oplus \Delta x_1)$. According to property 1 in section 2, we get one solution on average.
3. Evaluate $K_1 = P_i \oplus x_1$. By repeating the previous two steps for all the possible guesses of $\Delta x_2[0]$, we get 2^8 candidate values for K_1.
4. For each candidate of the 2^8 values of K_1 and for each guess of the 2^8 guesses of $EK_6[0]$, use C_i to get the rest of the 255 ciphertexts C_i^j for $j = 1, 2, \cdots, 255$, corresponding to the δs^0 generated by C_i as follows:
   - The value of $x_5[0]$ which is the first byte of the first state in δs^0 is evaluated as:
   $$x_5[0] = S^{-1}(L^{-1}(C_i)[0] \oplus EK_6[0]).$$
   - The set of different values of $x_5[0]$ in the states of δs^0 has the following structure:
   $$\{x_5[0], x_5[0] \oplus \Delta_1, x_5[0] \oplus \Delta_2, \cdots, x_5[0] \oplus \Delta_{255}\},$$
   and $\Delta_j = j$ for $j = 1, 2, \cdots, 255$. Accordingly, we can evaluate the 255 values of $L^{-1}(C_i^j)[0]$ corresponding to the values of $x_5[0] \oplus \Delta_j$ by
   $$L^{-1}(C_i^j)[0] = S(x_5[0] \oplus \Delta_j) \oplus EK_6[0].$$
   - Get the difference $\Delta^j L^{-1}(C_i)[0] = L^{-1}(C_i^j)[0] \oplus$

$L^{-1}(C_i)[0]$, and propagate it through the linear transformation to get the corresponding difference $\Delta^j C_i$. Finally, the required 255 values of C_i^j are evaluated by $C_i^j = C_i \oplus \Delta^j C_i$.

5. Get the 256 plaintexts $(P_i, P_i^1, \cdots, P_i^{255})$ corresponding to the ciphertexts generated in the previous step from the currently stored structure.
6. Using K_1, partially encrypt the plaintexts $(P_i, P_i^1, \cdots, P_i^{255})$ to get the 256 values of $\Delta^j x_2[0]$, which form the ordered sequence os^0.
7. Check if there is a match between the computed os^0 and the 2^{152} ordered sequences stored in the precomputed table. If there is no match, we discard the candidate K_1 and $EK_6[0]$ with certainty.

The probability of a wrong key producing a valid 255 byte ordered sequence is given by $2^{152+120+16-2040} = 2^{-1752}$. The memory requirements of the attack is given by $2^{152} \times 2040/128 \approx 2^{156}$ 128-bit states. The data complexity of the attack is 2^{113} chosen ciphertexts. The time complexity of the attack is attributed to the off-line precomputation phase for constructing the table and the online phase for recovering the master key. Thus the off-line time complexity is given by $\approx 2^{152}$ and the online time complexity for recovering the first round key is $\approx 2^{(120+16+8)} \times 2/5 \approx 2^{143}$. As it is fairly complex to deduce any relation between the recovered K_1 and $EK_6[0]$ that can aid us in the recovery of K_2 so that we can recover the master key, we exhaustively search for it which does not affect the attack online complexity significantly.

Partial sequence matching: One can reduce the number of encryption/decryption operations to $b < 2^8$, where b denotes the number of differences stored in the ordered sequence. In other words, since the probability of error is so small, it can be relaxed such that we match b bytes of the 2^8 of the ordered sequence to identify the right key. More precisely, if we accept the error probability to be 2^{-32}, which is still negligible, the required number of bytes, b, is evaluated by $2^{-32} = 2^{120+16+152-8b}$. Hence, it is enough to match 40 bytes of the ordered sequence Accordingly, the memory complexity of the attack is reduced to $2^{152} \times (320/128) \approx 2^{153.3}$ states, and the online time complexity is evaluated by $2^{120+16} \times 2^{5.3} \times 2/5 + 2^{128} \approx 2^{140.3}$.

5. Conclusion

In this letter, we have presented a MitM attack on the new draft of the Russian encryption standard using the idea of efficient differential enumeration. We have proposed an initial filtration stage which lowers the online time complexity of the basic approach by a factor of 2^{120}. Instead of trying random data pairs such that the truncated differential path is satisfied probabilistically, we carefully compose ciphertext pairs so that the lower half of the path is conformed to with certainty. Additionally, we have adopted partial sequence matching, by which we store and match parts of the ordered sequences while maintaining a negligible probability of error which reduces both the memory and time complexities of the attack. Our attack on the 5-round reduced cipher has a memory complexity of $2^{153.3}$, an online time complexity of $2^{140.3}$, and a data complexity of 2^{113} chosen ciphertext. While the proposed attack may not present direct threat to the security of Kuznyechik, it is considered a forward step in the public cryptanalysis of this soon to be the new Russian block cipher standard.

References

[1] "The National Standard of the Russian Federation GOST R 34._-20_." Russian Federal Agency on Technical Regulation and Metrology report, 2015. `http://www.tc26.ru/en/standard/draft/ENG_GOST_R_bsh.pdf`.

[2] V. Shishkin, D. Dygin, I. Lavrikov, G. Marshalko, V. Rudskoy, and D. Trifonov, "Low-Weight and Hi-End: Draft Russian Encryption Standard," CTCrypt, pp.183–188, 2014.

[3] "GOST 28147-89." Information Processing Systems. Cryptographic Protection. Cryptographic Transformation Algorithm. *(In Russian)*.

[4] R. AlTawy, O. Duman, and A.M. Youssef, "Fault analysis of Kuznyechik," CTCrypt, pp.302–3017, 2015. Available at: http://eprint.iacr.org/2015/347.

[5] H. Demirci and A. Selçuk, "A meet-in-the-middle attack on 8-round AES," FSE, ed. K. Nyberg, LNCS, vol.5086, pp.116–126, Springer, 2008.

[6] O. Dunkelman, N. Keller, and A. Shamir, "Improved single-key attacks on 8-round AES-192 and AES-256," ASIACRYPT, ed. M. Abe, LNCS, vol.6477, pp.158–176, Springer, 2010.

[7] P. Derbez, P.A. Fouque, and J. Jean, "Improved key recovery attacks on reduced-round AES in the single-key setting," EUROCRYPT, ed. T. Johansson and P. Nguyen, LNCS, vol.7881, pp.371–387, Springer, 2013.

[8] F. Mendel, C. Rechberger, M. Schläffer, and S.S. Thomsen, "The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl," FSE, ed. O. Dunkelman, LNCS, vol.5665, pp.260–276, Springer, 2009.

[9] L. Li, K. Jia, and X. Wang, "Improved single-key attacks on 9-round AES-192/256," FSE, ed. C. Cid and C. Rechberger, LNCS, Springer, 2014. *(to appear)*.

[10] J. Guo, J. Jean, I. Nikolić, and Y. Sasaki, "Meet-in-the-middle attacks on generic Feistel constructions," ASIACRYPT, ed. P. Sarkar and T. Iwata, LNCS, vol.8873, pp.458–477, Springer, 2014.

[11] O. Kazymyrov and V. Kazymyrova, "Algebraic aspects of the russian hash standard GOST R 34.11-2012," CTCrypt, pp.160–176, 2013. Available at: http://eprint.iacr.org/2013/556.

[12] N. Mouha, Q. Wang, D. Gu, and B. Preneel, "Differential and linear cryptanalysis using mixed-integer linear programming," Information Security and Cryptology, ed. P. Sarkar and T. Iwata, LNCS, vol.7537, pp.57–76, Springer, 2012.

[13] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1998.