Improved Meet-in-the-Middle Attacks on Reduced-Round Kalyna-128/256 and Kalyna-256/512

Li Lin and Wenling Wu

Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
{linli, wwl}@tca.iscas.ac.cn

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. In this paper, we focus on the key-recovery attacks on reduced-round Kalyna-128/256 and Kalyna-256/512 with meet-in-the-middle method. The differential enumeration technique and key-dependent sieve technique which are popular to analyze AES are used to attack them. Using the key-dependent sieve technique to improve the complexity is not an easy task, we should build some tables to achieve this. Since the encryption procedure of Kalyna employs a pre- and post-whitening operations using addition modulo 2^{64} applied on the state columns independently, we carefully study the propagation of this operation and propose an addition plaintext structure to solve this. For Kalyna-128/256, we propose a 6-round distinguisher, and achieve a 9-round (out of total 14-round) attack. For Kalyna-256/512, we propose a 7-round distinguisher, then achieve an 11-round (out of total 18-round) attack. As far as we know, these are currently the best results on Kalyna-128/256 and Kalyna-256/512.

Keywords: Block Cipher, Kalyna, Meet-in-the-Middle Attack, Differential Enumeration Technique, Key-Bridging Technique.

1 Introduction

Block cipher Kalyna [14,13] was selected during Ukrainian National Public Cryptographic Competition (2007-2010) and its slight modification was approved as the new encryption standard DSTU 7624:2014 of Ukraine in 2015. Kalyna-b/k has five variants, i.e, Kalyna-128/128, Kalyna-128/256, Kalyna-256/256, Kalyna-256/512 and Kalyna-512/512, where b is the block size and k is the key size.

In recent years, meet-in-the-middle attack is widely researched due to its effectiveness against block cipher AES [3]. At FSE 2008, Demirci and Selçuk give meet-in-the-middle attacks on AES based on δ-set (a set of plaintexts where one byte can take all the 256 different values and the other bytes remain constant). More specifically, the value of each byte after 4-round AES encryption is a function of the δ-set parameterized by 25 [5] and 24 [6] byte-parameters. They use this function to build a distinguisher in the precomputation phase, i.e., store all the sequences constructed from the δ-set in a

lookup table. In the online phase, a δ-set must be identified from the plaintexts, and then partially decrypt the δ-set through the last few rounds and test whether it is in the table. To solve the memory problems of the Demirci and Selçuk attacks, Dunkelman et al. propose many contributions at ASIACRYPT 2010 [9]. First of all, they use multiset, i.e., an unordered sequence that elements can occur many times, instead of the ordered sequence. Their major contribution is the differential enumeration technique which is based on a special property of a truncated differential trail that greatly reduce the number of byte-parameters of Demirci and Selçuk attack from 24 to 16. At EUROCRYPT 2013, Derbez et al. propose a rebound-like idea called efficient tabulation to improve the differential enumeration technique [7]. They find that the precomputation table cannot take all the possible values under the constraint of a special truncated differential trail. Actually, the number of byte-parameters is reduced to 10. After that, a new improvement is introduced by Li et al. at FSE 2014, called key-dependent sieve technique [11]. With this technique, some unreachable values in the precomputation table can be filtered by some relations of sub-keys. In [2], 7-round meet-in-the-middle attacks on Kalyna-128/256 and Kalyna-256/512 are given by AlTawy et al. And in [1], Akshima et al. give 9-round meet-in-the-middle attacks on Kalyna-128/256 and Kalyna-256/512.

Our contributions. In this paper, we focus on the key-recovery attacks on reduced-round Kalyna-128/256 and Kalyna-256/512 with meet-in-the-middle method. First, a 6-round distinguisher on Kalyna-128/256 is presented using the key-dependent sieve technique and differential enumeration technique. By adding three rounds at the end, a 9-round meet-in-the-middle attack on Kalyna-128/256 is introduced with time complexity of $2^{208.38}$ 9-round Kalyna-128/256 encryptions, memory complexity of 2^{204} 128-bit blocks and data complexity of 2^{127} chosen-plaintexts. After that, with a 7-round distinguisher and an addition plaintext structure (the usual structure is tweaked to better fit the modular addition), an attack on 11-round Kalyna-256/512 is introduced by adding one round at the beginning and three rounds at the end with time complexity of 2^{425} 11-round Kalyna-256/512 encryptions, memory complexity of $2^{403.5}$ 256-bit blocks and data complexity of 2^{233} chosen-plaintexts. As far as we know, these are currently the best attack results on Kalyna-128/256 and Kalyna-256/512. We present here a summary of our attack results on Kalyna-128/256 and Kalyna-256/512, and compare them to the best attacks known for them. This summary is given in Table 1.

Table 1. Summary of the best attacks on Kalyna-128/256 and Kalyna-256/512.

Cipher	Attack type	Rounds	Data	Memory (Blocks)	Time (Enc)	Source
Kalyna-128/256	MITM	7	2^{89} CPs	$2^{202.64}$	$2^{230.2}$	[2]
	MITM	9	2^{105} CPs	$2^{226.86}$	$2^{245.83}$	[1]
	MITM	9	2^{127} CPs	2^{204}	$2^{208.38}$	Sec. 3
Kalyna-256/512	MITM	7	2^{233} CPs	2^{170}	$2^{502.2}$	[2]
	MITM	9	2^{217} CPs	$2^{451.45}$	$2^{477.82}$	[1]
	MITM	11	2^{233} CPs	$2^{403.5}$	2^{425}	Sec. 4

CPs: Chosen-Plaintexts.

Organizations of this paper. In Section 2, we describe Kalyna with some definitions and properties used throughout this paper, and then review the former works of meet-in-the-middle attack. In Section 3, we give our attack on 9-round Kalyna-128/256. In Section 4, we give our attack on 11-round Kalyna-256/512. In Section 5, we conclude this paper.

2 Preliminaries

In this section we give a short description of Kalyna with some definitions and propositions used throughout this paper. After that, we review the former works of meet-in-the-middle attack. Finally, the attack scheme is given.

2.1 Description of Kalyna

The encryption procedures of Kalyna-128/256 and Kalyna-256/512 run a round function for 14 and 18 times on a byte matrix of size 8×2 and 8×4 respectively, and the bytes are numbered column-wise. As shown in Fig. 1, the encryption procedure employs a pre- and post-whitening operations using addition modulo 2^{64} applied on the state columns independently. Each round function consists of four basic operations:

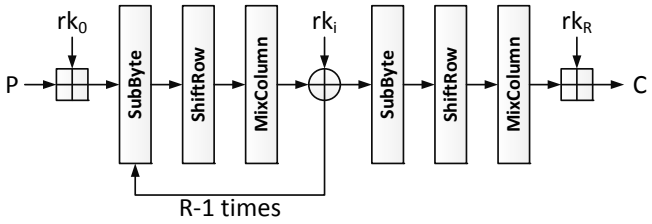

Fig. 1. Kalyna encryption function. Here, R is the number of rounds.

- **SubByte (SB):** Apply the non-linear 8×8 S-boxes in parallel on each byte of the state.
- **ShiftRow (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 denote the row number and state size, respectively.
- **MixColumn (MC):** A transformation that multiplies the columns of the state independently with an 8×8 MDS matrix.
- **AddRoundKey (AK):** The i^{th} b-bit round key rk_i is xored to the state.

The key-schedule of Kalyna is quite complicated, we refer to [14] for the details. We only give one important property of the key-schedule, i.e., odd indexed sub-keys are linearly computed from their previous even indexed sub-keys according to the formula:

$$rk_i = rk_{i-1} \lll (b/4 + 24),$$

where $\lll$ denotes circular left shift operation.

In this paper, let x_i denote the internal state entering Round i, and let y_i, z_i and w_i denote the internal states after the SubByte, ShiftRow and MixColumn operations of Round i, respectively. Let $x_i[j]$ denote the j^{th} byte of Round i, $x_i[j.w]$ denote the w^{th} bit of $x_i[j]$, $x_i[j_0 - j_1]$ denote the j_0^{th} byte to the j_1^{th} byte of x_i and $x_i^k[j]$ denote the k^{th} element of a set of some $x_i[j]$. Let $\Delta x_i^k[j]$ denote the difference of the k^{th} element and 0^{th} element of a set, i.e., $\Delta x_i^k[j] = x_i^k[j] \oplus x_i^0[j]$.

