A Meet-in-the-Middle Attack on Reduced-Round Kalyna-$b/2b$

Riham AlTawy, Ahmed Abdelkhalek, and Amr M. Youssef

Concordia Institute for Information Systems Engineering
Concordia University, Montréal, Québec, Canada.

Abstract. Kalyna is an SPN-based block cipher that was selected during Ukrainian national public cryptographic competition (2007-2010), and its slight modification was approved as the new encryption standard of Ukraine (DSTU 7624:2014) in 2015. The cipher supports a block size and a key length of 128, 256 and 512 bits where the size of the key can be either double or equal to that of the block length. According to its designers, the cipher provides strength to several cryptanalytic methods after the fifth and sixth rounds of the 128-bit and 256-bit block versions, respectively. In this paper, we present a meet-in-the-middle attack on the 7-round reduced versions of Kalyna where the key size is double the block length. Our attack is based on the differential enumeration approach where we carefully deploy a four round distinguisher in the first four rounds to bypass the effect of the carry bits resulting from the pre-whitening modular key addition. We also exploit the linear relation between consecutive odd and even indexed round keys which enables us to attack seven rounds and recover all the round keys incrementally. The attack on Kalyna with 128-bit block has a data complexity of 2^{89} chosen plaintexts, time complexity of $2^{230.2}$ and a memory complexity of $2^{202.64}$. The data, time and memory complexities of our attack on Kalyna with 256-bit block are 2^{233}, $2^{502.2}$ and 2^{170}, respectively.

Keywords: Cryptanalysis, Kalyna, DSTU 7624:2014, Meet-in-the-Middle attack, Differential Enumeration.

1 Introduction

Kalyna [17] is an SPN cipher that won the national public cryptographic competition [2] organized by the state service of special communication and information protection of Ukraine. This competition aimed to select a block cipher to become the new Ukrainian national encryption standard [16] and replace the legacy standard GOST 28147-89 [1]. Kalyna was chosen in 2014 and after a slight modification, in 2015, it officially became the new encryption standard of Ukraine known as DSTU 7624:2014 [1].

Kalyna supports block sizes of 128-bit, 256-bit, and 512-bit, and key sizes of 128-bit, 256-bit, and 512-bit where the key size can be equal to or double the block length. In this paper we will refer to a specific version of the cipher as Kalyna$-b/k$, where b and k denote the employed block and key lengths, respectively. Although the exact analysis of the resistance of Kalyna to various attacks has not been discussed by its designer in [17], they concluded that the cipher is sufficiently secure against several cryptanalytic methods after rounds five and six when the block size is 128-bit and 256-bit, respectively (cf. page 14 of [15]).

The classical meet-in-the-middle (MitM) attack [9] has not been successful on AES until Demirci and Selçuk proposed a modified MitM approach to cryptanalyze it [6]. They have shown that the value of a given byte of the output of a four round of encryption can be evaluated as a function of 25 byte parameters and a given active byte in the input. 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 distinguishing property to identify the right key guess. The main disadvantage of their technique is the high memory complexity which is required by a precomputation table that is used to store all the sequences resulting from all the possible combinations of the 25 byte parameters. Accordingly, the approach was only valid to attack seven and eight rounds of AES-192 and AES-256, but not the 128-bit version. Afterwards, the number of parameters was reduced to 24 bytes in [7] where differences were used instead of the exact values in the ordered sequence, which reduced the size of the table by a factor of 8.

In the sequel, Dunkelman *et al.* targeted the problem of the high memory requirements of the MitM attack by introducing two new techniques [10]. They first proposed the idea of multisets which provides efficient encoding of the ordered sequence which reduces the size of the table by a factor of 4. Additionally, they introduced differential enumeration that enables the generation of ordered sequence as a function of 16 byte parameters only instead of 24, which reduced the number of entries of the table from 2^{192} to 2^{128}. This memory cost reduction was achieved by employing a truncated differential characteristic where the generated sequence at its output can only take a restricted number of values. Accordingly, one must initially search through a large amount of input data pairs to find one pair that satisfies the chosen distinguisher. Indeed, their proposal has reduced the memory complexity of the attack at the expense of its data complexity required to search for the right input data pair.

Later on, Derbez *et al.* [8] improved the attack of Dunkelman *et al.* by borrowing ideas from the rebound attack [14] where they have proved that not all of the sequences in the table can be verified by input data satisfying the truncated distinguisher. Derbez *et al.* presented an efficient enumeration technique and showed that the whole set of sequences can take only 2^{80} values and not 2^{128} as with the case in the attack by Dunkelman *et al.* Accordingly, all the generated sequences require the knowledge of only 10 byte parameters, thus the number of entries of the precomputation table is further reduced to 2^{80}. A direct consequence of their improvement is that the memory complexity is not the bottleneck of the attack anymore but both the time and data complexities are. Nevertheless, their attack is considered the most efficient attack on the 7-round reduced AES-128 and 8-round reduced AES-192/256. They have also used a 5-round distingusher to attack the 9-rounds reduced AES-256.

Afterwards, Li *et al.* [13] employed a key-dependent sieve to further reduce the memory complexity of the attack and present an attack on 9 rounds AES-192 using a 5-round truncated differential distinguisher. MitM attacks using differential enumeration have been used to analyze mCrypton [12], the Russian encryption standard Kuznyechik [4], and Hierocrypt-3 [3]. The attack was further generalized to present a framework for cryptanalyzing Feistel-based ciphers [11].

