Single Key Recovery Attacks on 9-round Kalyna-128/256 and Kalyna-256/512

Akshima, Donghoon Chang, Mohona Ghosh, Aarushi Goel, and Somitra Kumar Sanadhya

Indraprastha Institute of Information Technology, Delhi (IIIT-D), India
{akshima12014, donghoon, mohonag, aarushi12003, somitra}@iiitd.ac.in

Abstract. The Kalyna block cipher has recently been established as the Ukranian encryption standard in June, 2015. It was selected in a Ukrainian National Public Cryptographic Competition running from 2007 to 2010. Kalyna supports block sizes and key lengths of 128, 256 and 512 bits. Denoting the variants of Kalyna as Kalyna-b/k, where b denotes the block size and k denotes the keylength, the design specifies $k \in \{b, 2b\}$. In this work, we re-evaluate the security bound of some reduced round Kalyna variants, specifically Kalyna-128/256 and Kalyna-256/512 against key recovery attacks in the single key model. We first construct new 6-round distinguishers and then use these distinguishers to demonstrate 9-round attacks on these Kalyna variants. These attacks improve the previous best 7-round attacks on the same.
Our 9-round attack on Kalyna-128/256 has data, time and memory complexity of 2^{105}, $2^{245.83}$ and $2^{226.86}$ respectively. For our 9-round attack on Kalyna-256/512, the data/time/memory complexities are 2^{217}, $2^{477.83}$ and $2^{443.45}$ respectively. The time and data complexities for Kalyna-256/512 reported in this work improve upon the previous best 7-round attack complexities on the same. The attacks presented in this work are currently the best on Kalyna. We apply multiset attack - a variant of meet-in-the-middle attack to achieve these results.

Keywords. Block cipher, Kalyna, Key Recovery, Differential enumeration, Single key model

1 Introduction

The block cipher Kalyna proposed by Oliynykov et al. has been recently selected as Ukranian encryption standard in 2015. The official Kalyna specification [11] defines three block sizes, i.e., 128-bit, 256-bit and 512-bit and three key sizes - 128-bit, 256-bit and 512-bit where key size can be equal to or double the block length. Consequently, if we denote a variant of Kalyna as Kalyna - b/k, where, b and k denote block size and key size respectively, then the five variants of Kalyna are: Kalyna - 128/128, Kalyna - 128/256, Kalyna - 256/256, Kalyna - 256/512 and Kalyna - 512/512. The number of rounds of these variants are - 10, 14, 14, 18 and 18 respectively. Kalyna block cipher adopts an SPN (substitution-permutation network) structure, similar to AES [2] but with increased MDS matrix size, a new set of four different S-boxes, pre-and post-whitening modular 2^{64} key addition and a new key scheduling algorithm.

The official version of Kalyna specification (in English) available publicly does not include any security analysis of the design. A preliminary study in [10], before this cipher was standardized, reports attack complexities for Kalyna -128/128 against various attacks such as differential, linear, integral, impossible differential, boomerang etc. and shows that upto 5 rounds of this variant can be broken. Similar results are claimed for other Kalyna variants as well. The designers of Kalyna thus claim brute force security against Kalyna for rounds ≥ 6. In [1], AlTawy et al. presented the first detailed key recovery attack against standardized Kalyna-128/256 and Kalyna-256/512. They applied meet-in-the-middle (MITM) attack [6,5] to break 7-rounds of both Kalyna variants and demonstrated the best attack on Kalyna so far.

In this work, we extend the number of rounds attacked and show the first 9-round key recovery attack against Kalyna-128/256 and Kalyna-256/512. Similar to [1], our attack is inspired from the multiset attack demonstrated by Dunkelman et al. on AES in [6]. Multiset attack is a variant of meet-in-the-middle attack presented by Demirci et al. on AES in [4]. Demirci et al.'s attack involves constructing a set of functions which map one active byte in the first round to another active byte after 4-rounds of AES. This set of functions depend on 'P' parameters and can be described using a table of 2^P ordered 256-byte sequence of entries. This table is precomputed and stored, thus allowing building a 4-round distinguisher and attacking upto 8 rounds of AES. However, Demirci's attacks suffered from a very high memory complexity. To reduce the memory complexity of Demirci's attacks on AES, Dunkelman et al. in [6], proposed multiset attack which replaces the idea of storing 256 ordered byte sequences with 256 unordered byte sequences (with multiplicity). This reduced both memory and time complexity of MITM attack on AES by reducing the parameters to 'Q' (where, Q<P). They also introduced the novel idea of differential

enumeration technique to significantly lower the number of parameters required to construct the multiset from 'Q' to 'R' (where, R<Q<P), thus further decreasing the attack complexities on AES. Derbez at al. in [5] improved Dunkelman et al.'s attack on AES-192/256 by refining the differential enumeration technique. By using rebound-like techniques [8], they showed that the number of reachable multisets are much lower than those counted in Dunkelman et al.'s attack. This improvement allowed mounting of comparatively efficient attacks on AES and also enabled extension of number of rounds attacked. Due to structural similarities between Kalyna and AES, a similar attack was applied to 7-rounds of Kalyna by AlTawy et al. in [1]. The multiset attack on AES-192/256 was further improved by Li et al. in [9]. They extended the number of rounds attacked to 9 by introducing the concept of key sieving, where dependencies between the AES round keys were exploited to filter out the wrong states thus reducing the number of possible multisets. Recently, in [12], Li et al. demonstrated the most efficient multiset attack on AES-256. By exploiting some more key sieving properties and clever MixColumn properties, they extended the number of rounds attacked to 10. Since this line of work has produced the best results on AES and Kalyna has not yet received significant attention of the cryptanalysis community, we investigate the effectiveness of improved multiset attack on Kalyna in this work.

In our attacks, we examine Kalyna-128/256 and Kalyna-256/512. We construct new 6-round distinguishers for both the variants and use it to extend our attacks up to 9 rounds. For Kalyna-256/512, we significantly reduce the data and time complexities of the previous best 7-round attack on the same [1]. The key schedule algorithm of Kalyna does not allow recovery of all subkeys or the master key from one subkey only unlike AES [2]. However, it allows recovery of odd-round keys from even-round keys and vice-versa. This property will be used by us in our attacks to reduce the attack complexities. To the best of our knowledge, our attacks are the first attacks on 9-round Kalyna-128/256 and Kalyna-256/512 respectively.

Our Contribution. The main contributions of this work are as follows:

– We present the first 9-round key recovery attack on Kalyna-128/256 and Kalyna-256/512.
– We apply multiset attack to construct new 6-round distinguishers on each of the above mentioned Kalyna variants.
– Our 9-round attack on Kalyna-128/256 has data/time/memory complexity of 2^{105}, $2^{245.83}$ and $2^{226.86}$ respectively.
– Our 9-round attack on Kalyna-256/512 has data/time/memory complexity of 2^{217}, $2^{477.83}$ and $2^{443.45}$ respectively. This improves upon the previous best attack [1] in terms of time and data complexity as well.

Our results are summarized in Table 1.

Table 1. Comparison of cryptanalytic attacks on round reduced variants of Kalyna. The blank entries were not reported in [10]. (The memory complexity header represents the number of 128-bit blocks for Kalyna-128 and 256-bit blocks for Kalyna-256 required to be stored in memory.)

Algorithm	Rounds attacked	Attack type	Time complexity	Data complexity	Memory complexity	Reference
Kalyna-128/128	2	Interpolation	–	-	-	[10]
	3	Linear Attack	$2^{52.8}$	-	-	[10]
	4	Differential	2^{55}	-	-	[10]
	4	Boomerang	2^{120}	-	-	[10]
	5	Impossible Differential	2^{62}	-	2^{66}	[10]
	5	Integral	2^{97}	-	2^{33+4}	[10]
Kalyna-128/256	7	Meet-in-the-Middle	$2^{230.2}$	2^{89}	$2^{202.64}$	[1]
	9	Meet-in-the-Middle	$2^{245.83}$	2^{105}	$2^{226.86}$	This work, § 4
Kalyna-256/512	7	Meet-in-the-Middle	$2^{502.2}$	2^{233}	2^{170}	[1]
	9	Meet-in-the-middle	$2^{477.83}$	2^{217}	$2^{443.45}$	This work, § 5

Organization. In § 2, we provide a brief description of Kalyna and the notations adopted throughout the work. In § 3, we give details of our 6-round distinguisher for Kalyna 128/256 followed by § 4 where we present our 9-round attack on the same. In § 5, we describe our 6-round distinguisher for Kalyna-256/512 and in § 6 we discuss our 9-round attack on the same.In § **??**, we discuss some errors in the attack presented by Li et al. in [9]. Finally in § 7, we summarize and conclude our work.

2 Preliminaries

In this section, we first describe Kalyna and then mention the key notations and definitions used in our cryptanalysis technique to facilitate better understanding.