We denote the sub-key of Round i by rk_{i+1}. Since MixColumn and AddRoundKey are linear operations, we can firsty xor the data with an equivalent sub-key $ru_i = MC^{-1}(rk_i)$ and then apply the MixColumn operation.

2.2 Definitions and Propositions

First of all, we give some definitions of particular structures of messages used in the attacks.

Definition 1 (δ-set of byte, [3]). *Let a δ-set be a set of 2^8 states that are all different in one state byte (active byte) and all equal in the other state bytes (inactive bytes).*

Definition 2 (Multiset of bytes, [7]). *A multiset generalizes the set concept by allowing elements to appear more than once. Here, a multiset of 256 bytes can take as many as $\binom{2^8+2^8-1}{2^8} \approx 2^{506.17}$ different values.*

Definition 3 (2-Multiset, [12]). *A 2-multiset is a multiset of 256 values where each value consists of two bytes. A 2-multisets can take as many as $\binom{2^{16}+2^8-1}{2^{16}-1} \approx 2^{2412.72}$ different values.*

In [4], Daemen et al. gave the definition of Super-box for AES. For Kalyna, we can give a similar definition as follows.

Definition 4 (Super-box). *For each value of one column of rk_2, a Kalyna Super-box maps one column of z_1 to one column of y_2 as shown in Fig. 2. It consists of one SB operation, one MC operation, one AK operation and one SB operation.*

For one S-box, we have the following proposition.

Proposition 1 (Differential Property of S-box, [7]). *Given Δ_i and Δ_0 two non-zero differences, the equation of S-box*

$$S(x) \oplus S(x \oplus \Delta_i) = \Delta_0, \tag{1}$$

has one solution in average.

This proposition also applies to Super-box.

Proposition 2 (Differential Property of Super-box). *Given Δ_i and Δ_0 two non-zero differences in $F_{2^{64}}$, the equation of Super-box*

$$Super - S(x) \oplus Super - S(x \oplus \Delta_i) = \Delta_0, \tag{2}$$

has one solution in average for each key value.

For ru_i, we have the following proposition.

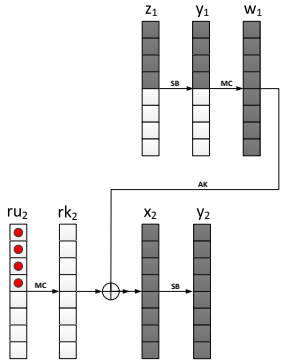

Fig. 2. Super-box for Kalyna.

Proposition 3. *As shown in Fig. 2, if the first column of z_1 is active only in the first 4 bytes, $z_1[0-3]||y_2[0-7]$ has one solution in average for each $\Delta z_1[0-3]||\Delta y_2[0-8]||ru_2[0-3]$.*

Proof. We use the equivalent sub-key in this proof. For each $y_2[0-7]$ and $ru_3[0-3]$, since $\Delta y_4[0-7]$ is known, one can get $\overline{w_1}[0-7]$ and $\Delta\overline{w_1}[0-7]$. With the probability of 2^{-32}, $y_1[0-7]$ is active only in the first four bytes. By adding $ru_2[0-3]$, one can get $\Delta z_1[0-3]$.

Therefore, for each Δ_i and Δ_0, the average number of input values of Super-box is $2^{64-32-32} = 1$ for each equivalent sub-key.

□

2.3 Reviews of Former Works

In this section, we review the previous meet-in-the-middle distinguishers on AES in [5,9,7,11].

Demirci and Selçuk distinguisher. Consider the set of functions

$$f : \{0,1\}^8 \longrightarrow \{0,1\}^8$$

that maps a byte of a δ-set to another byte of the state after four AES rounds. A convenient way is to view f as an ordered byte sequence $(f(0),\ldots,f(255))$ so that it can be represented by 256 bytes. This set is tiny since it can be described by 25 byte-parameters $(2^{25\cdot 8} = 2^{200})$ compared with the set of all functions of this type which counts as may as $2^{8\cdot 2^8} = 2^{2048}$ elements [5]. Considering the differences $(f(0)-f(0), f(1)-f(0),\ldots,f(255)-f(0))$ rather than values, the set of functions can be described by 24 byte-parameters [6]. The 24 byte-parameters which map $x_1[0]$ to $\Delta x_5[0]$ are presented as gray cells in Fig. 3.

Dunkelman et al. distinguisher and Derbez et al. distinguisher. In [9], Dunkelman et al. introduced two new improvements to further reduce the memory complexity of [6]. The first uses *multiset* which is an unordered sequence with multiplicity to replace ordered sequence in the offline phase, since there is enough information so that the attack

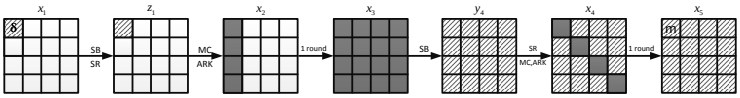

Fig. 3. The 4-round AES distinguisher used in [6]. The gray cells represent 24 byte-parameters, δ represents the δ-set and m represents the differential sequence to be stored.

succeeds. The second improvement uses a novel idea named **differential enumeration technique**. The main idea of this technique is to use a special 4-round property on a truncated differential trail to reduce the number of parameters which describes the set of functions from 24 to 16.

In [7], Derbez et al. used the efficient tabulation to improve Dunkelman et al.'s differential enumeration technique. Combining with the rebound-like idea, many values in the precomputation table are not reached at all under the constraint of a truncated differential trail.

Proposition 4 (Differential Enumeration Technique with Efficient Tabulation, [7]). *If a message of δ-set belongs to a pair conforming to the 4-round truncated differential trail outlined in Fig. 4, the values of multiset are only determined by 10 byte-parameters of intermediate states $\Delta z_1[0]||x_2[0,1,2,3]||\Delta x_5[0]||z_4[0,1,2,3]$ presented as gray cells in Fig. 4.*

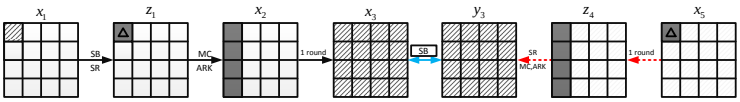

Fig. 4. The truncated differential trail of 4-round AES used in [5], the gray cells represent 10 byte-parameters, Δ represents difference.

The main idea of their works is that suppose one gets a pair of messages conforming to this truncated differential trail, Δx_3 is determined by $\Delta z_1[0]||x_2[0,1,2,3]$ and Δy_3 is determined by $\Delta x_5[0]||z_4[0,1,2,3]$. By Proposition 1, part of the 24 byte-parameters in the Demirci and Selçuk distinguisher, i.e. x_3, can be determined. Therefore, the number of parameters which determines the size of the precomputation table reduces from 16 to 10. In the rest of this paper, when we speak of differential enumeration technique, we mean differential enumeration technique with efficient tabulation.

Key-dependent sieve technique. At FSE 2014, Li et al. introduced the key-dependent sieve technique, which filters the wrong states based on the key relations, to further reduce the complexity in the precomputation phase of the attack on AES-192 [11]. More specifically, as shown in Fig. 5, the precomputation procedure allows to deduce $ru_2[3,6,9,12]$ and rk_3, independently. Meanwhile, by the key schedule of AES-192, it is obviously that the knowledge of rk_3 allows to deduce columns 0 and 1 of rk_2. This means that the value of the equivalent sub-key $ru_2[3,6]$ can be deduced from rk_3.

Thus there exists a contradiction between $ru_2[3,6]$ and rk_3 with a probability of 2^{-16}. Therefore, the size of precomputation table is improved by a factor of 2^{16}.

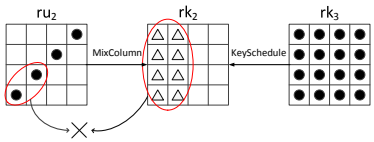

Fig. 5. Key-dependent sieve technique based on the key schedule algorithm of AES-192

2.4 Attack Scheme

In this section, we present a unified view of the meet-in-the-middle attack, where R rounds of block cipher can be split into three consecutive parts: r_1, r, and r_2, such that a particular set of messages may verify a certain property we denote ★ in the sequel in the middle r rounds as shown in Fig. 6.