In this work, we present a MitM attack on seven round reduced Kalyna-$b/2b$ utilizing the idea of efficient differential enumeration. Kalyna employs a pre- and post-whitening key mixing using addition modulo 2^{64}. Accordingly, we deploy a specific four round distinguisher that covers the first four rounds where the active byte is chosen to prevent the propagation of differences to the neighboring bytes. We also exploit the linear relation between odd and even indexed round keys to efficiently recover the last two round keys. The key schedule of Kalyna is designed to make it computationally infeasible to retrieve the master key from the round keys. For that reason, we propose an approach to recover all the round keys using parameters matching. Employing this proposed technique, we use the parameters corresponding to the matching multiset to filter pairs of two consecutive round keys guesses.

The rest of the paper is organized as follows. In the next section, the description of the Kalyna block cipher along with the notation used throughout the paper are provided. Afterwards, in section 3, we provide a detailed description of the proposed distinguisher, the adopted attack procedure, and our round keys recovery approach. Finally, the paper is concluded in section 4.

2 Specifications of Kalyna

Our attack targets Kalyna-$b/2b$ where the size of the key is double that of the state (i.e., Kalyna-128/256 and Kalyna-256/512). Accordingly, in this section, we give the description of the encryption and round key generation procedures of Kalyna-$b/2b$ when $b = 128$ bits. The encryption procedure of Kalyna-128/256 and Kalyna-256/512 runs an AES-like round function for 14 and 18 times updating an $8 \times c$ state, respectively, where c denotes the number of columns in the block state and is equal to 2 and 4 for the 128 and 256-bit block, respectively. As depicted in Figure 1, the encryption procedure employs a pre- and post-whitening stages using addition modulo 2^{64} applied on the state columns independently. In the sequel, the round function is iterated for 13 rounds. Each round applies the following transformations on the state:

- SubBytes (SB): A layer of 8-bit substitution boxes.
- ShiftRows(SR): A transformation that cyclically right shifts the rows of the state. The value of the shift is given by $\lfloor \frac{i \cdot b}{512} \rfloor$, where $i = 0, 1, \cdots 7$ and $b = 128$ denote the row number and state size, respectively.
- MixColumns(MC): A transformation that multiplies the columns of the state independently by an MDS matrix.

– X: A round key mixing layer consisting of xoring the state with the round keys.

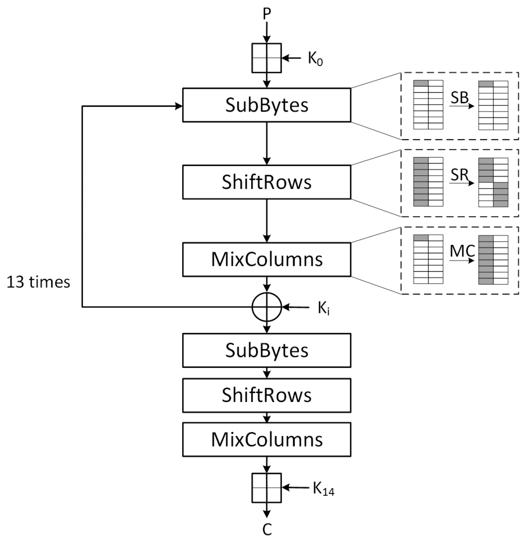

Fig. 1. Kalyna-128/256 encryption function.

In the last round, the X transformation is replaced by a post-whitening modular key addition. Hence, the full encryption function of the Kalyna-128/256 where the ciphertext C is evaluated from the plaintext P can be described as:

$$C = K_{14} + (MC \circ SR \circ SB \circ \cdots \circ X[K_1] \circ MC \circ SR \circ SB(P + K_0))$$

In our analysis, we use the following property of the Sbox:

Proposition 1 *Given two non-zero differences in* $\mathbb{F}_{256}$, Δ_x *and* Δ_y, *the average number of solutions for* $SB(x) \oplus SB(x \oplus \Delta_x) = \Delta_y$ *is one.*

Key schedule. Round keys are independently evaluated from the master key, K, and an intermediate key, K_σ. The process of calculating the intermediate key is illustrated in the left side of Figure 2 where the 8×2 byte state is initialized by the value of $\frac{b+k+64}{64}$, which is equal to $(128 + 256 + 64)/64 = 7$ for Kalyna-128/256. Given K_α and K_ω which denote the least and most significant $k/2 = 128$ bits of the master key, $K = K_\omega \parallel K_\alpha$, respectively, the state undergoes key mixing for three rounds where K_α and K_ω are used alternately. Afterwards, the even indexed round keys are generated independently by the process depicted on the right side of Figure 2. The round key state is first initialized by K_{in}, then it undergoes two encryption rounds where the intermediate key K_σ is added to a round constant tmv_i and used for key mixing. K_{in} is evaluated according to the round number, i, and is given by the least significant 128-bit of $(K >\gg 16 \cdot i)$ for round indices divisible by 4 and the most significant 128-bit of $(K \ggg 64 \cdot \lfloor i/4 \rfloor)$ for round indices not divisible by 4. Odd indexed round keys