2.1 Description of Kalyna

As discussed in §1, the block cipher Kalyna-b/k has five variants namely - 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. The 128-bit, 256-bit and 512-bit internal states are treated as a byte matrix of 8×2 size, 8×4 size and 8×8 size respectively where, the bytes are numbered column-wise (as shown in Fig. 1). The pre-whitening and post-whitening keys are added modulo 2^{64} to the plaintext and ciphertext respectively columnwise. [1] Each internal round consists of 4 basic operations (as shown in Fig. 2):

0	8
1	9
2	10
3	11
4	12
5	13
6	14
7	15

(a)

0	8	16	24
1	9	17	25
2	10	18	26
3	11	19	27
4	12	20	28
5	13	21	29
6	14	22	30
7	15	23	31

(b)

0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

(c)

Fig. 1. (a)Byte numbering in a state of Kalyna-128. (b) Byte numbering in a state of Kalyna-256. (c) Byte numbering in a state of Kalyna-512

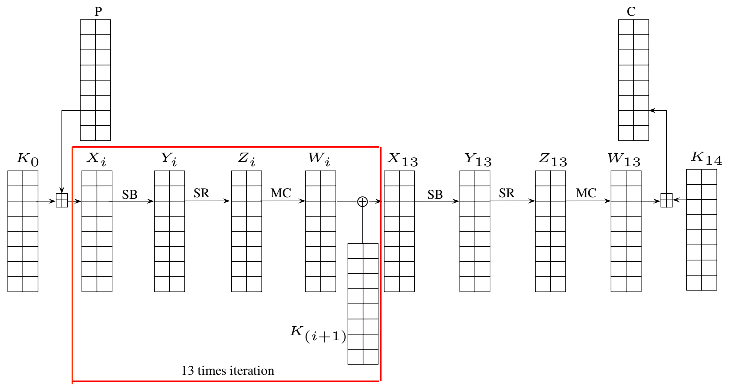

Fig. 2. One full encryption in Kalyna-128/256. (Refer to section 2.2 for notations)

1. *SubBytes (SB)* - The nonlinear substitution layer uses four types of 8-bit S-boxes: S_0, S_1, S_2 and S_3. Each S-Box is defined to be an affine transformation of the inversion function over GF(2^8). Each byte x of row j (where $0 \leq j \leq 7$) in an intermediate state s is substituted by $S_{j \ mod \ 4}\ (x)$.
2. *Shift Rows (SR)* - The linear shift rows operation performs circular right shift on each row of an internal state. The number of shifts (δ_j) in each row j (where $0 \leq j \leq 7$) is calculated by $\delta_j = \lfloor \frac{j \cdot b}{512} \rfloor$, where b denotes the block size. E.g., for row $j = 6$ and block size $b = 128$, the number of shifts $(\delta_j) = 1$. The inverse shift rows operation circularly left shifts the elements by δ_j in each row.
3. *MixColumn (MC)* - This linear transformation pre-multiplies each column of the state matrix by a 8×8 MDS matrix over GF(2^8). The vector `(0x01, 0x01, 0x05, 0x01, 0x08, 0x06, 0x07, 0x04)` forms the circulant MDS matrix for the MixColumn operation whereas the vector `(0xAD, 0x95, 0x76, 0xA8, 0x2F, 0x49, 0xD7, 0xCA)` forms the circulant MDS matrix for the inverse MixColumn operation. The branch factor of this MDS matrix is 9. The polynomial $x^8 + x^4 + x^3 + x^2 + 1$ (represented as `0x11D` in short) is used as the irreducible polynomial for Galois field multiplication. It is to be noted that unlike AES, in the last round of Kalyna, MixColumn operation is not omitted.
4. *Add Round Key (ARK)* - This step involves an exclusive-or operation with the round subkey. However, for the pre-whitening and post-whitening keys, key addition operation involves addition modulo 2^{64}. The round subkeys are of the same size as the intermediate state size.

[1] For addition operation, little endian format is used, i.e., less significant bytes have smaller indices. E.g., if the intermediate state value is: `0xCD 0x06 0x05 0x3 0x17 0x0A 0x03 0x02` where, `0xCD` is the most significant byte and `0x02` is the least significant byte. The representation in the little endian format would be `[ 0x02, 0x03, 0x0A, 0x17, 0x30, 0x05, 0x06, 0xCD]` corresponding to bytes [0, 1, 2, 3, 4, 5, 6, 7] of a column respectively.

Key Scheduling Algorithm. The key scheduling algorithm of Kalyna first involves splitting of the master key K into two parts - K_α and K_ω. If the block size and key size are equal, i.e., $(k = b)$, then $K_\alpha = K_\omega = K$, otherwise if $(k = 2b)$, then $K_\omega \mid\mid K_\alpha = K$, i.e., K_α is set as $b/2$ least significant bits of K and K_ω is set as $b/2$ most significant bits of K. Using these two parameters, an intermediate key K_σ is generated which is then used to independently generate even indexed round keys. Complete details of the key schedule algorithm are not relevant to the attacks described in this work and hence are omitted here. One may refer to [11] for the same. Two properties which are important for us are as follows:

1. Recovery of a subkey does not allow recovery of master key better than brute force.
2. The keys for round i where i is an odd number can be linearly computed from the key used in round $(i-1)$ and vice- versa as follows:

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

   where, $\lll$ denotes circular left shift operation.

2.2 Notations and Definitions

The following notations are followed throughout the rest of the paper.

P	: Plaintext
C	: Ciphertext
i	: Round number i, where, $0 \leq i \leq 8$
Kalyna-b	: Kalyna with state size of b-bits
Kalyna-b/k	: Kalyna with state size of b-bits and key size of k-bits
$\mathbf{K_i}$	: Subkey of round i
$\mathbf{U_i}$	: $MC^{-1}(\mathrm{K}_i)$, where, MC^{-1} is the inverse MixColumn operation
$\mathbf{X_i}$	: State before SB in round i
$\mathbf{Y_i}$	: State before SR in round i
$\mathbf{Z_i}$	: State before MC in round i
$\mathbf{W_i}$	: State after MC in round i
$\mathbf{\Delta s}$	: Difference in a state s
$\mathbf{s_i[m]}$	: m^{th} byte of state s in round i, where, $0 \leq m \leq l$ and $l = 15$ for Kalyna-128/256 and $l = 31$ for Kalyna-256/512
$\mathbf{s_i[p-r]}$	: p^{th} byte to r^{th} byte (both inclusive) of state s in round i, where, $0 \leq p < r \leq l$ and $l = 15$ for Kalyna-128/256 and $l = 31$ for Kalyna-256/512

In some cases we are interested in interchanging the order of the *MixColumn* and *Add Round Key* operations. As these operations are linear, they can be swapped, by first xoring the intermediate state with an equivalent key and then applying the *MixColummn* operation (as shown in Appendix A in Figs. 7, 8). This is exactly similar to what one can do in AES [5]. As mentioned above, we denote the equivalent round key by $U_i = MC^{-1}(K_i)$.

We utilize the following definitions for our attacks.

Definition 1 (δ-list). We define the δ-list as an ordered list of 256 16-byte (or 32-byte) distinct elements that are equal in 15 (or 31) bytes for Kalyna-128 (or Kalyna-256). Each of the equal bytes are called as passive bytes whereas the one byte that takes all possible 256 values is called the active byte [2]. We denote the δ-list as (x^0, x^1, $x^2, \ldots, x^{255}$) where x^j indicates the j^{th} 128-bit (or 256-bit) member of the δ-list for Kalyna-128 (or Kalyna-256). As mentioned in the notations, x_i^j [m] represents the m^{th} byte of x^j in round i.

Definition 2 (Multiset). A multiset is a set of elements in which multiple instances of the same element can appear. A multiset of 256 bytes, where each byte can take any one of the 256 possible values, can have $\binom{2^8+2^8-1}{2^8} \approx 2^{506.17}$ different values. [2]

Definition 3 (Super S-Box). The Kalyna Super S-box (denoted as SSB) can be defined similar to AES Super S-box [3]. For each 8-byte key, it produces a mapping between an 8-byte input array to an 8-byte output array. Formally, a two round Kalyna can be written as:

$$SB \rightarrow SR \rightarrow MC \rightarrow ARK \rightarrow SB \rightarrow SR \rightarrow MC \rightarrow ARK$$

[2] Count of multisets of cardinality r with elements from a set with cardinality $n = \binom{n+r-1}{r}$

or,

$$SR \rightarrow \underbrace{SB \rightarrow MC \rightarrow ARK \rightarrow SB} \rightarrow SR \rightarrow MC \rightarrow ARK\ ^{3} \qquad (2)$$

Since, MixColumns operation operates on a column of the state, the above map ($SB \rightarrow MC \rightarrow AK \rightarrow SB$) in Eq. 2 can be described as d parallel instances of SSB, where d = 2, 4 and 8 for Kalyna-128, Kalyna-256 and Kalyna-512 respectively.
Two important properties that will be used in our attacks are as follows:

Property 1a. (Kalyna S-box) For any given Kalyna S-box, say S_i (where, i = 0, 1, 2 or 3) and any non-zero input - output difference pair, say (Δ_{in}, Δ_{out}) in $F_{256} \times F_{256}$, there exists one solution in average, say y, for which the equation, $S_i(y) \oplus S_i(y \oplus \Delta_{in}) = \Delta_{out}$, holds true.

Proof. On analyzing the difference distribution table of the 4 S-boxes, it was observed that the number of solutions N (Δ_{in} - Δ_{out}) for any given Δ_{in} - Δ_{out} is either 0, 2, 4 or rarely 6 or 8. For a given Δ_{in} and its 256 possible Δ_{out}, the fequency of the solutions also vary depending on the choice of Δ_{in}. For any random (Δ_{in} - Δ_{out}), the probability that on average a solution exists can be calculated as: $\frac{256-a}{256}$ ($\frac{b}{256-a} \times 2 + \frac{c}{256-a} \times 4 + \frac{d}{256-a} \times 6 + \frac{e}{256-a} \times 8$) where a, b, c, d and e is the frequency of zeroes, twos, fours, sixes and eights respectively. Since 2b + 4c + 6d + 8e = 256, on average one solutions always exists.
E.g, in S_2, for Δ_{in} = 0xAF and all 256 Δ_{out}, there are 150 zeroes, 86 twos, 18 fours and 2 six. Hence, the average number of solutions can be calculated as: $\frac{256-150}{256}$ ($\frac{86}{106} \times 2 + \frac{18}{106} \times 4 + \frac{2}{106} \times 6$) = 1. Similar analysis can be done for other Δ_{in} in S_1 and all other 3 S-boxes, i.e., S_0, S_1 and S_3.

Property 1b. (Kalyna Super S-box) For any given Kalyna Super S-box, say SSB and any non-zero input - output difference pair, say (Δ_{in}, Δ_{out}) in $F_{2^{64}} \times F_{2^{64}}$, the equation, $SSB(z) \oplus SSB(z \oplus \Delta_{in}) = \Delta_{out}$ has one solution in average.