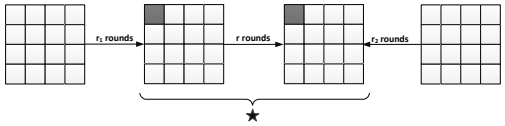

Fig. 6. General scheme of meet-in-the-middle attack, where some messages in the middle rounds may verify a certain ★ property used to perform the meet-in-the-middle method.

The general attack scheme uses two successive phases:

Precomputation phase

1. In the precomputation phase, we build a lookup table T containing all the possible sequences constructed from a δ-set such that one message verifies a truncated differential trail.

Online phase

2. In the online phase, we need to identify a δ-set containing a message m verifying the desired property. This is done by using a large number of plaintexts and ciphertexts, and expecting that for each key candidate, there is one pair of plaintexts satisfying the truncated differential trail.
3. Finally, we partially decrypt the associated δ-set through the last r_2 rounds and check whether it belongs to T.

3 Meet-in-the-Middle Attack on 9-Round Kalyna-128/256

In this section, we first propose a 6-round meet-in-the-middle distinguisher with differential enumeration technique and key-dependent sieve technique on Kalyna-128/256. Then, we apply this distinguisher to 9-round Kalyna-128/256 by adding 3 rounds at the end.

3.1 6-Round Distinguisher on Kalyna-128/256

Since the branch number of MC operation is 9, if the differences in any eight out of sixteen input/output bytes are known, then the differences in the other eight bytes are uniquely determined. If $z_5[4-7]$ and $w_5[4-7]$ are known, $w_5[3] = 0x67 \cdot z_5[4] \oplus 0x80 \cdot z_5[5] \oplus 0x90 \cdot z_5[6] \oplus 0x20 \cdot z_5[7] \oplus 0x6a \cdot w_5[4] \oplus 0x1b \cdot w_5[5] \oplus 0xde \cdot w_5[6] \oplus 0xeb \cdot w_5[7]$. Let $e_{in} = 0x67 \cdot z_5[4] \oplus 0x80 \cdot z_5[5] \oplus 0x90 \cdot z_5[6] \oplus 0x20 \cdot z_5[7]$ and $e_{out} = x_6[3] \oplus 0x6a \cdot x_6[4] \oplus 0x1b \cdot x_6[5] \oplus 0xde \cdot x_6[6] \oplus 0xeb \cdot x_6[7]$, then $\Delta e_{out} = \Delta e_{in}$.

Besides, since $rk_3 = rk_2 \lll 56$ and $ru_2[0] = 0xad \cdot rk_2[0] \oplus 0x95 \cdot rk_2[1] \oplus 0x76 \cdot rk_2[2] \oplus 0xa8 \cdot rk_2[3] \oplus 0x2f \cdot rk_2[4] \oplus 0x49 \cdot rk_2[5] \oplus 0xd7 \cdot rk_2[6] \oplus 0xca \cdot rk_2[7]$, we have $ru_2[0] = 0xad \cdot rk_3[9] \oplus 0x95 \cdot rk_3[10] \oplus 0x76 \cdot rk_3[11] \oplus 0xa8 \cdot rk_3[12] \oplus 0x2f \cdot rk_3[13] \oplus 0x49 \cdot rk_3[14] \oplus 0xd7 \cdot rk_3[15] \oplus 0xca \cdot rk_3[0]$. Let χ_0 denote $0xad \cdot x_3[9] \oplus 0x95 \cdot x_3[10] \oplus 0x76 \cdot x_3[11] \oplus 0xa8 \cdot x_3[12] \oplus 0x2f \cdot x_3[13] \oplus 0x49 \cdot x_3[14] \oplus 0xd7 \cdot x_3[15] \oplus x_3[0]$ and χ'_0 denote $0xad \cdot w_2[9] \oplus 0x95 \cdot w_2[10] \oplus 0x76 \cdot w_2[11] \oplus 0xa8 \cdot w_2[12] \oplus 0x2f \cdot w_2[13] \oplus 0x49 \cdot w_2[14] \oplus 0xd7 \cdot w_2[15] \oplus w_2[0] \oplus ru_2[0]$, we can deduce $\chi_0 = \chi'_0$. Similarly, we can define χ_i and χ'_i for $ru_2[i]$, where $i = 1, 2, 3, 12, 13, 14, 15$. Therefore, we can define χ and χ' as follows:

$$\begin{aligned} \chi &= (\chi_0, \chi_1, \chi_2, \chi_3, \chi_{12}, \chi_{13}, \chi_{14}, \chi_{15}) \\ \chi' &= (\chi'_0, \chi'_1, \chi'_2, \chi'_3, \chi'_{12}, \chi'_{13}, \chi'_{14}, \chi'_{15}). \end{aligned} \tag{3}$$

The 6-round distinguisher on Kalyna-128/256 is based on the proposition below.

Proposition 5. *If a message m belongs to a pair of states conforming to the truncated differential characteristic of Fig. 7(a), then the multiset of differences Δe_{out} obtained from the δ-set constructed from m in x_0, where $x_0[15]$ is the active byte, can only take about 2^{216} values.*

Proof. As shown in Fig. 7(a), for the encryption of the *δ-set*, the output multiset is determined by the 52 byte-parameters:

$$x_1^0[0-7] || x_2^0 || rk_3 || rk_4[4-11] || rk_5[12-15] \tag{4}$$

At Round 0, the differences $(x_0^0[15] \oplus x_0^0[15], x_0^1[15] \oplus x_0^0[15], \cdots, x_0^{255}[15] \oplus x_0^0[15])$ are known — these are exactly the 256 possible differences in byte 15 (the rest of the bytes are equal). Note that the order of the differences are not known, but this does not disturb the adversary since in our attack he is interested only in the multiset and not in the sequence. Since the SR, MC and AK operations are linear, $\Delta x_1^m[0-7]$ is also known, where $m = 0, 1, \cdots, 255$. Since we have guessed the value of $x_1^0[0-7]$, we can deduce $\Delta y_1^m[0-7] = S(x_1^0[0-7]) \oplus S(x_1^0[0-7] \oplus \Delta x_1^m[0-7])$. And since the SR, MC and

AK operations are linear, Δx_2^m can be deduced. Similarly, Δx_3^m can be deduced by the knowledge of x_2^0. Since x_3^0 can be deduced from x_2^0 and rk_3, $\Delta x_4^m[4-11]$ can be deduced by the knowledge of rk_3. Since $x_4^0[4-11]$ can be deduced from x_3^0 and $rk_4[4-11]$, $\Delta x_5^m[12-15]$ can be deduced by the knowledge of $rk_4[4-11]$. Since $x_5^0[12-15]$ can be deduced from $x_4^0[4-11]$ and $rk_5[12-15]$, $\Delta z_5^m[4-7]$ can be deduced by the knowledge of $rk_5[12-15]$. Then we get the value of $e_{in}^m \oplus e_{in}^0$. Since $e_{out}^m \oplus e_{out}^0 = e_{in}^m \oplus e_{in}^0$, we can get the corresponding multiset of Δe_{out}.

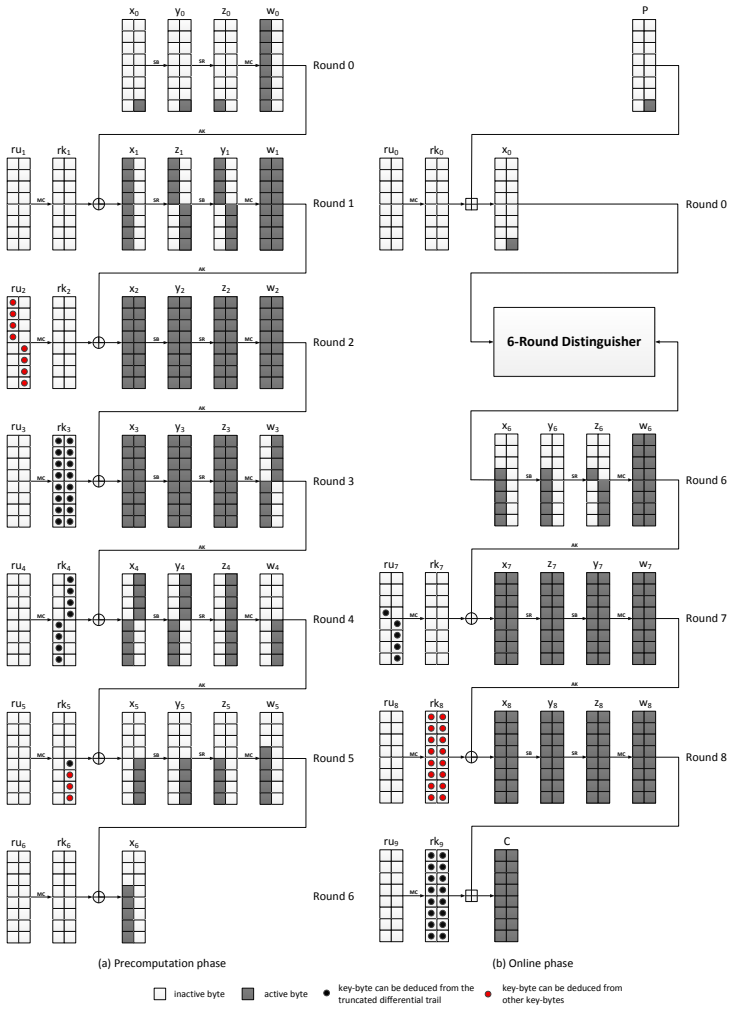

Fig. 7. The attack on 9-round Kalyna-128/256. The 6-round distinguisher is shown in (b), the online phase is shown in (b).