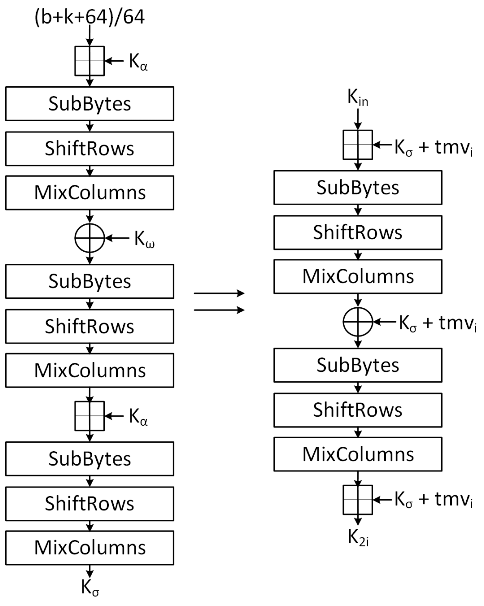

Fig. 2. Kalyna key schedule.

are linearly computed from their previous even indexed round keys according to the formula:

$$K_i = K_{i-1} \lll (b/4 + 24).$$

For further details regarding the SBoxes, the linear transformation or the key schedule of other versions, the reader is referred to [17].

2.1 Notations

The following notations are used throughout the paper:

- x_i, y_i, z_i, and w_i: the 8×2 bytes state after the X or addition module 2^{64}, SB, SR, and MC transformations at round i, respectively.
- $x_i[j]$: The j^{th} byte of the state x_i, where $j = 0, 1, \cdots, 15$, and the bytes are indexed column wise.
- x_i^j: The state at round i whose position within a set or a sequence is given by j.
- $x_i[j \cdots k]$: The bytes between the j^{th} and k^{th} positions inclusive of the state x_i.

– $\Delta x_i, \Delta x_i[j]$: The difference at state x_i and byte $x_i[j]$, respectively.

We measure memory complexity of our attack in b-bit Kalyna-$b/2b$ blocks and time complexity in reduced-round Kalyna-$b/2b$ encryptions. In the following section, we give the details of our MitM attack on Kalyna-128/256.

3 A Differential Enumeration MitM Attack on Kalyna-128/256

In the employed MitM attack, the analyzed cipher C_K is divided into three parts such that $C_K = C_{k_2} \circ C^m \circ C_{k_1}$, where C^m verifies a distinguishing property. The employed property is evaluated regardless of the key bits used in these middle rounds. Hence, 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* [5] is used as its input, the set of a given byte difference of the output state forms an ordered sequence which can be represented using a multiset.

Definition 1 (δ-set of Kalyna-128/256) *Let a δ-set be a set of 256 Kalyna-128/256 states where one byte at a particular state position takes all the 2^8 possible values and the rest of the 15 bytes are constants.*

Definition 2 (Multisets of bytes) *A multiset generalizes the set concept by allowing elements to appear more than once. In our case, a multiset of 256 bytes can take as many as* $\binom{2^8+2^8-1}{2^8} \approx 2^{506.17}$ *different values [10].*

In our 7-round MitM attack, we employ a four round distinguisher that covers the following transitions:

$$1 \to 8 \to 16 \to 8 \to 4.$$

As depicted in Figure 3, the distinguisher starts at x_0 where byte $x_0[15]$ takes all the possible 2^8 values and ends at z_4, where we evaluate the multiset of the 255 differences by partially encrypting the 256 values of x_0. We specifically locate the distinguisher in the first four rounds which enables us to exploit the linear relation between the last two round keys and attack seven rounds. Additionally, we choose the active byte at the beginning of the distinguisher in the most significant byte of the second column to prevent the propagation of the difference to the neighboring bytes, which can happen due to the carry propagation resulting from the modular addition key mixing. On the other hand, placing the distinguisher in the middle as in the traditional setting [6, 12] allows us to attack six rounds only. Also, we must deal with the probabilistic carry propagation in the analysis of the first round which reduces the path probability, and hence both the data and time complexities of the attack are increased. It should be noted that while our distinguisher ends with four active bytes which increases the path probability when we evaluate the multiset from the ciphertext side, this distinguisher does not affect the memory complexity because we store a multiset of the differences in only one of the four active bytes. In other words, since each byte out of the four active bytes at the end of the distinguisher forms an ordered sequence, we can choose any of them to distinguish between key candidates as long as the probability of error is negligible.

We denote the *δ-set* at state x_0 by δs, where

$$\delta s = \{x_0^0, x_0^1, \cdots, x_0^{255}\}.$$

We also denote the set of 255 differences at bytes $z_4[0\cdots3]$ by ds, where

$$ds = \{\Delta^1 z_4[0\cdots3], \Delta^2 z_4[0\cdots3], \cdots, \Delta^{255} z_4[0\cdots3]\},$$