Property 2. (Kalyna MixColumns) If the values (or the differences) in any eight out of its sixteen input/output bytes of the Kalyna MixColumn operation are known, then the values (or the differences) in the other eight bytes are uniquely determined and can be computed efficiently. This is similar to AES MixColumn property stated in [12].

Proof. The Kalyna Mixcolumn works on a column of 8 bytes. Thus, the inputs and outputs of Kalyna MixColumn operation can be related through 8 equations. Therefore, out of 16 variables (8 input and 8 output), if 8 variables are known then the other 8 variables can be uniquely determined through the 8 equations. This is because as mentioned in [7], any sub-matrix of a MDS matrix is invertible which guarantees existence of an unique solution.

The time complexity of the attack is measured in terms of 9-round Kalyna encryptions required. The memory complexity is measured in units of b-bit Kalyna (where, b = 128 or 256) blocks required.

3 Construction of distinguisher for 6-round Kalyna-128/256

In this section, we construct a distinguisher on the 6-inner rounds of Kalyna-128/256. Before, we proceed further, we first establish the following relation for Kalyna-128/256. According to *Property 2*, we can form an equation using any 11 out of 16 input-output bytes in the Kalyna MixColumn operation. For any round j, where, $0 \leq j \leq 8$:

$$\begin{aligned}\texttt{0xCA} \cdot Z_j[12] \oplus \texttt{0xAD} \cdot 0xZ_j[13] \oplus \texttt{0x49} \cdot Z_j[14] \oplus \texttt{0xD7} \cdot Z_j[15] = {} & \texttt{0x94} \cdot W_j[8] \oplus \texttt{0xB4} \cdot W_j[9] \oplus \texttt{0x4E} \cdot W_j[10] \oplus \\ & \texttt{0x7E} \cdot W_j[11] \oplus \texttt{0xC0} \cdot W_j[13] \oplus \texttt{0xDA} \cdot W_j[14] \oplus \\ & \texttt{0xC5} \cdot W_j[15] \qquad (3)\end{aligned}$$

or,

$$\begin{aligned}\texttt{0xCA} \cdot Z_j[12] \oplus \texttt{0xAD} \cdot Z_j[13] \oplus \texttt{0x49} \cdot Z_j[14] \oplus \texttt{0xD7} \cdot Z_j[15] = {} & \texttt{0x94} \cdot (K_j[8] \oplus X_{j+1}[8]) \oplus \texttt{0xB4} \cdot (K_j[9] \oplus X_{j+1}[9]) \\ & \oplus \texttt{0x4E} \cdot (K_j[10] \oplus X_{j+1}[10]) \oplus \texttt{0x7E} \cdot (K_j[11] \oplus X_{j+1}[11]) \\ & \oplus \texttt{0xC0} \cdot (K_j[13] \oplus X_{j+1}[13]) \oplus \texttt{0xDA} \cdot (K_j[14] \oplus X_{j+1}[14]) \\ & \oplus \texttt{0xC5} \cdot (K_j[15] \oplus X_{j+1}[15]) \qquad (4)\end{aligned}$$

[3] Note that Sub Bytes and Shift Row operations in the first round have been interchanged as these functions commute with each other

where, $W_j = K_j \oplus X_{j+1}$. Derivation of Eq. 3 is shown in Appendix B. Let,

$$P_j = \texttt{0xCA} \cdot Z_j[12] \oplus \texttt{0xAD} \cdot Z_j[13] \oplus \texttt{0x49} \cdot Z_j[14] \oplus \texttt{0xD7} \cdot Z_j[15] \quad (5)$$
$$Q_j = \texttt{0x94} \cdot X_{j+1}[8] \oplus \texttt{0xB4} \cdot X_{j+1}[9] \oplus \texttt{0x4E} \cdot X_{j+1}[10] \oplus \texttt{0x7E} \cdot X_{j+1}[11] \oplus \texttt{0xC0} \cdot X_{j+1}[13] \oplus \texttt{0xDA} \cdot X_{j+1}[14] \oplus \texttt{0xC5} \cdot X_{j+1}[15] \quad (6)$$
$$Const = \texttt{0x94} \cdot K_j[8] \oplus \texttt{0xB4} \cdot K_j[9] \oplus \texttt{0x4E} \cdot K_j[10] \oplus \texttt{0x7E} \cdot K_j[11] \oplus \texttt{0xC0} \cdot K_j[13] \oplus \texttt{0xDA} \cdot K_j[14] \oplus \texttt{0xC5} \cdot K_j[15] \quad (7)$$

then, Eq. 4 can be rewritten as,

$$P_j = Q_j \oplus Const \quad (8)$$

Eq. 8 will be used to establish the distinguishing property as shown next.

3.1 Distinguishing Property for Kalyna-128/256

Given, a list of 256 distinct bytes (M^0, M^1, ... , M^{255}), a function $f : \{0,1\}^{128} \mapsto \{0,1\}^{128}$ and a 120-bit constant T, we define a multiset v as follows :

$$C^i = f(T \mid\mid M^i), \text{where } (0 \le i \le 255) \quad (9)$$
$$u^i = \texttt{0x94} \cdot C^i[8] \oplus \texttt{0xB4} \cdot C^i[9] \oplus \texttt{0x4E} \cdot C^i[10] \oplus \texttt{0x7E} \cdot C^i[11] \oplus \texttt{0xC0} \cdot C^i[13] \oplus \texttt{0xDA} \cdot C^i[14] \oplus \texttt{0xC5} \cdot C^i[15] \quad (10)$$
$$v = \{u^0 \oplus u^0, u^1 \oplus u^0, \ldots, u^{255} \oplus u^0\} \quad (11)$$

Note that, (T || M^0, T || M^1, ... , T || M^{255}) forms a δ-list and atleast one element of v (i.e., $u^0 \oplus u^0$) is always zero.

Distinguishing Property. Let us consider $\mathcal{F}$ to be a family of permutations on 128-bit. Then, given any list of 256 distinct bytes (M^0, M^1, ... , M^{255}), the aim is to find how many multisets v (as defined above) are possible when, $f \xleftarrow{\$} \mathcal{F}$ and $T \xleftarrow{\$} \{0,1\}^{120}$.

In case, when $\mathcal{F}$ = family of all permutations on 128-bit and $f \xleftarrow{\$} \mathcal{F}$. Under such setting, since in the multiset v, we have 255 values (one element is always 0) that are chosen uniformly and independently from the set {0, 1, ..., 255 }, the total number of possible multisets v are atmost $\binom{2^8-1+2^8-1}{2^8-1} \approx 2^{505.17}$.

In case, when $\mathcal{F}$ = 6-full rounds of Kalyna-128/256 and $f \xleftarrow{\$} \mathcal{F}$. Here, $f \xleftarrow{\$} \mathcal{F} \Leftrightarrow K \xleftarrow{\$} \{0,1\}^{256}$ and $f = E_K$. Let us consider the 6 inner rounds of Kalyna-128/256 as shown in Fig. 3. Here, C in Eq. 9 is represented by X_6 and Eq. 10 is defined as :

$$u^i = \texttt{0x94} \cdot X_6^i[8] \oplus \texttt{0xB4} \cdot X_6^i[9] \oplus \texttt{0x4E} \cdot X_6^i[10] \oplus \texttt{0x7E} \cdot X_6^i[11] \oplus \texttt{0xC0} \cdot X_6^i[13] \oplus \texttt{0xDA} \cdot X_6^i[14] \oplus \texttt{0xC5} \cdot X_6^i[15] \quad (12)$$

It is to be noted that under this setting, for each i where, $(0 \le i \le 255)$, Eq. 12 is same as Eq. 6 computed at round 5, i.e., $u^i = Q_5^i$. Now, we state the following *Observation 1.*

Observation 1. The multiset v is determined by the following 52 single byte parameters only :

- $X_1^0[0 - 7]$ (8-bytes)
- $X_2^0[0 - 15]$ (16-bytes)
- $X_3^0[0 - 15]$ (16-bytes)
- $X_4^0[0 - 3, 12 - 15]$ (8-bytes)
- $X_5^0[4 - 7]$ (4-bytes)

Thus, the total number of possible multisets is $2^{52\times 8} = 2^{416}$ since, each 52-byte value defines one sequence.

Proof. In round 0 (in Fig. 3), the set of differences $\{X_0^0[15] \oplus X_0^0[15],\, X_0^1[15] \oplus X_0^0[15],\, \dots,\, X_0^{255}[15] \oplus X_0^0[15]\}$ (or, equivalently the set of differences at $X_0[15]$) is known to the attacker as there are exactly 256 differences possible. This is so, because in the plaintext we make the most significant byte as the active byte. Hence, when the pre-whitening key is added (columnwise), the carry-bit in the most significant bit is ignored limiting the possible values (and the differences) at $X_0[15]$ to 256 only. Since S-box is injective, exactly 256 values exist in the set $\{Y_0^0[15] \oplus Y_0^0[15],\, Y_0^1[15] \oplus Y_0^0[15],\, \dots,\, Y_0^{255}[15] \oplus Y_0^0[15]\}$. As *Shift Row* (SR), *MixColumn* (MC) and *Add Round Key* (ARK) are linear operations, the set of differences at $X_1[0-7]$ will be known to the attacker.