However, if a pair of messages (m_0, m_1) conforms to the truncated differential trail outlined in Fig. 7(a), the above 52 byte-parameters are determined by the 38 byte-parameters (the values are corresponding to m_0 and the differences are corresponding to $m_0 \oplus m_1$):

$$\Delta z_0[7]||x_1[0-7]||y_3||y_4[4-11]||y_5[12-15]||\Delta z_5[4] \tag{5}$$

Since $\Delta z_0[7]$ is known, we can get $\Delta x_1[0-7]$. Since $\Delta y_1[0-3,12-15]$ can be deduced by the knowledge of $x_1[0-7]$, we can deduce Δx_2. For the backward direction, since $\Delta z_5[0,1,2,3] = \Delta w_5[0,1,2] = 0$, we can deduce $\Delta z_5[5] = 0x13 \cdot \Delta z_5[4]$, $\Delta z_5[6] = 0x30 \cdot \Delta z_5[4]$ and $\Delta z_5[7] = 0xb2 \cdot \Delta z_5[4]$. So we can get $\Delta z_5[4-7]$. For the same reason as the forward direction, Δy_2 can be deduced by the knowledge of $y_5[12-15]||y_4[4-11]||y_3||\Delta z_5[4-7]$. According to Proposition 1, we get one value of intermediate state $x_2||y_2$ on average for the fixed difference $\Delta x_2||\Delta y_2$. Apparently, $ru_2[0-3,12-15]||rk_3||rk_4[4-11]||rk_5[12-15]$ is also deduced for every 38 byte-parameters (by letting m_0 be corresponding to the 0^{th} element of the δ-set).

Since rk_3 can be deduced from x_2 and y_3 and $ru_2[0-3,12-15]$ can be deduced from $x_1[0-7]$ and x_2, this means that we can get rk_3 and $ru_2[0-3,12-15]$ independently. According to the key-schedule of Kalyna-128/256, $ru_2[0-3,12-15]$ can be deduced from rk_3. Thus there exists a contradiction between $ru_2[0-3,12-15]$ and rk_3 with a probability of 2^{-64}. Therefore, the size of precomputation table is improved by a factor of 2^{64}. Similarly, since we can get $rk_4[4,5,6]$ and $rk_5[13,14,15]$ independently and $rk_4[4,5,6]$ can be deduced from $rk_5[13,14,15]$ by the key-schedule, there exists a contradiction between $rk_4[4,5,6]$ and $rk_5[13,14,15]$ with a probability of 2^{-24}. Therefore, By the key-dependent sieve technique, there are 2^{216} possible values for the 38 byte-parameters. So the 52 byte-parameters (4) are determined by 38 byte-parameters (5), i.e., the multiset of Δe_{out} can take about 2^{216} values.

□

3.2 Attack on 9-Round Kalyna-128/256

The attack is made up of two phases: precomputation phase and online phase (we change the order of SB and SR at Round 1 and Round 7).

Precomputation phase: In the precomputation phase, we need to build a table T_3 that contains all the the multisets of Δe_{out} described in Propostion 5. To use the key-dependent sieve technique to improve the complexity, we need to build two more tables T_1 and T_2. Otherwise, we need to build two more tables T_4^1 and T_4^2 for online phase.

1. As shown in Fig. 7(a), for each value of $\Delta z_5[4]||\Delta w_4[12-15]||\Delta w_3[4-11]$: compute $\Delta z_5[4-7]$ since $\Delta z_5[0,1,2,3] = 0$ and $\Delta w_5[0,1,2] = 0$. Deduce $x_5[12-15]$ and $x_4[4-11]$ according to Proposition 1. Deduce $rk_5[12-15]$ from $x_5[12-15]$ and $x_4[4-11]$, then compute $w_3[4,5,6]$ from $x_4[4,5,6]$ and $rk_5[12,13,14]$. Store $rk_5[12-15]||x_4[7-11]$ in a table T_1 with the index of $w_3[4,5,6]||\Delta w_3[4-11]$. There are about 2^{16} values for each index.
2. For each value of $\Delta x_3||\Delta w_3[4-11]$, deduce $x_3||y_3$ according to Proposition 1. Deduce χ and $w_3[4-11]$ from x_3. Store $x_3||w_3[4-11]||\Delta w_3[4-11]$ in a table T_2 with the index of $\chi||\Delta w_2$. There is about 1 value for each index.

3. For each 64-bit value of $ru_2[0-3,12-15]$, do the following sub-steps.
   (a) For all 2^{136} values of $\Delta z_0[7]$ and Δw_2, deduce $\Delta z_1[0-3,12-15]$ and Δy_2. According to Proposition 3, we can get $z_1[0-3,12-15]$ and y_2. Deduce χ' from $ru_2[0-3,12-15]$ and y_2. Look up the table T_2 to get about one value of $x_3||w_3[4-11]||\Delta w_3[4-11]$ with the index of $\chi'||\Delta w_2$. Then look up the table T_1 to get about 2^{16} values of $rk_5[12-15]||x_4[7-11]$ with the index of $w_3[4,5,6]||\Delta w_3[4-11]$. Deduce $rk_4[4-11]$ from $rk_5[13,14,15]$, $x_4[7-11]$ and $w_3[7-11]$, and deduce rk_3 from x_3 and y_2. Therefore, we get the 52 byte-parameters (4).
   (b) For each of the 52 byte-parameters we get, compute the multiset of Δe_{out}, and store them in a table T_3.
4. We build two more tables T_4^1 and T_4^2 for online phase.
   (a) As shown in Fig. 7(b), for all 2^{168} values of $\Delta z_6[3,12,13,14,15]||\Delta y_8[0-7]||rk_8[0-7]$, deduce $\Delta z_7[0-7]$. Then deduce $z_7[0-7]||y_8[0-7]$ according to Proposition 2. Deduce $z_8[0-3,12-15]$ and $\Delta z_8[0-3,12-15]$, and store $z_8[0-3]||\Delta z_8[0-3]||rk_8[7]||x_7[0-7]||\Delta z_6[3,12,13,14,15]$ in a table T_4^1 with the index of $rk_8[0-6]||z_8[12-15]||\Delta z_8[12-15]$. There are 2^{48} values for each index.
   (b) For all 2^{168} values of $\Delta z_6[3,12,13,14,15]||\Delta y_8[8-15]||rk_8[8-15]$, deduce $\Delta z_7[8-15]$, and deduce $z_7[8-15]$ and $y_8[8-15]$ according to Proposition 2. Deduce $z_8[4-11]$ and $\Delta z_8[4-11]$, and store $z_8[8-11]||\Delta z_8[8-11]||rk_8[15]||z_7[8-15]||\Delta z_6[3,12,13,14,15]$ in a table T_4^2 with the index of $rk_8[8-14]||z_8[4-7]||\Delta z_8[4-7]$. There are 2^{48} values for each index.

Online phase: In the online phase of the attack, we first find at least one pair which satisfies the truncated differential trail in Fig. 7(a). To find the right pair, instead of guessing the sub-keys and checking whether this pair satisfies the truncated differential trail, we deduce the sub-keys which make it satisfy the truncated differential trail for each pair. Then we identify the δ-set, caculate the corresponding mutiset of Δe_{out} and check whether it belongs to the table T_3.