and $\Delta^l z_4[0\cdots3] = z_4^0[0\cdots3] \oplus z_4^l[0\cdots3]$, for $l = 1, 2, \cdots, 255$. We opt for variating the most significant byte of state x_0 because plaintext pairs that differ in this byte result in one byte difference in x_0 after the modular addition key mixing as the carry is inhibited at the most significant bit. Since, we are using a multiset to encode the resulting set of differences, ds is evaluated by partially encrypting the 256 bytes which are different in state y_0 for 4 rounds as these set of states also form an unordered delta set corresponding to δs. We employ multisets to encode sets of differences in one of the resulting four byte differences only, which is possible because the probability of having a false match when using one byte differences is almost negligible. From the path depicted in Figure 3, we find that ds is evaluated by the knowledge of the values of 37 bytes. More precisely, given the values of $\Delta^l y_0[15]$, 8 bytes at $x_1[8\cdots15]$, 16 byte at x_2, 8 bytes at $x_3[0\cdots3]$, $x_3[12\cdots15]$, and 4 bytes at $x_4[0\cdots3]$, one can compute the value of $\Delta^l z_4[0\cdots3]$. However, by employing the rebound based differential enumeration technique [8], we deduce that if x_0^0 of δs belongs to a pair of plaintexts that conforms to the differential path in Figure 3, then the corresponding multiset of differences ds has only 2^{200} values. Accordingly, a given multiset of differences can be computed by the knowledge of 25 byte parameters only. These parameters are $\Delta y_0[15]$, $x_1[8\cdots15]$, $x_3[0\cdots3]$, $x_3[12\cdots15]$, $x_4[0\cdots3]$, and $\Delta z_4[0\cdots3]$, where $\Delta y_0[15]$ and $\Delta z_4[0\cdots3]$ denote the differences generated by a conforming message pair. In what follows, we give the procedure of the attack and show how we evaluate the 2^{200} multisets from these 25 parameters.

3.1 Attack Procedure

The attack exploits the linear relationship between consecutive even and odd indexed round keys to recover the last two 128-bit round keys K_7 and K_6. However, even with the knowledge of these two round keys, the recovery of the master key requires a time complexity equals to that of the exhaustive search. Consequently, once these two keys are known, we propose an additional step that recovers all the preceding round keys using parameters matching. The attack is composed of precomputation and online phases. In the precomputation phase, for each value of the values of the 25-byte parameters, we deduce the corresponding

37 bytes values which are then used to compute the multiset and store it in a hash table. The online phase is further divided into data collection, and key recovery phases. In the data collection phase, we query the encryption oracle with chosen plaintext pairs to find at least one pair that satisfies the 7-round path shown in Figure 3. In the key recovery phase, we test guesses of K_7 and K_6 with each plaintext pair to evaluate the multiset and search for it in the precomputed table.

Precomputaion phase: In this phase, we build a lookup table that contains 2^{200} multisets of the 255 difference in ds. Using the rebound approach, we iterate over the 2^{200} possible values of the 25 bytes $\Delta y_0[15]$, $x_1[8 \cdots 15]$, $x_3[0 \cdots 3]$, $x_3[12 \cdots 15]$, $x_4[0 \cdots 3]$, and $\Delta z_4[0 \cdots 3]$ to construct 2^{200} multisets of differences. The procedure can be summarized as follows:

– For each of the 2^{200} values of $\Delta y_0[15] \parallel x_1[8 \cdots 15] \parallel x_3[0 \cdots 3], x_3[12 \cdots 15] \parallel x_4[0 \cdots 3] \parallel \Delta z_4[0 \cdots 3]$, evaluate the value of x_2 as follows:
  1. Linearly propagate $\Delta y_0[15]$ forward to evaluate $\Delta x_1[8 \cdots 15]$.
  2. Using $x_1[8 \cdots 15]$, deduce Δx_2.
  3. Compute $\Delta y_3[0 \cdots 3]$, $\Delta y_3[12 \cdots 15]$ using $\Delta z_4[0 \cdots 3]$ and $x_4[0 \cdots 3]$.
  4. Using $x_3[0 \cdots 3]$, $x_3[12 \cdots 15]$, deduce Δy_2.
  5. Find x_2, such that $SB(x_2) \oplus SB(x_2 \oplus \Delta x_2) = \Delta y_2$. According to proposition 1, we get one solution on average.

– Having the value of x_2, we can now compute the 255 unordered differences $\Delta^i z_4[0 \cdots 3]$ in ds for $i = 1, 2, \cdots, 255$ as follows:
  1. Set $\Delta^i y_0[15] = i$. As the SBox is a permutation over $\mathbb{F}_{256}$, the sequence of $\Delta y_0[15]$ corresponds to the unordered sequence of $\Delta x_0[15]$ of the delta set.
  2. Linearly propagate $\Delta^i y_0[15]$ forward and compute the value of $\Delta^i x_1[8..15]$.
  3. Using the value of $x_1[8 \cdots 15]$ and $\Delta^i x_1[8 \cdots 15]$, pass the substitution layer with certainty and evaluate $\Delta^i x_2$.
  4. Using the value of x_2 and $\Delta^i x_2$, evaluate $\Delta^i x_3[0 \cdots 3]$, $\Delta^i x_3[12 \cdots 15]$.
  5. Propagate $\Delta^i x_3[0 \cdots 3]$, $\Delta^i x_3[12 \cdots 15]$ with the knowledge of $x_3[0 \cdots 3]$, $x_3[12 \cdots 15]$ through the Sboxes to evaluate $\Delta^i x_4[0 \cdots 3]$.
  6. Using $x_4[0 \cdots 3]$ and $\Delta^i x_4[0 \cdots 3]$, compute $\Delta^i z_4[0 \cdots 3]$.

– Encode the sequence of one out of the four generated byte differences, (i.e., $\Delta^i z_4[j]$, $j = 0, 1, 2, 3$) using a multiset and store it in a hash table.