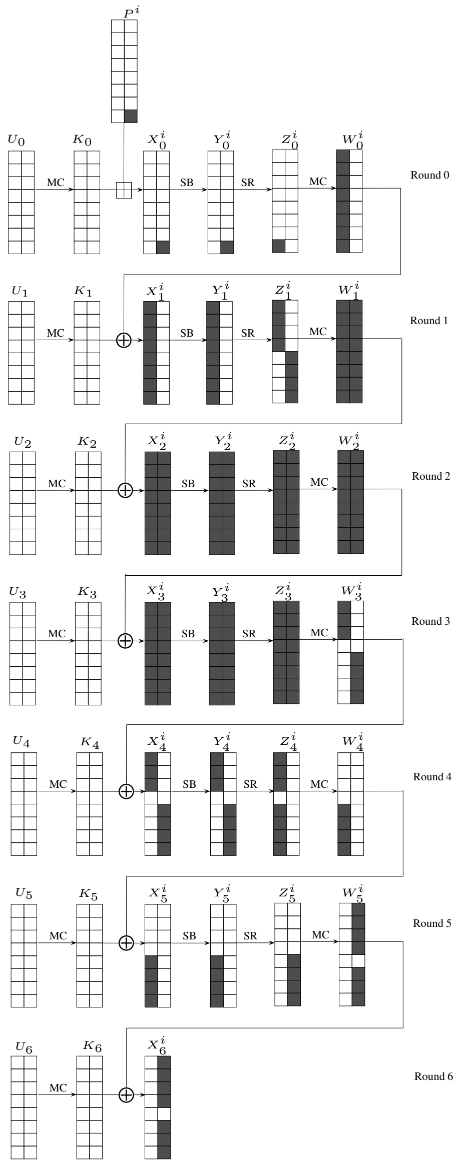

Fig. 3. 6-Round distinguisher in Kalyna-128/256 . Here, P^i denotes $(T \,||\, M^i)$ and X_j^i, Y_j^i, Z_j^i, W_j^i denote intermediate states corresponding to P^i in round j. The round subkeys K_j, where, $0 \le j \le 6$ are generated from the master key K.

Owing to the non-linearity of the S-box operation, the set of differences at Y_1 [0-7] cannot be computed to move forward. To allieviate this problem, it is sufficient to guess X_1^0[0-7], i.e., values of the active bytes of the first state (out of 256 states) at X_1 as it allows calculating the other X_1^i[0-7] states (where, 1 ≤ i ≤ 255) and cross SB layer in round 1. Since, SR, MC and ARK operations are linear, the set of differences at $X_2[0-15]$ is known. Continuing in a similar manner as discussed above, if the attacker guesses full states X_2^0 [0-15] and X_3^0 [0-15], then the set of differences at Z_3, i.e., $\{Z_3^0 \oplus Z_3^0,\ Z_3^1 \oplus Z_3^0,\ \ldots,\ Z_3^{255} \oplus Z_3^0\}$ can be easily computed. Now at this stage, she can easily calculate the set of differences at W_3 [0, 1, 2, 3, 12, 13, 14, 15] which is equal to the set of differences at X_4 [0, 1, 2, 3, 12, 13, 14, 15]. [4]. By guessing X_4^0 [0, 1, 2, 3, 12, 13, 14, 15], the attacker can cross the SB layer in round 4 and calculate the set of differences at W_4 [4, 5, 6, 7]. By guessing X_5^0 [4, 5, 6, 7], the attacker can obtain the set of values $\{Z_5^0[12-15],\ Z_5^1[12-15],\ \ldots\ ,\ Z_5^{255}[12-15]\}$. Using these, she can compute P_5^i at Z_5^i as $P_5^i = CA_x \cdot Z_5^i[12] \oplus AD_x \cdot Z_5^i[13] \oplus 49_x \cdot Z_5^i[14] \oplus D7_x \cdot Z_5^i[15]$ (according to Eq. 5) and thus the set $\{P_5^0 \oplus P_5^0,\ P_5^0 \oplus P_5^1,\ \ldots,\ P_5^{255} \oplus P_5^0\}$. Since, according to Eq. 8, $P_j^i \oplus P_j^0 = (Q_j^i \oplus\ Const) \oplus (Q_j^0 \oplus\ Const) = Q_j^i \oplus Q_j^0$ and $u^i = Q_5^i$ (mentioned above), the attacker can easily calculate the multiset $v = \{Q_5^0 \oplus Q_5^0,\ Q_5^1 \oplus Q_5^0,\ \ldots,\ Q_5^{255} \oplus Q_5^0\}$. This shows that the multiset v depends on 52 parameters and can take 2^{416} possible values. □

Since, there are 2^{416} possible multisets, if we precompute and store these values in a hash table, then the precomputation complexity goes higher than brute force for Kalyna-128/256. In order to reduce the number of multisets, we apply the Differential Enumeration technique suggested by Dunkelman et al. in [6] and improved by Derbez et al. in [5]. We call the improved version proposed in [5] as *Refined Differential Enumeration*.

Refined Differential Enumeration. The basic idea behind this technique is to choose a δ-set such that several of the parameters mentioned in *Observation 1* equal some pre-determined constants. To achieve so, we first construct a 6-round truncated differential trail in round 0 - round 5 (as shown in Fig. 3) where, the input difference is non-zero at one byte and output difference is non zero in 7 bytes. The probability of such a trail is 2^{-112} as follows: the one byte difference at $\Delta P[15]$ and correspondingly at $\Delta X_0[15]$ propagates to 8-byte difference in $\Delta X_1[0-7]$ and 16-byte difference in $\Delta X_2[0-15]$ and further till $\Delta Z_3[0-15]$ with probability 1. Next, the probability that 16-byte difference in $\Delta Z_3[0-15]$ propagates to 7-byte difference in $\Delta W_3[0-2, 12-15]$ $(= \Delta X_4[0-2, 12-15])$ is 2^{-72}. This 7-byte difference in ΔX_4 propagates to 4-byte difference in $\Delta W_4[4-7]$ followed by 7-byte difference in $\Delta W_5[8-11, 13-15]$ with a probability of 2^{-32} and 2^{-8} respectively. Thus, the overall probability of the differential from ΔP to ΔZ_5 is $2^{-(72+32+8)} = 2^{-112}$.
In other words, we require 2^{112} plaintext pairs to get a right pair. Once, we get a right pair, say (P^0, P^1), we state the following *Observation 2*.

Observation 2. Given a right pair (P^0, P^1) that follows the truncated differential trail shown in Fig. 3, the 52 parameters corresponding to P^0, mentioned in Observation 1 can take one of atmost 2^{224} fixed 52-byte values (out of the total 2^{416} possible values), where each of these 2^{224} 52-byte values are defined by each of the 2^{224} values of the following 39 parameters:

- $\Delta Z_0[7]$ (1-byte)
- $X_1^0[0-7]$ (8-bytes)
- $Y_3^0[0-15]$ (16-bytes)
- $Y_4^0[0-3, 12-15]$ (8-bytes)
- $Y_5^0[5-7]$ (3-bytes)
- $\Delta Z_5[12-14]$ (3-bytes)

Proof. Given a right pair (P^0, P^1), the knowledge of these 39 new parameters allows us to compute all the differences shown in Fig. 3. This is so because the knowledge of $\Delta Z_0[7]$ allows us to compute $\Delta X_1[0-7]$. Then, if the values of $X_1^0[0-7]$ are known, one can compute the corresponding $X_1^1[0-7]$ and cross the S-box layer in round 1 to get ΔX_2.

From the bottom side, it can be seen that $\Delta W_5[12] = \Delta Z_5[8] = \Delta Z_5[9] = \Delta Z_5[10] = \Delta Z_5[11] = 0$. Thus, if $\Delta Z_5[12, 13, 14]$ are known, then using *Property 2* (as 8 bytes are known), we can deduce $\Delta Z_5[15]$ (and ΔW_5 [8-11, 13-15]). Knowledge of $\Delta Z_5[8-15]$ allows us to to compute $\Delta Y_5[4-7]$. Then, by guessing $Y_5^0[5-7]$, we can determine the corresponding $Y_5^1[5-7]$ and compute $\Delta X_5[5-7]$ (and $\Delta W_4[5-7]$). Now again, we know that $\Delta W_4[0] = \Delta W_4[1] = \Delta W_4[2] = \Delta W_4[3] = \Delta Z_4[3] = 0$. Using *Property 2* (as 8 bytes are known), we can deduce $\Delta W_4[4]$ (and $\Delta Z_4[0-2, 4-7]$). This allows us to compute $\Delta X_5[4]$ as well. Since we already know $\Delta Y_5[4]$ (from $\Delta Z_5[12]$ guessed previously), using *Property 1a.*, the possible values of $X_5[4]$ and $Y_5[4]$ can be computed.

Now, knowledge of $\Delta Z_4[0-7]$ allows us to compute $\Delta Y_4[0-3, 12-15]$. By guessing $Y_4^0[0-3, 12-15]$, we can obtain $\Delta Y_3[0-15]$. Using the value of $Y_3^0[0-15]$, we can compute ΔY_2. Then using *Property 1a.*, the possible

[4] In Fig. 3, byte 3 in states W_3, X_4, Y_4 and Z_4 have not been colored grey for a purpose which will be cleared when we reach *Observation 2*

values of X_2^0 and Y_2^0 can be computed. At this stage, the total possible values of these 39 parameters are $2^{39\times 8} = 2^{312}$.

However, for each value of this 39-byte parameter, the following key bytes - $U_2[0-3,12-15]$, K_3, $K_4[0-3,12-15]$ and $K_5[4-7]$ can be deduced as follows:

1. Knowledge of $X_1^0[0-7]$ allows us to compute the corresponding $Z_1^0[0-3,12-15]$. Xoring these values with $X_2^0[0-3,12-15]$ helps us in deducing $U_2[0-3,12-15]$.
2. Knowledge of X_2^0 allows us to compute the corresponding W_2^0. Xoring W_2^0 with X_3^0 helps us in deducing K_3.
3. Knowledge of X_3^0 and $X_4^0[0-3,12-15]$ (from $Y_4^0[0-3,12-15]$) can be used to deduce $K_4[0-3,12-15]$.
4. Knowledge of $X_4^0[0-3,12-15]$ and $X_5^0[4-7]$ (from $Y_5^0[4-7]$) helps us in deducing $K_5[4-7]$.