1. **Phase A – Detecting the right pair.**
   Define a structure of 2^8 plaintexts where $P[15]$ take all the possible values, and the remaining 15 bytes are fixed to some constants. Hence, we can generate $2^8 \times (2^8-1)/2 \approx 2^{15}$ pairs satisfying the plaintext difference. Choose 2^{105} structures to get about $2^{15+105} = 2^{120}$ pairs. As shown in Fig. 7(b), the probability to get the truncated differential trail in the forward and backward direction is $2^{(1-16)\times 8} = 2^{-120}$, then about 1 pair follows the truncated differential trail for each guess of the key.
2. **Phase B – Checking the δ-set.**
   For each of the 2^{120} remaining pairs, we do the following sub-steps.
   (a) Guess $rk_9[0-7]$, and deduce $rk_8[7-14]$, $z_8[0-7]$ and $\Delta z_8[0-7]$. Look up the table T_4^2 to get about 2^{48} values of $z_8[8-11]||\Delta z_8[8-11]||rk_8[15]||z_7[8-15]||\Delta z_6[3,12,13,14,15]$ with the index of $rk_8[8-14]||z_8[4-7]||\Delta z_8[4-7]$. Store $z_7[8-15]||rk_9[0-7]$ in a table T_5^1 with the index of $\Delta z_8[0-3,8-11]||z_8[0-3,8-11]||rk_8[7,15]||\Delta z_6[3,12,13,14,15]$. The size of this table is 2^{112}.

(b) Guess $rk_9[8-15]$, and deduce $rk_8[0-6,15]$, $z_8[8-15]$ and $\Delta z_8[8-15]$. Look up the table T_4^1 to get about 2^{48} values of $z_8[0-3]||\Delta z_8[0-3]||rk_8[7]||z_7[0-7]||\Delta z_6[3,12,13,14,15]$ with the index of $rk_8[0-6]||z_8[12-15]||\Delta z_8[12-15]$. Store $z_7[0-7]||rk_9[8-15]$ in a table T_5^2 with the index of $\Delta z_8[0-3,8-11]||z_8[0-3,8-11]||rk_8[7,15]||\Delta z_6[3,12,13,14,15]$. The size of this table is 2^{112}.

(c) Merge T_5^1 and T_5^2 with the index of $\Delta z_8[0-3,8-11]||z_8[0-3,8-11]||rk_8[7,15]||\Delta z_6[3,12,13,14,15]$ and get about 2^{40} values of $rk_9||z_7||\Delta z_6[3,12,13,14,15]$. Guess $\Delta x_6[3]$, and deduce $\Delta x_6[3-7]$ since $\Delta x_6[0,1,2]=0$ and $\Delta z_5[0-3]=0$. According to Proposition 1, we can deduce $z_6[3,12,13,14,15]$ from $\Delta x_6[3,4,5,6,7]$ and $\Delta z_6[3,12,13,14,15]$. Deduce $ru_7[3,12,13,14,15]$ from $z_6[3,12,13,14,15]$ and x_7. Therefore, we get about 2^{48} values of rk_9 (rk_8 can be deduced from rk_9) and $ru_7[3,12,13,14,15]$.

(d) Pick one member of the pair and get the δ-set where $P[15]$ is the active byte, then get the corresponding ciphertexts. For about 2^{48} values of rk_9, rk_8 and $ru_7[3,12,13,14,15]$, decrypt the ciphertexts through the last 3 rounds and get the multiset of Δe_{out}. Check whether it lies in the precomputation table T_3. If not, try another one.

3. **Exhaustively search the rest of the key:** Since the key-schedule algorithm of Kalyna does not allow recovery of master key from any subkey. As rk_9, rk_8 and $ru_7[3,12,13,14,15]$ have been recovered, we can recover all the sub-keys following the method in [2].

Complexity analysis. The probability for a wrong guess to pass this test is $2^{216-506.17} \approx 2^{-290.17}$ [7]. In the precomputation phase, in order to construct T_3, we need to perform 2^{216} partial encryptions on 2^8 messages. The time complexity of this phase is about $2^{216+8-1.62} = 2^{222.38}$ 9-round Kalyna-128/256 encryptions, the memory complexity is about $2^{216+2} = 2^{218}$ 128-bit blocks. In the online phase, the major time complexity comes from step 2a and 2b, so the time complexity of this phase is about $2^{120+64-3+1} = 2^{182}$ 9-round Kalyna-128/256 encryptions, the data complexity is $2^{8+105} = 2^{113}$ chosen-plaintexts. With data/time/memory tradeoff, the adversary only need to precompute a fraction of 2^{-14} of possible sequences, then the time complexity becomes $2^{222.38-14} = 2^{208.38}$, the memory complexity becomes 2^{204} 128-bit blocks. But in the online phase , the adversary will repeat the attack 2^{14} times to offset the probability of the failure. So the data complexity increases to 2^{127} chosen-plaintexts, and the time complexity increases to 2^{196}. In total, the time complexity of this attack is $2^{208.38}$ 9-round Kalyna-128/256 encryptions, the data complexity is 2^{127} chosen-plaintexts and the memory complexity is 2^{204} 128-bit blocks.

4 Meet-in-the-Middle Attack on 11-Round Kalyna-256/512

In this section, we first propose an addition plaintext structure and a 7-round meet-in-the-middle distinguisher with differential enumeration technique and key-dependent sieve technique on Kalyna-256/512. Then, we apply these to 11-round Kalyna-256/512 by adding one round at the beginning and three rounds at the end.

4.1 Addition Plaintext Structure

Considering the plaintext structure of meet-in-the-middle attacks on AES and other block ciphers, i.e., some bytes of the plaintexts take all the possible values while the other bytes remain constant, since the whiten key of Kalyna is modular added to the plaintext, this structure makes no sense. Here, we propose an addition plaintext structure to solve this as follows. For the second column of x_0, we have the following 2 cases (Fig. 8):

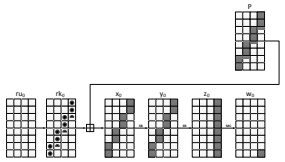

Fig. 8. Addition plaintext structure of Kalyna.

- If $rk_0[14.0] = 0$, we let $x_0[12,13]$ take all the possible values, the other bytes remain constant and the first bit of $x_0[14]$ be 1. After $x_0 \boxminus rk_0$, the second column of P has difference only in byte 12, byte 13 and the first bit of byte 14.
- If $rk_0[14.0] = 1$, we let $x_0[12,13]$ take all the possible values, the other bytes remain constant and the first bit of $x_0[14]$ be 0. After $x_0 \boxminus rk_0$, the second column of P has difference only in byte 12, byte 13 and the first bit of byte 14.

For the third and fourth columns of Fig. 8, we have the similar case as the second column. Therefore, the plaintext structure of Kalyna takes all the possible values of $P[6,7,12,13,$
$14.0,18,19,20.0,24,25,26.0]$.

4.2 7-Round Distinguisher on Kalyna-256/512

Since the branch number of MC operation is 9, if the differences in any eight out of sixteen input/output bytes are known, then the differences in the other eight bytes are uniquely determined. By the MC matrix, we have the following two equations:

$$\begin{aligned} z_7[2] \oplus 0xbb \cdot z_7[3] =& 0x3c \cdot w_7[0] \oplus 0x8d \cdot w_7[1] \oplus 0x49 \cdot w_7[2] \oplus 0x68 \cdot w_7[3] \\ & \oplus 0xbb \cdot w_7[4] \oplus 0x75 \cdot w_7[5] \oplus 0xbf \cdot w_7[6] \\ z_7[2] \oplus 0x2d \cdot z_7[3] =& 0x6f \cdot w_7[1] \oplus 0x0e \cdot w_7[2] \oplus 0xb3 \cdot w_7[3] \oplus 0x06 \cdot w_7[4] \\ & \oplus 0xea \cdot w_7[5] \oplus 0x90 \cdot w_7[6] \oplus 0x36 \cdot w_7[7]. \end{aligned} \tag{6}$$

Let $e^0_{in} = z_7[2] \oplus 0xbb \cdot z_7[3]$, $e^1_{in} = z_7[2] \oplus 0x2d \cdot z_7[3]$, $e^0_{out} = 0x3c \cdot x_8[0] \oplus 0x8d \cdot x_8[1] \oplus 0x49 \cdot x_8[2] \oplus 0x68 \cdot x_8[3] \oplus 0xbb \cdot x_8[4] \oplus 0x75 \cdot x_8[5] \oplus 0xbf \cdot x_8[6]$ and $e^1_{out} = 0x6f \cdot x_8[1] \oplus 0x0e \cdot x_8[2] \oplus 0xb3 \cdot x_8[3] \oplus 0x06 \cdot x_8[4] \oplus 0xea \cdot x_8[5] \oplus 0x90 \cdot x_8[6] \oplus 0x36 \cdot x_8[7]$,

then $(\Delta e^0_{out}, \Delta e^1_{out}) = (\Delta e^0_{in}, \Delta e^1_{in})$. Since the key size of Kalyna-256/512 is 512-bit and the total number of 2-multiset is $2^{2412.72}$, the probability for a wrong key past the test is $2^{512-2412.72} = 2^{-1900.72}$, it's almost impossible, so we choose 2-multiset. The 7-round distinguisher on Kalyna-256/512 is based on the proposition below.