Online phase: This phase is divided into two stages, data collection and key recovery. In the first stage, we collect enough pairs of plaintexts and their corresponding ciphertexts so that we acquire at least one pair that follows the path depicted in Figure 3. The second stage employs key guesses for K_7 and K_6 to evaluate candidate multisets from the collected ciphertext pairs, and matches them against the ones stored in the precomputed table to identify the correct round keys.

Data collection In this stage, we query the encryption oracle with structures of chosen plaintexts to get enough pairs such that one of them conforms to the whole truncated differential path. For each structure, we let the most significant state byte take all the possible 2^8 values and set the remaining 15 bytes to a constant value. We specifically choose to variate the most significant byte to ensure that we get one active byte at the beginning of the distinguisher after the modular addition key mixing. In other words, the carry generated by the modular addition is inhibited in the last bit of this byte, thus one active byte in the plaintext propagates to one active byte in x_0 with certainty. This structure results in about $\frac{2^8 \times (2^8-1)}{2} \approx 2^{15}$ pairs. While a chosen plaintext pair follows the forward path with certainty, the probability that its corresponding ciphertext pair conforms to the backward path is 2^{-96}. This probability is due to the $16 \rightarrow 8$ and $8 \rightarrow 4$ transitions through the inverse MixColumn transformation in rounds six and five, respectively. Accordingly, it is expected that when trying 2^{96} plaintext pairs, the corresponding ciphertext pair of one of them follows the path in Figure 3. Since, each structure provides 2^{15} pairs, one requires about 2^{81} structures. All in all, we ask for the encryption of $2^{81} \times 2^8 = 2^{89}$ chosen plaintexts to get the required 2^{96} pairs.

Key recovery: In this stage, for each plaintext pair (P_0, P'_0), we pick P_0 and construct the rest of the 255 plaintexts in its delta set by $P_i = P_0 \oplus i$ for $i = 1, 2, \cdots, 255$. Then, we get their corresponding 256 cipher texts C_i for $i = 0, 1, \cdots, 255$, partially decrypt them using guesses for K_7 and K_6 to get the 255 differences $\Delta^i z_4[0 \cdots 3]$. Note that we do not require to guess any bits from K_5 because the difference in x_5 can be linearly propagated to get the difference in z_4. In this stage, we exploit the linear relation between even and odd indexed round keys to identify the right K_7 and K_6 by guessing K_7 only and getting K_6 candidates for free. Finally, we evaluate the multiset of the 255 differences in one out of the four bytes in $\Delta^i z_4[0 \cdots 3]$ (the same byte that was used in the precomputation phase), and look for a match in the precomuted table. If a match is not found, we can discard that key candidate. The probability of a false match is given by $2^{200+96+128-467.6} = 2^{-43.6}$ which is negligible. Note that the probability of randomly having a match in the table is $2^{-467.6}$ (and not $2^{-506.7}$) because the number of ordered sequences associated to a multiset is not constant [8].

Attack Complexity: The memory requirement of the attack is dominated by the precomputation table needed to store 2^{200} multisets, each of 512 bits. Hence, the memory complexity of the attack is $2^{200} \times 512/128 = 2^{202}$ 128-bit blocks. The data complexity is determined by the data collection stage where we query the encryption oracle with 2^{89} chosen plaintexts. The time complexity of the offline phase is due to performing 2^{200} partial encryptions on 256 messages, which is equivalent to $2^{200+8} \times 5/7 \approx 2^{208}$ encryptions. The time complexity of the online phase to recover K_7 and K_6 is given by $2^{96+128+8} \times 2/7 \approx 2^{230.2}$.

3.2 Recovering the Remaining Round Keys

The key schedule of Kalyna is designed so that each even indexed round key is generated independently from the master key which is used in a two rounds encryption of an input state. Accordingly, the recovery of the master key from the round keys is computationally infeasible. For this reason, to be able to perform encryption and decryption, one can recover all the eight round keys instead of the master key. The MitM attack recovers the last two round keys, K_7 and K_6. These two keys are recovered when a specific plaintext pair and its corresponding ciphertext pair results in a match in the precomputed table. The matching sequence is calculated from a specific set of parameters which correspond to actual state values. Accordingly, if the parameters that represent state bytes are stored in the table along with the multiset, one can use them to filter round key guesses. In the precomputation phase, we store the values of the parameters $x_1[8 \cdots 15]$, x_2, $x_3[0 \cdots 3]$, $x_3[12 \cdots 15]$, and $x_4[0 \cdots 3]$ along with the multiset in the table entries. When a match is found using a given plaintext-ciphertext pairs, we partially decrypt one of the ciphertexts using the recovered K_7 and K_6 to evaluate state x_5. Then, we retrieve the stored parameters corresponding to the matching multiset, and perform the following incremental filtering to recover the remaining round key:

- For all the possible 2^{128} values of K_5, using the correct x_5, evaluate x_4. It is expected that 2^{96} key candidates produce states with bytes $x_4[0 \cdots 3]$ equal to that of the retrieved parameters.
- Evaluate K_4 candidates using the surviving 2^{96} K_5 candidates from the previous step. Now for all the 2^{96} (K_5, K_4) candidates, compute x_3 and filter them using the retrieved $x_3[0 \cdots 3], x_3[12 \cdots 15]$. It is expected that 2^{32} (K_5, K_4) candidates survive this stage.
- For all the possible 2^{128} values of K_3 and all the 2^{32} (K_5, K_4) candidates, evaluate x_2. About 2^{32} (K_5, K_4, K_3) candidates are expected to produce values of x_2 equal to that of the retrieved parameters.
- Evaluate K_2 values from the surviving candidates of K_3. Then for all the 2^{32} (K_5, K_4, K_3, K_2) remaining candidates, compute x_1. It is expected that only one (K_5, K_4, K_3, K_2) value results in $x_1[8 \cdots 15]$ equal to the retrieved parameters.
- Finally, for all the 2^{128} possible values of (K_1, K_0), partially decrypt the correct x_1 that was evaluated in the previous step and compare the result with the plaintext corresponding to the ciphertext used in the procedure. We expect one (K_1, K_0) candidate to pass this filtering stage which leaves us with one value for all round keys that decrypt a ciphertext to its corresponding plaintext.

Recovering the remaining round keys does not affect the data complexity of the attack. However, the memory complexity is slightly increased since we now need to store an additional 36 bytes in each entry. Accordingly, the memory complexity of the attack is given by $2^{200} \times (512 + 36 \times 8)/128 = 2^{202.64}$ 128-bit blocks. The time complexity of this stage is evaluated by $2^{128} \times 1/7 + 2^{96} \times 2/7 +$

$2^{160} \times 3/7 + 2^{32} \times 4/7 + 2^{128} \times 1/7 \approx 2^{159}$ reduced Kalina-128/256 encryptions. Consequently, the online time complexity of the whole attack is dominated by the last two rounds key recovery stage which is given by $2^{230.2}$ encryptions.

4 Analysis of the attack on Kalyna-256/512

Our MitM attack can also be applied on Kalyna-256/512 where the state has four columns. The differential path used in the attack is depicted in Figure 4. The attack steps are similar to the steps on Kalyna-128/256 except the precomputation phase. As shown in Figure 4, due to the ShiftRow operation, the distinguisher ends in two active bytes (instead of four for Kalyna-128/256). Accordingly, we only need 21 byte parameters which are $\Delta y_0[31]$, $x_1[16 \cdots 23]$, $x_3[0,1]$, $x_3[26,27]$, $x_3[20,21]$, $x_3[14,15]$, $x_4[0,1]$, and $\Delta z_4[0,1]$. We assume the independence of ordered sequences generated by individual bytes and thus their corresponding multisets. Accordingly, we store two multisets of differences in the two active bytes of z_4 so that the probability of false matches is very low. Hence, for the path depicted in Figure 4, the memory requirement for recovering the last two round keys is dominated by the precomputation table needed to store $2^{21 \times 8}$ two multisets, each of 512 bits. Thus, the memory complexity of the attack is $2^{168} \times 1024/256 = 2^{170}$ 256-bit blocks. The data complexity is determined according to the path probability which is equal to 2^{-240}. Accordingly, we need 2^{240} plaintexts pairs from 2^{225} structures where each structure consists of 2^8 chosen plaintexts. Thus the data complexity of the attack is given by $2^{225+8=233}$ chosen plaintexts. The time complexity of the offline phase is due to performing 2^{168} partial encryptions on 256 messages, which is equivalent to $2^{168+8} \times 5/7 \approx 2^{176}$ encryptions. The time complexity of the online phase is given by $2^{240+256+8} \times 2/7 \approx 2^{502.2}$. The probability of a false match is given by $2^{168+240+256-467.6\times 2} = 2^{-271.2}$ which is negligible.

5 Conclusion

In this work, we have presented a MitM attack on the new Ukranian standard encryption algorithm Kalyna. Our attack targets the 7-round reduced versions of the standard where the key size is double the block length. According to the security analysis performed by the designers of the cipher, Kalyna is resistant to various cryptanalytic methods after rounds five and six of the 128-bit and 256-bit block versions, respectively. In our attack, we construct our distinguisher such that the effect of the carry propagation is avoided after the modular key addition which improves the path probability and accordingly both the data and time complexities are reduced. Finally, due to the infeasibility of recovering the master key from the round keys, we exploit the linear relation between consecutive even and odd indexed round keys to recover all the round keys efficiently. Our results are considered the first steps towards the public cryptanalysis of the new Ukrainian encryption standard.