Now, according to the key schedule algorithm of Kalyna-128/256, from K_3, we can compute K_2 (according to Eq. 1) which allows us to compute the corresponding U_2. Thus, by comparing the computed $U_2[0-3,12-15]$ with the deduced $U_2[0-3,12-15]$, a sieve of 8-bytes (since matching probability is 2^{-64}) can be applied to eliminate the wrong guesses. Similarly, again from Eq. 1, knowledge of $K_5[4-7]$ allows us to compute K_4 [12], K_4 [13] and K_4 [14] as $K_4[12] = K_5[5]$, $K_4[13] = K_5[6]$ and $K_4[14] = K_5[7]$. This allows us a filtering of further 3-bytes. Thus by key sieving, the total possible guesses of 39-byte parameter reduces from $2^{39\times 8}$ to $2^{(39-(8+3))\times 8} = 2^{28\times 8} = 2^{224}$. □

Using *Observation 1* and *Observation 2*, we state the following third *Observation 3*:

Observation 3. Given (M^0, M^1, ... , M^{255}) and $f \xleftarrow{\$} \mathcal{F}$ and $T \xleftarrow{\$} \{0,1\}^{120}$, such that $T \mid\mid M^0$ and $T \mid\mid M^j$, (where, $j \in \{ 1, \dots, 255 \}$) is a right pair that follows the differential trail shown in Fig. 3, atmost 2^{224} multisets v are possible.

Proof. From *Observation 1*, we know that each 52-byte parameter defines one multiset and *Observation 2* restricts the possible values of these 52-byte parameters to 2^{224}. Thus, atmost 2^{224} multisets are only possible for Kalyna-128/256. □

As the number of multisets in case of 128-bit random permutation ($= 2^{505.17}$) is much higher than 6-round Kalyna-128/256 ($= 2^{224}$), a valid distinguisher is constructed.

4 Key Recovery Attack on 9-Round Kalyna-128/256

In this section, we use our Observation 3 to launch meet-in-the-middle attack on 9-round Kalyna-128/256 to recover the key. The distinguisher is placed in round 0 to round 5, i.e, the set of plaintexts is considered as the δ-list with byte 15 being the active byte and the multiset sequence being checked at X_6 (as shown in Fig. 4). Three rounds are added at the bottom of the 6-round distinguisher. The attack consists of the following three phases:

4.1 Precomputation Phase

In this phase, we build a lookup table T to store 2^{224} sequences to be used for comparison in the online phase. The construction of this table requires us to create two more hash tables (T_0 and T_1) in the intermediate steps. The entire procedure is as follows:

1. For each K_3
   - We guess $\Delta Z_1[0-3,12-15]||\Delta X_4[0-2,12-15]$ to compute the difference ΔX_2 and ΔY_3 respectively. We resolve (ΔX_2 - ΔY_3) using *Property 1b* to compute the corresponding $X_2||X_3$. We then deduce K_2 from K_3 and compute the corresponding value of $Z_1[0-3,12-15]$. Using the guessed value of $\Delta Z_1[0-3,12-15]$ and the computed value of $Z_1[0-3,12-15]$, we compute $\Delta Z_0[0-7]$. If $\Delta Z_0[0-6] = 0$ (which happens with a probability of 2^{-56}), we store the corresponding $X_1[0-7]||\Delta Z_1[0-3,12-15]||X_2||X_3||W_3[12-14]||\Delta X_4[0-2,12-15]$ at index K_3 in table T_0. There are about 2^{64} entries for each index.

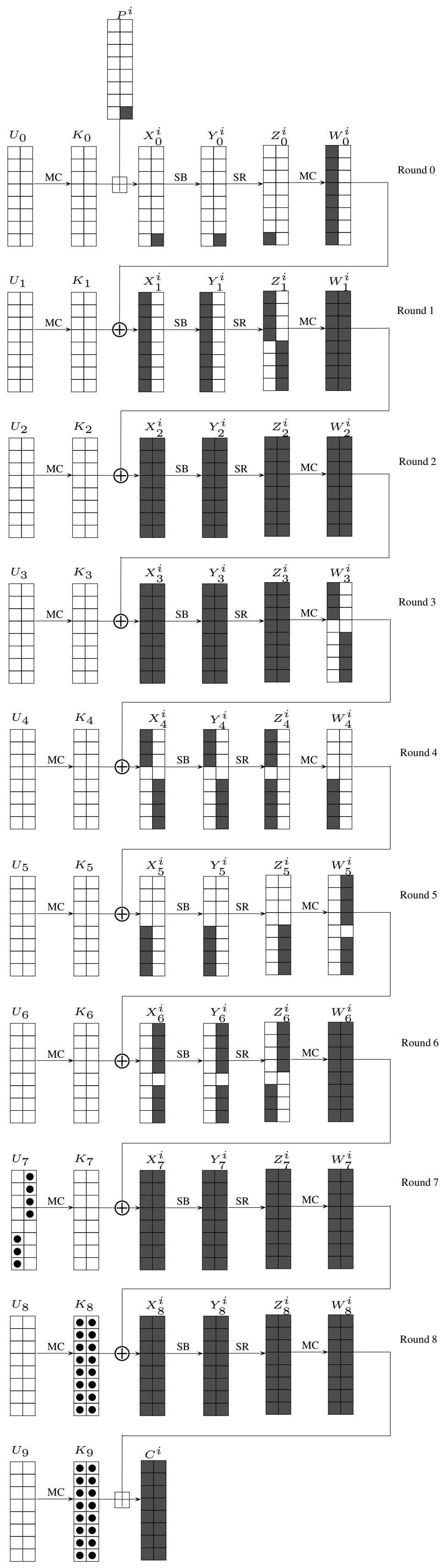

Fig. 4. 9-round attack on Kalyna-128/256. The subkey bytes guessed are shown dotted.

2. For each guess of $\Delta Z_5[12-14]$
   - We compute $\Delta Z_5[15]$ using *Property 2.*
   - We guess $Y_5[5-7]$, compute $X_5[5-7]$ and $\Delta X_5[0-3,5-7]$ where, $\Delta X_5[0-3]=0$. Since, $\Delta X_5[0-3,5-7]=\Delta W_4[0-3,5-7]$ and we know that $\Delta Z_4[3]=0$, thus we can compute $\Delta X_5[4]$ $(=\Delta W_4[4])$ and $\Delta Z_4[0-2,4-7]$ again using *Property 2.* Since $\Delta Y_5[4]$ is known from $\Delta Z_5[12]$, we can resolve ($\Delta X_5[4]$-$\Delta Y_5[4]$) to get $X_5[4]$.
   - We guess $Y_4[0-3,12-15]$ and compute corresponding $X_4[0-3,12-15]$ in the backward direction and $W_4[4-7]$ in the forward direction. This allows us to calculate $K_5[4-7]$ and deduce the corresponding $K_4[12-14]$. We use this to compute $W_3[12-14]$.
   - We store $X_4[0-3,12-15]||X_5[4-7]$ at index value $W_3[12-14]||\Delta X_4[0-2,12-15]$ in table T_1. There are about 2^{32} entries for each index.
3. For each of the 2^{128} index of K_3 in table T_0, we have 2^{64} entries of $W_3[12-14]||\Delta X_4[0-2,12-15]$ and corresponding to each of these we have 2^{32} entries of $X_4[0-3,12-15]||X_5[4-7]$ in table T_1. So in all, after merging T_0 and T_1, we get $2^{128+64+32}=2^{224}$ unique set of 39-byte parameters, that are required to construct the multiset v.
4. For each of these 2^{224} 39-byte parameters, we calculate the corresponding 52-byte parameters for all the elements of the δ-list and compute the multiset $v=\{u^0\oplus u^0,\ u^1\oplus u^0,\ \dots,\ u^{255}\oplus u^0\}$. We store the multiset along with the 52-byte parameters in the table T.

The time complexity to construct [5] $T_0=2^{(16+8+7)\times 8}\times 2^{-2.17}=2^{245.83}$. The time complexity to construct $T_1=2^{(3+3+8)\times 8}\times 2^{-2.17}=2^{109.83}$. The time complexity to merge T_0 and $T_1=2^{128+64+32}=2^{224}$. Finally, the time complexity to construct $T=2^{224}\times 2^{8}\times 2^{-0.58}=2^{231.41}$.

4.2 Online Phase

In this phase we extend the differential trail described in Section 3, by adding 3 more rounds at the bottom (as shown in Fig. 4). The steps of the online phase are as follows:

1. We encrypt 2^{97} structures of 2^8 plaintexts each where byte 15 takes all possible values and rest of the bytes are constants. We store the corresponding ciphertexts in the hash table.
2. For each of the 2^{112} $(P_0,\ P_0')$ plaintext pairs, do the following:
   - We guess 2^{128} values of K_9 and deduce the corresponding values of K_8 from K_9. We decrypt each of the ciphertext pairs through 2 rounds, to get X_7 and ΔX_7. Then, we deduce the corresponding ΔW_6 and ΔZ_6.
   - We filter out the keys, which do not give zero difference at $\Delta Z_6[0-4,12-15]$. 2^{56} key guesses are expected to remain.
   - We pick one member of the pair, say P_0, create the δ-list by constructing the rest of the 255 plaintexts as $P_i=P_0\oplus i$, where, $1\le i\le 255$ and get their corresponding ciphertexts.
   - For each remaining 2^{56} key guesses of K_8 and K_9, we guess $U_7[5-11]$, compute the corresponding $Z_6[5-11]$ and $Y_6[8-11,13-15]$ and then obtain the multiset $\{\ u^0\oplus u^0,\ u^1\oplus u^0,\ \dots,\ u^{255}\oplus u^0\}$.
   - We check whether this multiset exists in the precomputation table T or not. If not, then we discard the corresponding guesses.

The probability for a wrong guess to pass the test is $2^{224}\times 2^{-467.6}=2^{-243.6}$. [6] Since we try only $2^{112+56}=2^{168}$ multisets, only the right subkey should verify the test.

4.3 Recovering the remaining Subkey bytes

The key schedule algorithm of Kalyna does not allow recovery of master key from any subkey better than brute-force [11]. However, knowledge of all round keys enables encryption/decryption. We follow a similar approach as described in [1] to recover all the round subkeys. When a match with a multiset is found using a given plaintext-ciphertext pair, we choose one of the ciphertexts and perform the following steps:

1. We already know the corresponding K_8 and K_9 and U_7[5-11].
2. We guess the remaining 9 bytes of U_7, and deduce the corresponding 2^{72} values of K_7 and K_6.