Proposition 6. *If a message m belongs to a pair of states conforming to the truncated differential characteristic of Fig. 9(a), then the multiset of differences* $(\Delta e^0_{out}, \Delta e^1_{out})$ *obtained from the* δ*-set constructed from m in* x_0*, where* $x_1[31]$ *is the active byte, can only take about* 2^{400} *values.*

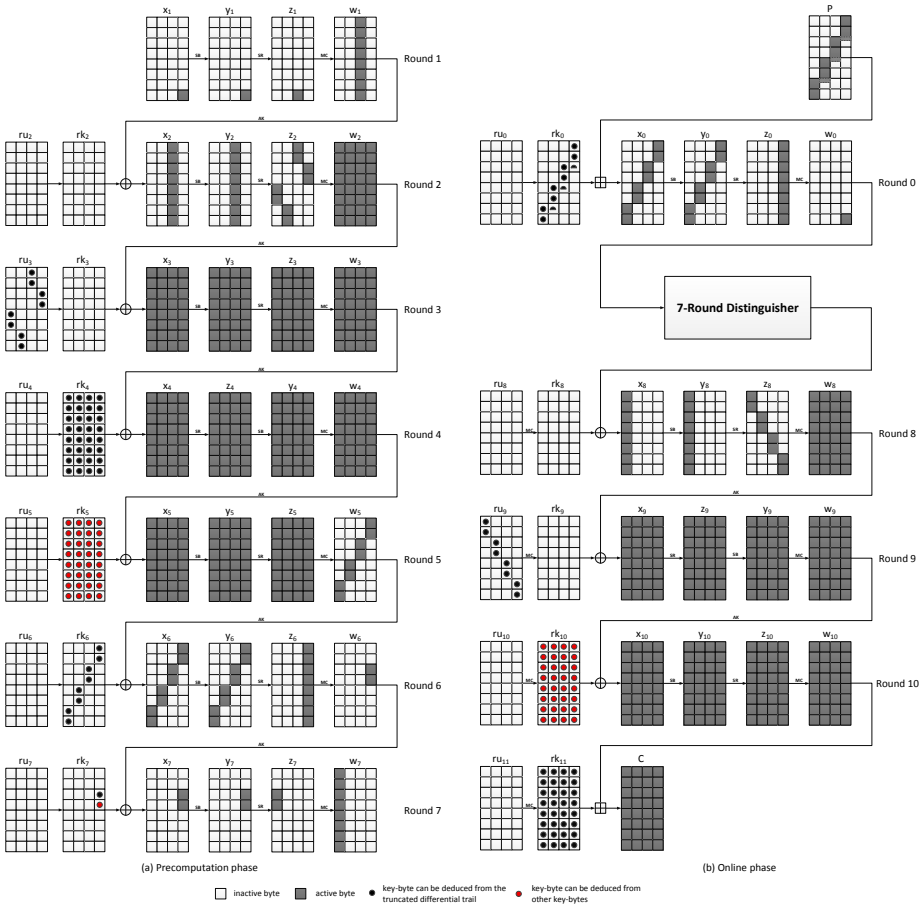

Fig. 9. The attack on 11-round Kalyna-256/512. The 7-round distinguisher is shown in (b), the online phase is shown in (b).

Proof. As shown in Fig. 9(a), for the encryption of the δ-set, the output 2-multiset $(\Delta e^0_{out}, \Delta e^1_{out})$ is determined by the 112 byte-parameters:

$$x_2[16-23]||x_3||rk_4||rk_5||rk_6[6,7,12,13,18,19,24,25] \tag{7}$$

However, if a pair of messages conforms to the truncated differential trail outlined in Fig. 9(a), the above 112 byte-parameters are determined by the 83 byte-parameters:

$$\Delta z_1[23]||x_2[16-23]||x_3||x_5||y_6[6,7,12,13,18,19,24,25]||\Delta z_7[2,3]. \tag{8}$$

Meanwhile, $ru_3[4,5,14,15,16,17,26,27]||rk_4||rk_5||rk_6[6,7,12,13,18,19,24,25]||rk_7[26,27]$ can be determined by the above 83 byte-parameters. Since rk_4 can be deduced from rk_5, $rk_7[27]$ can de deduced from $rk_6[6]$, by the key-dependent sieve technique, there are 2^{400} possible values for the 112 byte-parameters (7).

So the 112 byte-parameters (7) are determined by 83 byte-parameters (8), i.e., the 2-multiset of $(\Delta e^0_{out}, \Delta e^1_{out})$ can take about 2^{400} values.

□

4.3 Attack on 11-Round Kalyna-256/512

The attack is made up of two phases: precomputation phase and online phase (we change the order of SB and SR at Round 4 and Round 9).

Precomputation phase: In the precomputation phase, we need to build a table T_6 that contains all the the 2-multisets of $(\Delta e^0_{out}, \Delta e^1_{out})$ described in Propostion 6. To use the key-dependent sieve technique to improve the complexity, we need to build seven more tables T_1, T_2, T_3, $T_4^{0,2}$, $T_4^{1,3}$, T_5^1 and T_5^2. Otherwise, we need to build three more tables T_7, T_8^1 and T_8^2 for online phase.

1. As shown in Fig. 9(a), for each value of $\Delta z_1[23]||\Delta z_2[4,5,14,15,16,17,26,27]$, deduce $\Delta x_2[16-23]$ and $\Delta y_2[16-23]$, then deduce $x_2[16-23]$ according to Proposition 1. Store $x_2[16-23]$ in a table T_1 with the index of $\Delta z_2[4,5,14,15,16,17,26,27]$. There are about 2^8 values for each index.
2. For each value of $\Delta z_2[4,5,14,15,16,17,26,27]||\Delta w_3$, compute Δx_3 and Δy_3. Deduce x_3 and y_3 according to Proposition 1. Store x_3 in a table T_2 with the index of $\Delta z_2[4,5,14,15,16,17,26,27]||\Delta w_3$. There is about one value for each index.
3. For each value of $\Delta z_7[2,3]||\Delta x_7[26,27]||\Delta x_6[6,7,12,13,18,19,24,25]$, deduce $x_6[6,7,12,13,18,19,24,25]$ and $x_7[26,27]$ according to Proposition 1. Deduce $rk_7[26,27]$ from $x_6[6,7,12,13,18,19,24,25]$ and $x_7[26,27]$. Since $rk_6[6] = rk_7[27]$, we can deduce $w_5[6]$. Store $rk_7[26,27]||x_6[6,7,12,13,18,19,24,25]$ in a table T_3 with the index of $w_5[6]||\Delta x_6[6,7,12,13,18,19,24,25]$. There are about 2^{24} values for each index.
4. (a) For each value of $\Delta z_4[0-7,16-23]||\Delta w_5[6,7,12,13,18,19,24,25]||rk_5[0-7,16-23]$, deduce $\Delta y_5[0-7,16-23]$, then deduce $z_4[0-7,16-23]||y_5[0-7,16-23]$ according to Proposition 2. Deduce $rk_4[0,1,11,14,15,16,17,27,30,31]$ from $rk_5[21,22,0,3,4,5,6,16,19,20]$ according to the key-schedule algorithm. Deduce $w_3[0,1,11,14,15,16,17,27,30,31]$ from $z_4[0,1,19,6,7,16,17,3,22,23]$ and $rk_4[0,1,11,14,15,16,17,27,30,31]$. Store $z_4[0-7,16-23]||y_5[0$

$-7,16-23]||rk_5[0-7,16-23]$ in a table $T_4^{0,2}$ with the index of $w_3[0,1,11,14,15,16,17,27,30,31]||\Delta z_4[0-7,16-23]||\Delta w_5[6,7,12,13,18,19,24,25]$. There are about 2^{48} values for each index.