References

1. Government committee of the USSR for standards. GOST 28147-89. State standard of the USSR, information processing systems, cryptographic protection, algorithm of cryptographic transformation, 1999. (*in Russian*).
2. State service of special communication and information protection of Ukraine. statement on public competition of cryptographic algorithms, 2006. `http://www.dstszi.gov.ua/dstszi/control/ua/publish/printablearticle?art\_id=48387` (*in Ukrainian*).
3. Abdelkhalek, A., AlTawy, R., Tolba, M., and Youssef, A. M. Meet-in-the-middle attacks on reduced-round Hierocrypt-3. In *Latimcrypt* (2015), LNCS, Springer. (*to appear*).
4. AlTawy, R., and Youssef, A. M. A meet in the middle attack on reduced round Kuznyechik. Cryptology ePrint Archive, Report 2015/096, 2015. `http://eprint.iacr.org/`.
5. Daemen, J., and Rijmen, V. AES proposal: Rijndael, 1998.
6. Demirci, H., and Selçuk, A. A meet-in-the-middle attack on 8-round AES. In *FSE* (2008), K. Nyberg, Ed., vol. 5086 of *LNCS*, Springer, pp. 116–126.
7. Demirci, H., Taşkn, I., Çoban, M., and Baysal, A. Improved meet-in-the-middle attacks on AES. In *INDOCRYPT* (2009), B. Roy and N. Sendrier, Eds., vol. 5922 of *LNCS*, Springer, pp. 144–156.
8. Derbez, P., Fouque, P.-A., and Jean, J. Improved key recovery attacks on reduced-round AES in the single-key setting. In *EUROCRYPT* (2013), T. Johansson and P. Nguyen, Eds., vol. 7881 of *LNCS*, Springer, pp. 371–387.
9. Diffie, W., and Hellman, M. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer 10*, 6 (1977), 74–84.
10. Dunkelman, O., Keller, N., and Shamir, A. Improved single-key attacks on 8-round AES-192 and AES-256. In *ASIACRYPT* (2010), M. Abe, Ed., vol. 6477 of *LNCS*, Springer, pp. 158–176.
11. Guo, J., Jean, J., Nikolić, I., and Sasaki, Y. Meet-in-the-middle attacks on generic Feistel constructions. In *ASIACRYPT* (2014), P. Sarkar and T. Iwata, Eds., vol. 8873 of *LNCS*, Springer, pp. 458–477.
12. Hao, Y., Bai, D., and Li, L. A meet-in-the-middle attack on round-reduced mCrypton using the differential enumeration technique. In *Network and System Security* (2014), M. Au, B. Carminati, and C.-C. Kuo, Eds., vol. 8792 of *LNCS*, Springer, pp. 166–183.
13. Li, L., Jia, K., and Wang, X. Improved single-key attacks on 9-round AES-192/256. In *FSE* (2015), C. Cid and C. Rechberger, Eds., vol. 8540 of *LNCS*, Springer, pp. 127–146.
14. Mendel, F., Rechberger, C., Schläffer, M., and Thomsen, S. S. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE* (2009), O. Dunkelman, Ed., vol. 5665 of *LNCS*, Springer, pp. 260–276.
15. Oliynykov, R., Gorbenko, I., Kazymyrov, O., Ruzhentsev, V., Gorbenko, Y., and Dolgov, V. A new encryption standard of Ukraine: The block cipher "Kalyna" (DSTU 7624:2014), 2015. Online presntation: `http://www.slideshare.net/oliynykov/kalyna-english`, accessed: 25-July-2015.
16. Oliynykov, R., Gorbenko, I., Kazymyrov, O., Ruzhentsev, V., Kuznetsov, O., Gorbenko, Y., Dyrda, O., Dolgov, V., Pushkaryov, A., Mordvinov, R., and Kaidalov, D. DSTU 7624:2014. National standard of Ukraine. Information technologies. Cryptographic data security. Symmetric block transformation algorithm. Ministry of economical development and trade of Ukraine, 2015.

17. Oliynykov, R., Gorbenko, I., Kazymyrov, O., Ruzhentsev, V., Kuznetsov, O., Gorbenko, Y., Dyrda, O., Dolgov, V., Pushkaryov, A., Mordvinov, R., and Kaidalov, D. A new encryption standard of Ukraine: The Kalyna block cipher. Cryptology ePrint Archive, Report 2015/650, 2015. http://eprint.iacr.org/.