[5] The normalization factor $2^{-2.17}$ is calculated by finding the ratio number of rounds encrypted/decrypted to 9 (i.e., the number of rounds of Kalyna considered in this paper). Similarly all other normalisation factors have been calculated.

[6] Note that the probability of randomly having a match is $2^{-467.6}$ and not $2^{-505.17}$ since the number of ordered sequences associated to a multiset is not constant [6].

3. For each 2^{72} guesses of (K_7, K_6), from X_7 we compute X_5. We discard the key guesses for which $X_5[4-7]$ does not match with the values of $X_5[4-7]$ obtained from the corresponding matched multiset in the pre-computation table.
4. For the remaining $2^{72-32} = 2^{40}$ guesses of (K_9, K_8, K_7, K_6), we guess 2^{128} values of K_5. We deduce X_4 and discard the key guesses for which X_4[0-2, 12-15] does not match with the values obtained corresponding to the correct multiset sequence from the precomputation table. From a total of $2^{128+40} = 2^{168}$ key guesses, 2^{112} key guesses are expected to remain.
5. We deduce K_4 from K_5 for the remaining key guesses and compute X_3. We compare this to the value obtained from the precomputation table corresponding to the correct multiset sequence and discard those that do not match. Only one value of (K_9, K_8, K_7, K_6, K_5, K_4) is expected to remain.
6. One value of K_3 and K_2 corresponding to the matching sequence is already known from the pre-computation table. We deduce X_1 for the remaining one value of (K_9, K_8, K_7, K_6, K_5, K_4, K_3, K_2).
7. We guess 2^{128} values of K_1, deduce K_0 and compute the plaintext. We compare this to the plaintext corresponding to ciphertext being decrypted. We are left with only one value of (K_9, K_8, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0).

Complexities. The time complexity of the precomputation phase is dominated by step 1 and is $2^{248} \times 2^{-2.17} = 2^{245.83}$ Kalyna-128/256 encryptions. The time complexity of the online phase is dominated by step 2 (part 1) and is $2^{112} \times 2^{128} \times 2^{-2.17} = 2^{233.83}$. The time complexity of the Subkey recovery phase is dominated by step 4 which is $2^{168} \times 2^{-3.17} = 2^{164.83}$. Clearly the time complexity of the whole attack is dominated by the time complexity of the precomputation phase, i.e., $2^{245.83}$. It was shown in [5] that each 256-byte multiset requires 512-bits space. Hence, to store each entry in table T, we require 512-bits to store the multiset and $52 \times 8 = 416$-bits to store the 52-byte parameters, i.e., a total of 928-bits ($= 2^{9.86}$). Therefore, the memory complexity of this attack is $2^{224} \times 2^{9.86-7} = 2^{226.86}$ Kalyna 128-bit blocks. The data complexity of this attack is 2^{105} plaintexts.

5 Construction of distinguisher for 6-Round Kalyna-256/512

In this section, we construct a distinguisher for the 6-inner rounds of Kalyna-256/512. The distinguisher construction details are similar to Kalyna-128/256 (discussed in Sec. 3) except the fact that here instead of counting multisets, we count 256-byte ordered sequences. The reason for opting ordered sequences would be discussed in Sec. 6.

We first establish the following relation for Kalyna-256/512. According to *Property 2*, it is possible to construct an equation using any 12 out of 16 input-output bytes in the Kalyna MixColumn operation. For any round j, where, $0 \leq j \leq 8$:

$$\begin{aligned} Z_j[8] \oplus Z_j[9] \oplus Z_j[12] \oplus Z_j[13] = {} & EA_x \cdot W_j[8] \oplus 54_x \cdot W_j[9] \oplus 7D_x \cdot W_j[10] \oplus C3_x \cdot W_j[11] \\ & \oplus E0_x \cdot W_j[12] \oplus 5E_x \cdot W_j[13] \oplus 7D_x \cdot W_j[14] \\ & \oplus C3_x \cdot W_j[15] \end{aligned} \tag{13}$$

Derivation of Eq. 13 is shown in Appendix C. Similar to as shown in Section 3, since, $W_j = K_j \oplus X_{j+1}$, if

$$P_j = Z_j[8] \oplus Z_j[9] \oplus Z_j[12] \oplus Z_j[13] \tag{14}$$

$$\begin{aligned} Q_j = {} & EA_x \cdot X_{j+1}[8] \oplus 54_x \cdot X_{j+1}[9] \oplus 7D_x \cdot X_{j+1}[10] \oplus C3_x \cdot X_{j+1}[11] \oplus E0_x \cdot X_{j+1}[12] \\ & \oplus 5E_x \cdot X_{j+1}[13] \oplus 7D_x \cdot X_{j+1}[14] \oplus C3_x \cdot X_{j+1}[15] \end{aligned} \tag{15}$$

$$\begin{aligned} Const = {} & EA_x \cdot K_j[8] \oplus 54_x \cdot K_j[9] \oplus 7D_x \cdot K_j[10] \oplus C3_x \cdot K_j[11] \oplus E0_x \cdot K_j[12] \oplus 5E_x \cdot K_j[13] \\ & \oplus 7D_x \cdot K_j[14] \oplus C3_x \cdot K_j[15] \end{aligned} \tag{16}$$

then, Eq. 13 can be rewritten as,

$$P_j = Q_j \oplus Const \tag{17}$$

5.1 Construction of 6-round distinguisher for Kalyna-256/512

Given a list of 256 distinct bytes (M^0, M^1, ... , M^{255}), a function $f : \{0,1\}^{256} \mapsto \{0,1\}^{256}$ and a 248-bit constant T, we define an ordered sequence ov as follows:

$$C^i = f(T \mid\mid M^i), \text{where } (0 \leq i \leq 255) \tag{18}$$

$$ou^i = EA_x \cdot C^i[8] \oplus 54_x \cdot C^i[9] \oplus 7D_x \cdot C^i[10] \oplus C3_x \cdot C^i[11] \oplus E0_x \cdot C^i[12] \oplus 5E_x \cdot C^i[13] \oplus 7D_x \cdot C^i[14] \oplus C3_x \cdot C^i[15] \tag{19}$$

$$ov = \{ou^0 \oplus ou^0, ou^1 \oplus ou^0, \ldots, ou^{255} \oplus ou^0\} \tag{20}$$

Note that, ($T \mid\mid M^0$, $T \mid\mid M^1$, ... , $T \mid\mid M^{255}$) forms a δ-list and the first element of ov (i.e., $ou^0 \oplus ou^0$) is always zero.

Distinguishing Property. Let us consider $\mathcal{F}$ to be a family of permutations on 256-bit. Then, given any list of 256 distinct bytes (M^0, M^1, ... , M^{255}), the aim is to find how many ordered sequences ov (as defined above) are possible when, $f \xleftarrow{\$} \mathcal{F}$ and $T \xleftarrow{\$} \{0,1\}^{248}$.

In case, when $\mathcal{F}$ = family of all permutations on 256-bit and $f \xleftarrow{\$} \mathcal{F}$. Under such setting, since, ov is a 256-byte ordered sequence in which the first byte is always zero and the rest 255 bytes are chosen uniformly and independently from the set {0, 1, ... , 255}, the total possible values of ov are $(256)^{255} = 2^{2040}$.

In case, when $\mathcal{F}$ = 6-full rounds of Kalyna-256/512 and $f \xleftarrow{\$} \mathcal{F}$. Here, $f \xleftarrow{\$} \mathcal{F} \Leftrightarrow K \xleftarrow{\$} \{0,1\}^{512}$ and $f = E_K$. Let us consider the first 6 inner rounds of Kalyna-256/512 as shown in Fig. 5. Here, C in Eq. 18 is represented by X_6 and Eq. 19 is defined as :

$$ou^i = EA_x \cdot X_6^i[8] \oplus 54_x \cdot X_6^i[9] \oplus 7D_x \cdot X_6^i[10] \oplus C3_x \cdot X_6^i[11] \oplus E0_x \cdot X_6^i[12] \oplus 5E_x \cdot X_6^i[13] \oplus 7D_x \cdot X_6^i[14] \oplus C3_x \cdot X_6^i[15] \tag{21}$$

It is to be noted that here, for each i where, $(0 \leq i \leq 255)$, Eq. 21 is same as Eq. 15 computed at round 5, i.e., $ou^i = Q_5^i$. Now, we state the following *Observation 4*.

Observation 4. The ordered sequence ov is determined by the following 93 single byte parameters only :

- $X_0^0[31]$ (1-byte)
- $X_1^0[16-23]$ (8-bytes)
- $X_2^0[0-31]$ (32-bytes)
- $X_3^0[0-31]$ (32-bytes)
- $X_4^0[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29]$ (16-bytes)
- $X_5^0[8, 9, 28, 29]$ (4-bytes)

Thus, the total number of ordered sequences is $2^{93\times 8} = 2^{744}$ since each 93-byte value defines one sequence.

Proof. In round 0 (in Fig. 5), the ordered list of differences at $\{X_0^0[31] \oplus X_0^0[31], X_0^1[31] \oplus X_0^0[31], \ldots, X_0^{255}[31] \oplus X_0^0[31]\}$ (or, equivalently the list of differences at $X_0[31]$) is known to the attacker as the list [7] of differences at $X_0[31]$ = list of differences at $P[31]$, i.e., $P^i[31] \oplus P^0[31] = X_0^i[31] \oplus X_0^0[31]$ for $(1 \leq i \leq 255)$. This is so, because in the plaintext, we make the most significant byte as the active byte. Hence, when the pre-whitening key is added (columnwise), the carry-bit in the most significant bit is ignored, thus converting the addition operation to xor operation. Since the value of $X_0^0[31]$ is known, the attacker can compute the other $X_0^i[31]$. This allows her to cross the *SB* and *SR* layer in round 0. Since, *MixColumn* (MC) and *Add Round Key* (ARK) are linear operations, the list of differences at $X_1[16-23]$ can be computed by the attacker.