(b) For each value of $\Delta z_4[8-15,24-31]||\Delta w_5[6,7,12,13,18,19,24,25]||rk_5[8-15,24-31]$, deduce $\Delta y_5[8-15,24-31]$, then deduce $z_4[8-15,24-31]||y_5[8-15,24-31]$ according to Proposition 2. Deduce $rk_4[3,6,7,8,9,19,22,23,24,25]$ from $rk_5[24,27,28,29,30,8,11,12,13,14]$ according to the key-schedule algorithm. Deduce $w_3[3,6,7,8,9,19,22,23,24,25]$ from $z_4[11,30,31,8,9,27,14,15,24,25]$ and $rk_4[3,6,7,8,9,19,22,23,24,25]$. Store $z_4[8-15,24-31]||y_5[8-15,24-31]||rk_5[8-15,24-31]$ in a table $T_4^{1,3}$ with the index of $w_3[3,6,7,8,9,19,22,23,24,25]||\Delta z_4[8-15,24-31]||\Delta w_5[6,7,12,13,18,19,24,25]$. There are about 2^{48} values for each index.

5. For each 384-bit value of $\Delta z_4||\Delta z_2[4,5,14,15,16,17,26,27]||\Delta w_5[6,7,12,13,18,19,24,25]$, look up the table T_2 to get about one value of x_3 with the index of $\Delta z_2[4,5,14,15,16,17,26,27]||\Delta x_4$. Then do the following sub-steps:

(a) Look up the table $T_4^{0,2}$ to get about 2^{48} values of $z_4[0-7,16-23]||y_5[0-7,16-23]||rk_5[0-7,16-23]$ with the index of $w_3[0,1,11,14,15,16,17,27,30,31]||\Delta z_4[0-7,16-23]||\Delta w_5[6,7,12,13,18,19,24,25]$. Deduce $rk_4[2,12,13,18,28,29]$ from $rk_5[23,1,2,7,17,18]$, deduce $rk_4[4,5,10,20,21,26]$ from x_3 and $z_4[0-7,16-23]$, and deduce $rk_5[9,10,15,25,26,31]$ from $rk_4[20,21,26,4,5,10]$. Store $y_5[0-7,16-23]||rk_5[0-7,16-23]$ in a table T_5^1 with the index of $rk_4[2,12,13,18,28,29]||rk_5[9,10,15,25,26,31]$.

(b) Look up the table $T_4^{1,3}$ to get about 2^{48} values of $z_4[8-15,24-31]||y_5[8-15,24-31]||rk_5[8-15,24-31]$ with the index of $w_3[3,6,7,8,9,19,22,23,24,25]||\Delta z_4[8-15,24-31]||\Delta w_5[6,7,12,13,18,19,24,25]$. Deduce $rk_4[2,12,13,18,28,29]$ from x_3 and $z_4[8-15,24-31]$. Store $y_5[8-15,24-31]||rk_5[8-15,24-31]$ in a table T_5^2 with the index of $rk_4[2,12,13,18,28,29]||rk_5[9,10,15,25,26,31]$.

(c) Look up the tables T_5^1 and T_5^2 to get about one value of $y_5||rk_5$ with the index of $rk_4[2,12,13,18,28,29]||rk_5[9,10,15,25,26,31]$, then deduce $w_5[6,7,12,13,18,19,24,25]$ and rk_4. Look up the table T_1 to get about 2^8 values of $x_2[16-23]$ with the index of $\Delta z_2[4,5,14,15,16,17,26,27]$, and look up the table T_3 to get about 2^{24} values of $rk_7[26,27]||x_6[6,7,12,13,18,19,24,25]$ with the index of $w_5[6]||\Delta x_6[6,7,12,13,18,19,24,25]$. Deduce $rk_6[6,7,12,13,18,19,24,25]$ from $w_5[6,7,12,13,18,19,24,25]$ and $x_6[6,7,12,13,18,19,24,25]$. Therefore, we get the 112 byte-parameters (7). Compute the 2-multiset of $(\Delta e^0_{out},\Delta e^1_{out})$, and store them in a table T_6.

6. We build one table T_7 for online phase. As shown in Fig. 8, for each 72-bit value $\Delta x_0[6,7,12,13,18,19,24,25]||\Delta w_0[31]$, deduce $x_0[6,7,12,13,18,19,24,25]$ according to Proposition 1. Guess $rk_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]$, and deduce the value of $x_0[14.0,20.0,26.0]$ from $rk_0[14.0,20.0,26.0]$ (subsection 4.1). Deduce $P_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]$ and $\Delta P_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]$ from $rk_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]$, $\Delta x_0[6,7,12,13,18,19,24,25]$ and $x_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]$. Store $\Delta x_0[6,7,12,13,18,19,24,25]||rk_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]||\Delta w_0[31]$ in a

table T_7 with the index of $P_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]||\Delta P_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]$. There are 2^5 values for each index.

7. We build two more tables T_8^1 and T_8^2 for online phase:
   (a) As shown in Fig. 9(b), for all 2^{320} values of $\Delta z_8[0,1,10,11,20,21,30,31]||\Delta y_{10}[8-23]||rk_{10}[8-23]$, deduce $\Delta z_9[8-23]$. Then deduce $z_9[8-23]$ and $y_{10}[8-23]$ according to Proposition 2. Deduce $z_{10}[4-9,14-19,26-29]$ and $\Delta z_{10}[4-9,14-19,26-29]$, and store $z_{10}[16-19,26-29]||\Delta z_{10}[16-19,26-29]||rk_{10}[8,9,10]||z_9[8-23]||\Delta z_8[0,1,10,11,20,21,30,31]$ in a table T_8^1 with the index of $rk_{10}[11-23]||z_{10}[4-9,14,15]||\Delta z_{10}[4-9,14,15]$. There are about 2^{88} values for each index.
   (b) For all 2^{320} values of $\Delta z_8[0,1,10,11,20,21,30,31]||\Delta y_{10}[0-7,24-31]||rk_{10}[0-7,24-31]$, deduce $\Delta z_9[0-7,24-31]$. Then deduce $z_9[0-7,24-31]$ and $y_{10}[0-7,24-31]$ according to Proposition 2. Deduce $z_{10}[0-3,10-13,20-25,30,31]$ and $\Delta z_{10}[0-3,10-13,20-25,30,31]$, and store $z_{10}[0-3,10-13]||\Delta z_{10}[0-3,10-13]||rk_{10}[24,25,26]||z_9[0-7,24-31]||\Delta z_8[0,1,10,11,20,21,30,31]$ in a table T_8^2 with the index of $rk_{10}[0-7,27-31]||z_{10}[20-25,30,31]||\Delta z_{10}[20-25,30,31]$. There are about 2^{88} values for each index.

Online phase: In the online phase of the attack, we first find at least one pair which satisfies the truncated differential trail in Fig. 9(a). To find the right pair, instead of guessing the sub-keys and checking whether this pair satisfies the truncated differential trail, we deduce the sub-keys which make it satisfy the truncated differential trail for each pair. Then we identify the δ-set, caculate the corresponding 2-multiset of $(\Delta e_{out}^0, \Delta e_{out}^1)$ and check whether it belongs to the table T_6.