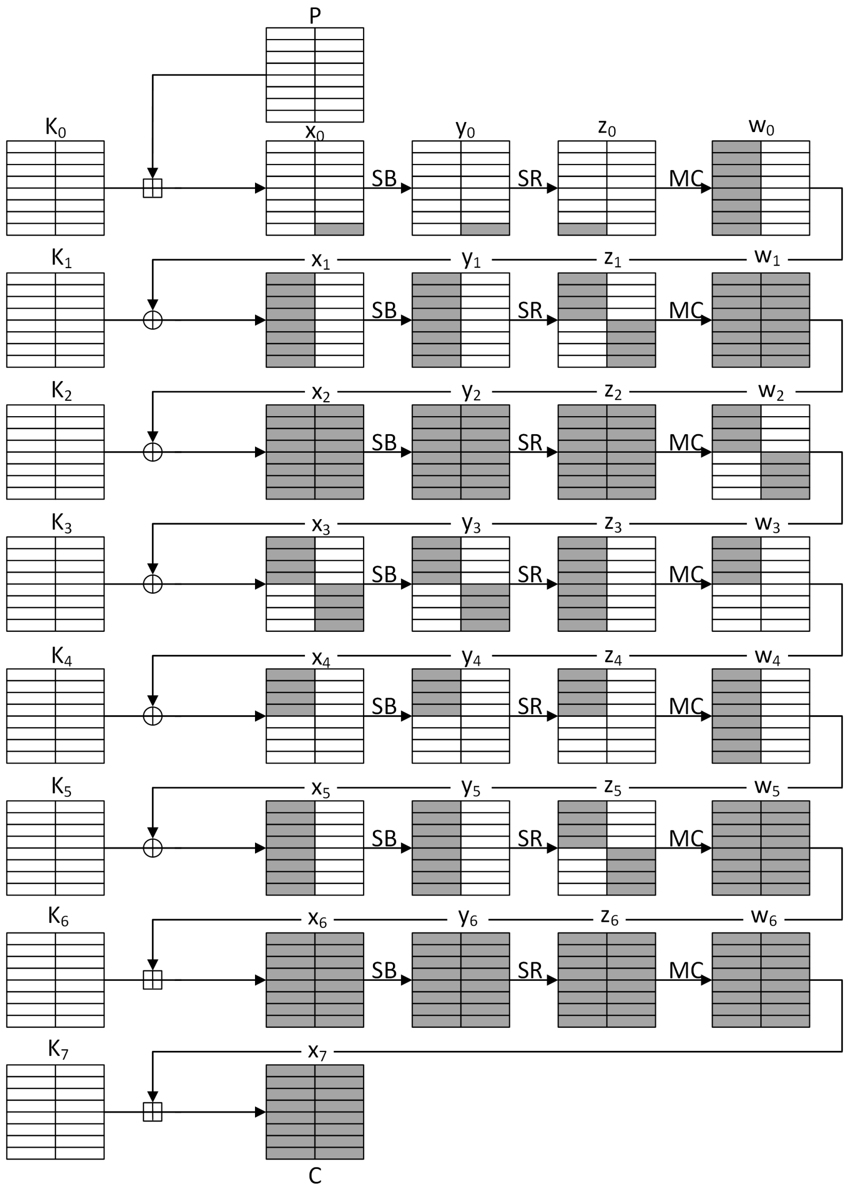

Fig. 3. Differential path used in the attack on the 7-round reduced Kalyna-128/256.

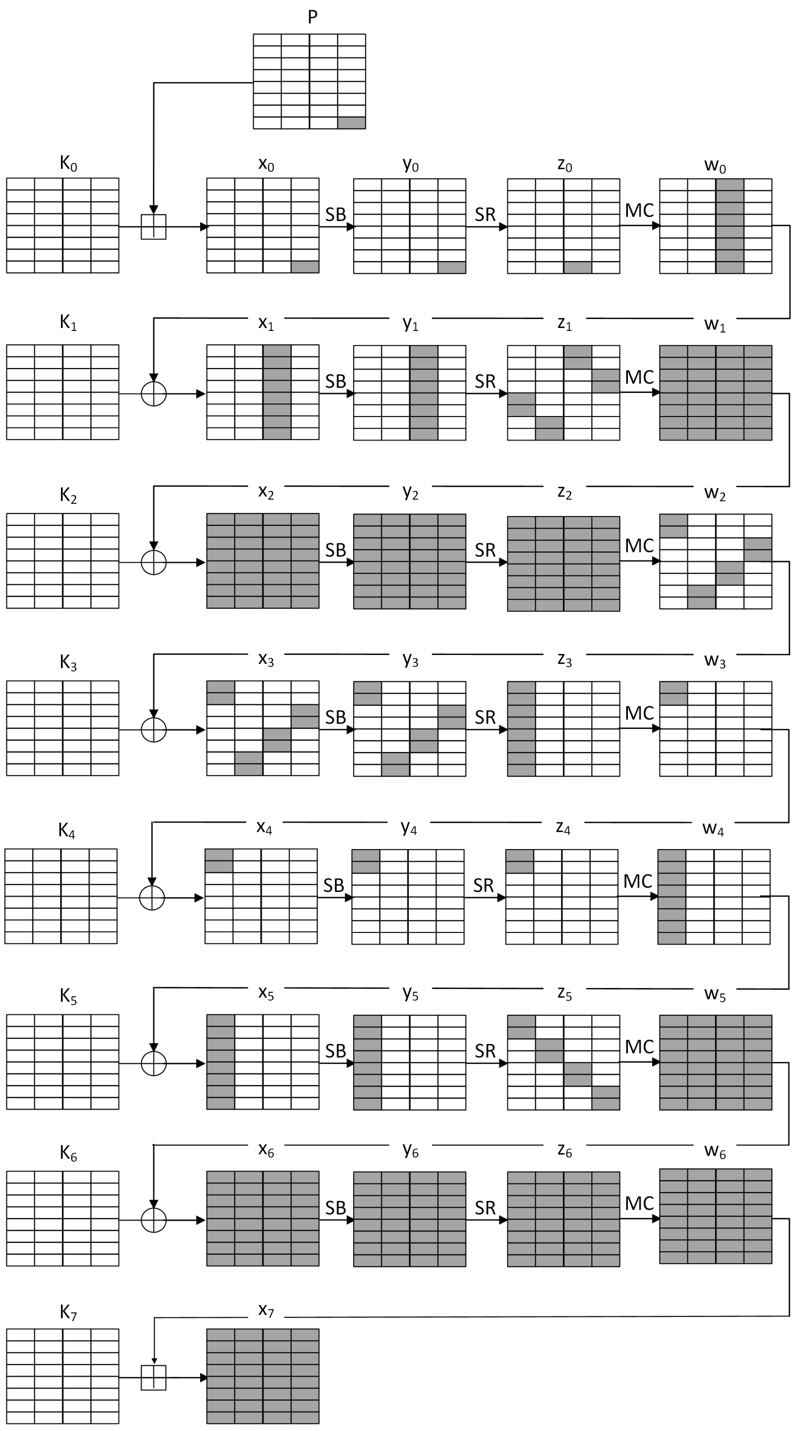

Fig. 4. Differential path used in the attack on the 7-round reduced Kalyna-256/512.