Owing to the non-linearity of the S-box operation, the list of differences at $Y_1[16-23]$ cannot be computed to move forward. To alleviate this problem, it is sufficient to guess $X_1^0[16-23]$ as it allows calculating other $X_1^i[16-23]$ states and cross SB layer in round 1. Since, SR, MC and ARK operations are linear, the list of differences at $X_2[0-31]$ is known. Continuing in a similar manner as discussed above, if the attacker guesses full states $X_2^0[0-31]$ and $X_3^0[0-31]$, then the list of differences at Z_3, i.e., $\{Z_3^0 \oplus Z_3^0, Z_3^1 \oplus Z_3^0, \ldots, Z_3^{255} \oplus Z_3^0\}$ can be easily computed.

[7] From now onwards, list denotes an ordered list

This also allows her to calculate the list of differences at $X_4[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$. By guessing $X_4^0[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$, the attacker can cross the SB layer in round 4 and calculate the list of differences at $X_5[8,9,28,29]$. By guessing X_5^0 [8, 9, 28, 29], the attacker can obtain the list of values $\{Z_5^0[8,9,12,13], Z_5^1[8,9,12,13], \ldots, Z_5^{255}[8,9,12,13]\}$. Using these, she can compute P_5^i at Z_5^i using Eq. 14 and thus the list $\{P_5^0 \oplus P_5^0, P_5^0 \oplus P_5^1, \ldots, P_5^{255} \oplus P_5^0\}$. Since, according to Eq. 17, $P_5^i \oplus P_5^0 = Q_5^i \oplus Q_5^0$ and $ou^i = Q_5^i$ (mentioned above), the attacker can easily calculate the ordered sequence $ov = \{Q_5^0 \oplus Q_5^0, Q_5^1 \oplus Q_5^0, \ldots, Q_5^{255} \oplus Q_5^0\}$. This shows that ov depends on 93 parameters and can take 2^{744} possible values. $\square$

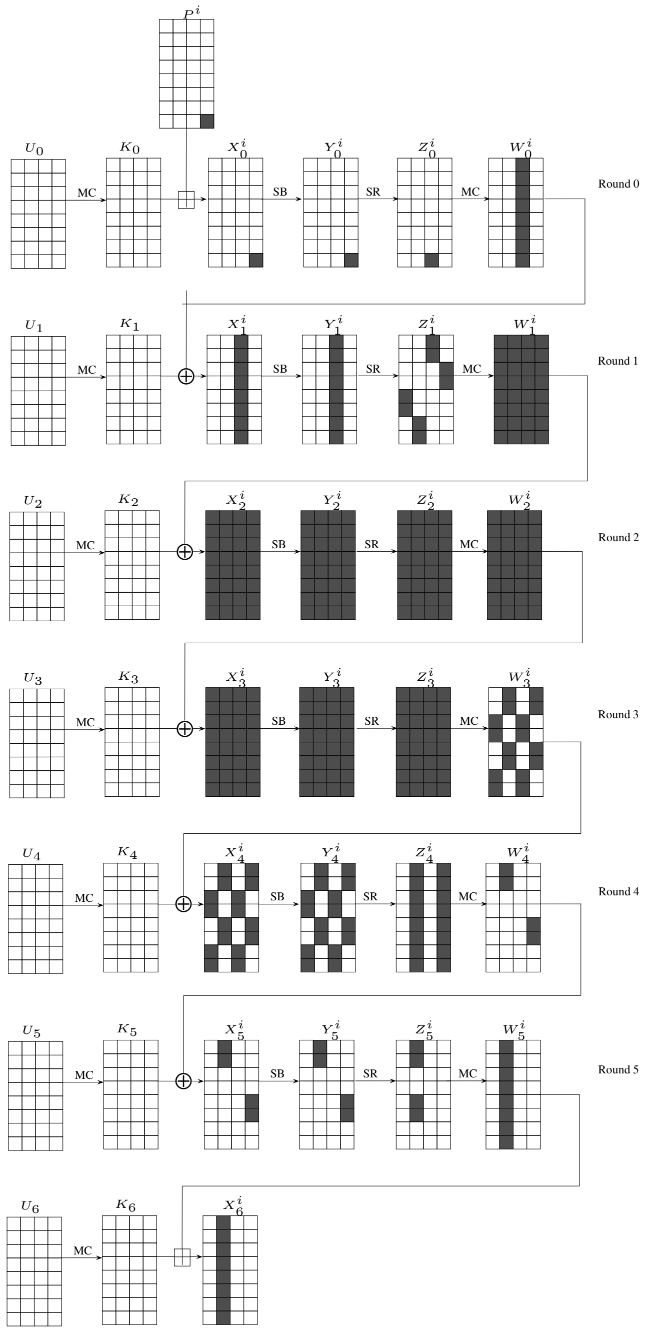

Fig. 5. 6-Round distinguisher in Kalyna-256/512 . Here, P^i denotes $(T \mid\mid M^i)$ and X_j^i, Y_j^i, Z_j^i, W_j^i denote intermediate states corresponding to P^i in round j. The round subkeys K_j, where, $0 \leq j \leq 6$ are generated from the master key K.

Since, there are 2^{744} possible ordered sequences, if we precompute and store these values in a hash table, then the precomputation complexity goes higher than brute force for Kalyna-256/512. In order to reduce the number of ordered sequences, we apply the Refined Differential Enumeration technique as follows:

Number of admissible ordered sequences. Consider the 6-round truncated differential trail in round 0 - round 5 (as shown in Fig. 5) where, the input difference is non-zero at one byte and output difference is non zero in 8 bytes. The probability of such a trail is 2^{-224} as follows: the one byte difference at $\Delta P[31]$ propagates to 32-byte difference in $\Delta Z_3[0-31]$ with probability 1. Next, the probability that 32-byte difference in $\Delta Z_3[0-31]$ propagates to 16-byte difference in $\Delta X_4[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$ is 2^{-128}. This 16-byte difference in ΔX_4 propagates to 4-byte difference in $\Delta W_4[8,9,28,29]$ followed by 8-byte difference in $\Delta W_5[8-15]$ with a probability of 2^{-96}. Thus, the overall probability of the differential trail from ΔP to ΔW_5 is $2^{-(128+96)} = 2^{-224}$.

In other words, we require 2^{224} plaintext pairs to get a right pair. Once, we get a right pair, say (P^0, P^1), we state the following *Observation 5.*

<u>Observation 5.</u> Given a right pair (P^0, P^1) that follows the truncated differential trail $(\Delta P \rightarrow \Delta W_5)$, the 93 parameters corresponding to P^0, mentioned in Observation 4 can take one of atmost 2^{440} fixed 93-byte values (out of the total 2^{744} possible values), where each of these 2^{440} 93-byte values are defined by each of the 2^{440} values of the following 66 parameters:

- $Y_0^0[31]$ (1-byte)
- $\Delta Z_0[23]$ (1-byte)
- $X_1^0[16-23]$ (8-bytes)
- $Y_3^0[0-31]$ (32-bytes)
- $Y_4^0[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$ (16-bytes)
- $Y_5^0[8,9,28,29]$ (4-bytes)
- $\Delta Z_5[8,9,12,13]$ (4-bytes)

Proof. Given a right pair (P^0, P^1), the knowledge of these 66 new parameters allows us to compute all the differences shown in Fig. 5as follows. Knowledge of $Y_0^0[31]$ allows us to compute $X_0^0[31]$. Knowing $\Delta Z_0[23]$ allows one to compute the difference $\Delta X_1[16-23]$. Then, if the values of $X_1^0[16-23]$ are known, one can compute the corresponding $X_1^1[16-23]$ and compute ΔX_2.

From the bottom side, knowing $\Delta Z_5[8,9,12,13]$ allows one to compute $\Delta Y_5[8,9,28,29]$. Knowledge of Y_5^0 $[8,9,28,29]$ allows one to compute $Y_5^1[8,9,28,29]$, cross the SB layer in round 5 and obtain $\Delta Y_4^0[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29]$. Proceeding in a similar manner, knowing $Y_4^0[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29]$ and $Y_3^0[0-31]$ allows one to compute $\Delta Y_2^0[0-31]$. Then, using *Property 1a.*, the possible values of X_2^0 and Y_2^0 can be computed. At this stage, the total possible values of these 65 parameters are $2^{66\times 8} = 2^{528}$.

However, for each value of this 66-byte parameter, the following key bytes - $U_2[4,5,14,15,16,17,26,27]$, K_3, $K_4[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$ and $K_5[8,9,28,29]$ can be deduced as follows:

1. Knowledge of $X_1^0[16-23]$ allows us to compute the corresponding $Z_1^0[4,5,14,15,16,17,26,27]$. Xoring these values with $X_2^0[4,5,14,15,16,17,26,27]$ helps us in deducing $U_2[4,5,14,15,16,17,26,27]$.
2. Knowledge of X_2^0 and Y_3^0 helps us in deducing K_3.
3. Knowledge of Y_3^0 and $Y_4^0[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$ can be used to deduce $K_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29]$.
4. Knowledge of $Y_4^0[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$ and $Y_5^0[8,9,28,29]$ helps us in deducing K_5 $[8,9,28,29]$.