1. **Phase A – Detecting the right pair.** Define a structure of 2^{67} plaintexts where $P_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]$ takes all the possible values, and the other bits are fixed to some constants. Hence, we can generate $2^{67} \times (2^{67}-1)/2 \approx 2^{133}$ pairs satisfying the plaintext difference. Choose 2^{166} structures to get about $2^{166+133} = 2^{299}$ pairs. As shown in Fig. 9(b), the probability to get the truncated differential trail in the forward and backward direction is $2^{(-7-30)\times 8-3} = 2^{-299}$, then about one pair follows the truncated differential trail for each guess of the key.
2. **Phase B – Checking the δ-set.** For each of the 2^{299} remaining pairs, we do the following sub-steps.
   (a) Guess $rk_{11}[0-15]$, and deduce $rk_{10}[11-26]$, $z_{10}[0-15]$ and $\Delta z_{10}[0-15]$. Look up the table T_8^1 to get about 2^{88} values of $z_{10}[16-19,26-29]||\Delta z_{10}[16-19,26-29]||rk_{10}[8,9,10]||z_9[8-23]||\Delta z_8[0,1,10,11,20,21,30,31]$ with the index of $rk_{10}[11-23]||z_{10}[4-9,14,15]||\Delta z_{10}[4-9,14,15]$. Store $z_9[8-23]||rk_{11}[0-15]||\Delta z_8[0,1,10,11,20,21,30,31]$ in a table T_9^1 with the index of $\Delta z_{10}[0-3,10-13,16-19,26-29]||z_{10}[0-3,10-13,16-19,26-29]||rk_{10}[21,22,23,29,30,31]||\Delta z_8[0,1,10,11,20,21,30,31]$. The size of this table is 2^{216}.
   (b) Guess $rk_{11}[16-31]$, and deduce $rk_{10}[0-10,27-31]$, $z_{10}[16-31]$ and $\Delta z_{10}[16-31]$. Look up the table T_8^2 to get about 2^{88} values of $z_{10}[0-3,10-13]||\Delta z_{10}[0-3,10-13]||rk_{10}[24,25,26]||z_9[0-7,24-31]||\Delta z_8[0,1,10,11,20,21,30,31]$ with the index of $rk_{10}[20-25,24-28]||z_{10}[20-25,30,31]||\Delta z_{10}[20-25,30,31]$. Store $z_9[0-7,24-31]||rk_{11}[16-31]||\Delta z_8[0,1,10,11,20,21,30,31]$ in a table T_9^2 with the index of $\Delta z_{10}[0-3,10-13,16-19,26-29]||z_{10}[0-3,10-$

$13,16-19,26-29]||rk_{10}[21,22,23,29,30,31]||\Delta z_8[0,1,10,11,20,21,30,31]$. The size of this table is 2^{216}.

(c) Merge T_9^1 and T_9^2 with the index of $z_{10}[0-3,10-13,16-19,26-29]||z_{10}[0-3,10-13,16-19,26-29]||rk_{10}[21,22,23,29,30,31]||\Delta z_8[0,1,10,11,20,21,30,31]$, and get about 2^{64} values of $rk_{11}||x_9||\Delta z_8[0,1,10,11,20,21,30,31]$. Guess $\Delta z_7[2,3]$, and deduce $\Delta x_8[0-7]$. According to Proposition 1, we can deduce $z_8[0,1,10,11,20,21,30,31]$ from $\Delta x_8[0-7]$ and $\Delta z_8[0,1,10,11,20,21,30,31]$. Deduce $ru_9[0,1,10,11,20,21,30,31]$ from $z_8[0,1,10,11,20,21,30,31]$ and x_9. Therefore, we get about 2^{80} values of rk_{11} (rk_{10} can be deduced from rk_{11}) and $ru_9[0,1,10,11,20,21,30,31]$.

(d) Look up the table T_7 to get about 2^5 values of $\Delta x_0[6,7,12,13,18,19,24,25]||rk_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]||\Delta w_0[15]$ with the index of $P_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]||\Delta P_0[6,7,12,13,14.0,18,19,20.0,24,25,26.0]$.

(e) Pick one member of the pair and get the δ-set where $w_0[15]$ is the active byte, then get the corresponding ciphertexts. For about 2^{80} values of rk_{11}, rk_{10} and $ru_9[0,1,10,11,20,21,30,31]$, decrypt the ciphertexts through the last 3 rounds and get the 2-multiset of $(\Delta e^0_{out}, \Delta e^1_{out})$. Check whether it lies in the precomputation table T_6. If not, try another one.

3. **Exhaustively search the rest of the key:** We can use the same method as [2] to recover all the sub-keys.

Complexity analysis. The probability for a wrong guess to pass this test is $2^{400-2412.72} \approx 2^{-2012.72}$. In the precomputation phase, in order to construct T_6, we need to perform 2^{400} partial encryptions on 2^8 messages. The time complexity of this phase is about $2^{400+8-1.62} = 2^{406.38}$ 11-round Kalyna-256/512 encryptions, the memory complexity is about $2^{400+3.5} = 2^{403.5}$ 256-bit blocks. In the online phase, the major time complexity comes from step 2a and 2b, so the time complexity of this phase is about $2^{299+128-3+1} = 2^{425}$ 11-round Kalyna-256/512 encryptions. In total, the time complexity of this attack is 2^{425} 11-round Kalyna-256/512 encryptions, the data complexity is 2^{233} chosen-plaintexts and the memory complexity is $2^{403.5}$ 256-bit blocks.

5 Conclusions

In this paper, we discussed the security of Kalyna-128/256 and Kalyna-256/512 against meet-in-the-middle attacks. Using the differential enumeration technique and key-dependent sieve technique, we proposed a 6-round meet-in-the-middle distinguisher on Kalyna-128/256. Based on this distinguisher, we added three rounds at the end to present a 9-round attack. After that, with a 7-round distinguisher and an addition plaintext structure, we got an attack on 11-round Kalyna-256/512 by adding one round at the beginning and three rounds at the end. To the best of our knowledge, these are the currently best attacks on Kalyna-128/256 and Kalyna-256/512.

References

1. Akshima, Donghoon Chang, Mohona Ghosh, Aarushi Goel, and Somitra Kumar Sanadhya. Single key recovery attacks on 9-round kalyna-128/256 and kalyna-256/512. In *Information*

Security and Cryptology - ICISC 2015 - 18th International Conference, Seoul, South Korea, November 25-27, 2015, Revised Selected Papers, pages 119–135, 2015.

2. Riham AlTawy, Ahmed Abdelkhalek, and Amr M. Youssef. A meet-in-the-middle attack on reduced-round kalyna-*b*/2*b*. *IEICE Transactions*, 99-D(4):1246–1250, 2016.
3. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.
4. Joan Daemen and Vincent Rijmen. Understanding two-round differentials in AES. In *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, pages 78–94, 2006.
5. Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, pages 116–126, 2008.
6. Hüseyin Demirci, Ihsan Taskin, Mustafa Çoban, and Adnan Baysal. Improved meet-in-the-middle attacks on AES. In *Progress in Cryptology - INDOCRYPT 2009, 10th International Conference on Cryptology in India, New Delhi, India, December 13-16, 2009. Proceedings*, pages 144–156, 2009.
7. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 371–387, 2013.
8. Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *IEEE Computer*, 10(6):74–84, 1977.
9. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 158–176, 2010.
10. Leibo Li, Keting Jia, and Xiaoyun Wang. Improved meet-in-the-middle attacks on aes-192 and prince. Cryptology ePrint Archive, Report 2013/573, 2013. http://eprint.iacr.org/2013/573.
11. Leibo Li, Keting Jia, and Xiaoyun Wang. Improved single-key attacks on 9-round AES-192/256. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 127–146, 2014.
12. Li Lin, Wenling Wu, Yanfeng Wang, and Lei Zhang. General model of the single-key meet-in-the-middle distinguisher on the word-oriented block cipher. In *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*, pages 203–223, 2013.
13. Roman Oliynykov, Ivan Gorbenko, Oleksandr Kazymyrov, Victor Ruzhentsev, Oleksandr Kuznetsov, Yurii Gorbenko, Oleksandr Dyrda, Viktor Dolgov, Andrii Pushkaryov, Ruslan Mordvinov, and Dmytro Kaidalov. DSTU 7624:2014. national standard of ukraine. information technologies. cryptographic data security. symmetric block transformation algorithm. *Ministry of Economical Development and Trade of Ukraine (in Ukrainian).*, 2015.
14. Roman Oliynykov, Ivan Gorbenko, Oleksandr Kazymyrov, Victor Ruzhentsev, Oleksandr Kuznetsov, Yurii Gorbenko, Oleksandr Dyrda, Viktor Dolgov, Andrii Pushkaryov, Ruslan Mordvinov, and Dmytro Kaidalov. A new encryption standard of ukraine: The kalyna block cipher. Cryptology ePrint Archive, Report 2015/650, 2015. http://eprint.iacr.org/.