Now, according to the key schedule algorithm of Kalyna-128/256 from K_3, we can compute K_2 (according to Eq. 1) which allows us to compute the corresponding U_2. Thus, by comparing the computed $U_2[4, 5, 14, 15, 16, 17, 26, 27]$ with the deduced $U_2[4,5,14,15,16,17,26,27]$, a sieve of 8-bytes (since matching probability is 2^{-64}) can be applied. Similarly, knowledge of $K_4[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$ allows us to compute K_5 $[8, 28, 29]$ which can then be matched with the deduced $K_5[8,28,29]$. This allows us a filtering of further 3-bytes. Thus, by key sieving, a total of 11-byte filtering can be applied and the possible guesses of 66-byte parameter reduces from $2^{66\times 8}$ to $2^{(66-11)\times 8} = 2^{55\times 8} = 2^{440}$. □

Using *Observation 4* and *Observation 5*, we state the following third *Observation 6*:

<u>Observation 6.</u> Given $(M^0, M^1, \ldots, M^{255})$ and $f \overset{\$}{\leftarrow} \mathcal{F}$ and $T \overset{\$}{\leftarrow} \{0,1\}^{248}$, such that $T \mid\mid M^0$ and $T \mid\mid M^j$, (where, $j \in \{1, \ldots, 255\}$) is a right pair that follows the differential trail shown in Fig. 5, atmost 2^{440} multisets v are possible.

Proof. From *Observation 4*, we know that each 93-byte parameter defines one ordered sequence and *Observation 5* restricts the possible values of these 93-byte parameters to 2^{440}. Thus, atmost 2^{440} ordered sequences are only possible for Kalyna-256/512. □

As the number of ordered sequences in case of 256-bit random permutation ($= 2^{2040}$) is much higher than 6-round Kalyna-256/512 ($= 2^{440}$), a valid distinguisher is therefore constructed.

6 Key Recovery Attack on 9-Round Kalyna-256/512

In this section, we use our Observation 6 to launch meet-in-the-middle attack on 9-round Kalyna-256/512 to recover the key. The distinguisher is placed in round 0 to round 5 (as shown in Fig. 6) and three rounds are added at the bottom of the 6-round distinguisher. The attack consists of the following three phases:

6.1 Precomputation Phase

In this phase, we build a lookup table T to store 2^{440} sequences to be used for comparison in the online phase. The construction of this table requires us to create two more hash tables (T_0 and T_1) in the intermediate steps. The entire procedure is as follows:

1. For each K_3
   - We guess $\Delta Z_1[4,5,14,15,16,17,26,27]$ || ΔX_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29] to compute ΔX_2 and ΔY_3 respectively. We resolve (ΔX_2 - ΔY_3) using *Property 1b* to compute the corresponding $X_2||Y_3$. We then deduce K_2 from K_3 and compute the corresponding value of $Z_1[4,5,14,15,16,17,26,27]$. Using the guessed value of $\Delta Z_1[4,5,14,15,16,17,26,27]$ and the computed value of $Z_1[0-3,12-15]$, we compute $\Delta Z_0[16-23]$. If $\Delta Z_0[16-22] = 0$ (which happens with a probability of 2^{-56}), we store the corresponding $X_1[16-23]$ || $\Delta Z_1[4,5,14,15,16,17,26,27]$ || X_2 || X_3 || $W_3[7,8,19]$ || ΔX_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29] at index K_3 in table T_0. There are about 2^{136} entries for each index.
2. For each guess of $\Delta Z_5[8,9,12,13]$ || $Y_5[8,9,28,29]$, compute $X_5[8,9,28,29]$, $\Delta W_4[8,9,28,29]$ and ΔY_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29]. Guess Y_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29] to compute X_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29] and $\Delta X_4[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$ in the backward direction and $W_4[8,9,28,29]$ in the forward direction. From, $W_4[8,9,28,29]$ and $X_5[8,9,28,29]$ compute $K_5[8,9,28,29]$. Deduce $K_4[7,8,19]$ (where, $K_5[8] = K_4[19]$, $K_5[28] = K_4[7]$ and $K_5[29] = K_4[8]$). Using, $X_4[7,8,19]$ and $K_4[7,8,19]$, compute $W_3[7,8,19]$.
3. For each entry of $W_3[7,8,19]$ || $\Delta X_4[2,3,6,7,8,9,12,13,18,19,22,23,24,25,28,29]$, we store X_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29] || $X_5[8,9,28,29]$ in a table T_1. There are 2^{40} entries per index.
4. For each of the 2^{256} index of K_3 in table T_0, we have 2^{136} entries of $W_3[7,8,19]$ || ΔX_4 [2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29] and corresponding to each of these we have 2^{40} entries of X_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29] || $X_5[8,9,28,29]$ in table T_1. So in all, after merging T_0 and T_1, we get $2^{256+136+40} = 2^{432}$ unique set of 65-byte parameters mentioned in *Observation 5.*
5. For each guess of $X_0[31]$, combine the above merged entries with $X_0[31]$ to complete the set of 66-parameters mentioned in *Observation 5.* Now, there are a total of $2^{432+8} = 2^{440}$ entries.
6. For each of these 2^{440} 66-byte parameters, we calculate the corresponding 93-byte parameters for all the elements of the δ-list and compute the ordered sequence $ov = \{ou^0 \oplus ou^0,\ ou^1 \oplus ou^0,\ \ldots,\ ou^{255} \oplus ou^0\}$. We store the ordered sequence along with the 93-byte parameters in the table T.

The time complexity to construct $T_0 = 2^{(32+8+16)\times 8} \times 2^{-2.17} = 2^{445.83}$. The time complexity to construct $T_1 = 2^{(4+4+16)\times 8} \times 2^{-2.17} = 2^{189.83}$. The time complexity to merge T_0 and T_1 along with each guess of $X_0[31] = 2^{256+136+40+8} = 2^{440}$. Finally, the time complexity to construct $T = 2^{440} \times 2^8 \times 2^{-0.58} = 2^{447.42}$. Hence, overall time complexity is $2^{445.83} + 2^{447.42} \approx 2^{447.83}$.

6.2 Online Phase

In this phase we extend the distinguisher in Section 5, by adding 3 more rounds at the bottom (as shown in Fig. 6). The steps of the online phase are as follows:

1. We encrypt 2^{209} structures of 2^8 plaintexts each where byte 31 takes all possible values and rest of the bytes are constants. We store the corresponding ciphertexts in the hash table.

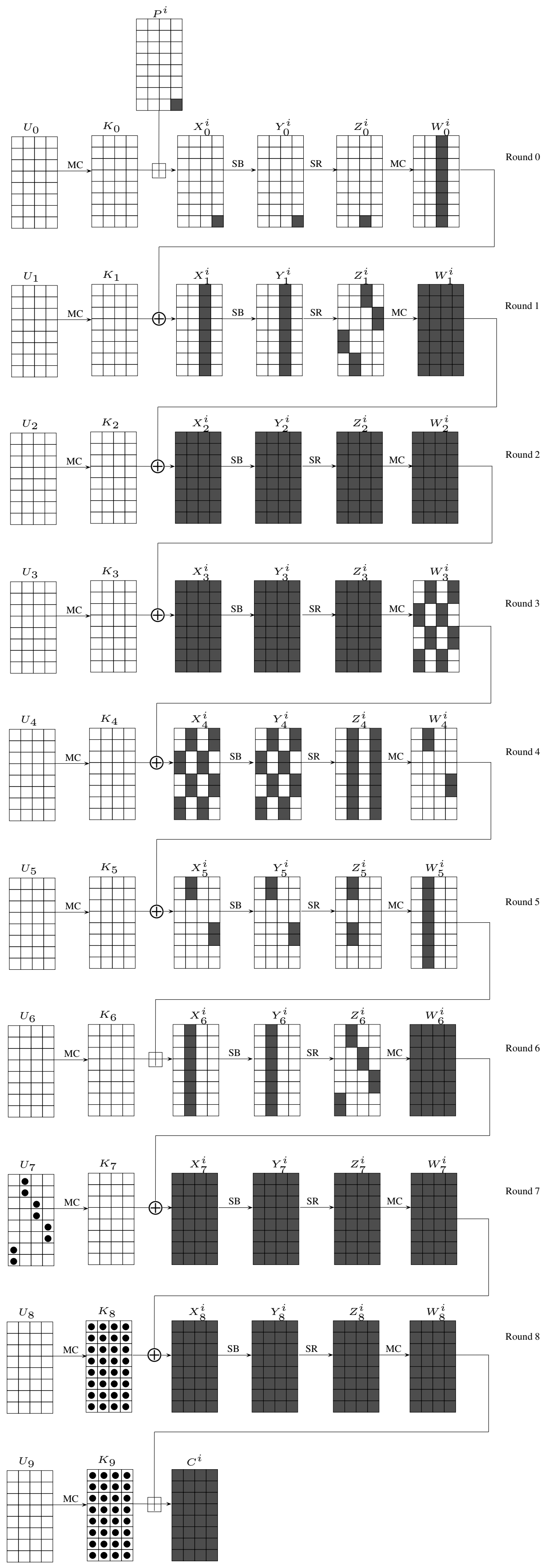

Fig. 6. 9-round attack on Kalyna-256/512. The subkey bytes guessed are shown dotted.

2. For each of the 2^{224} (P_0, P_0') plaintext pairs, do the following:
   - We guess 2^{256} values of K_9 and deduce the corresponding values of K_8 from K_9. We decrypt each of the ciphertext pairs through 2 rounds, to get X_7 and ΔX_7. Then, we deduce the corresponding ΔW_6 and ΔZ_6.
   - We filter out the keys, which do not give zero difference at $\Delta Z_6[0-5, 10-17, 20-27, 30, 31]$. This creates a filtering of 2^{-192} and hence only 2^{64} key guesses are expected to remain.
   - We pick one member of the pair, say P_0, create the δ-list by constructing the rest of the 255 plaintexts as $P_i = P_0 \oplus i$, where, $1 \leq i \leq 255$ and get their corresponding ciphertexts.
   - For each of the remaining 2^{64} key guesses of K_8 and K_9, we guess $U_7[6, 7, 8, 9, 18, 19, 28, 29]$, compute the corresponding $Z_6[6, 7, 8, 9, 18, 19, 28, 29]$ and $Y_6[8-15]$ and then obtain the ordered sequence { $ou^0 \oplus ou^0$, $ou^1 \oplus ou^0$, …, $ou^{255} \oplus ou^0$}.
   - We check whether this sequence exists in the precomputation table T or not. If not, then we discard the corresponding guesses.

Reason for counting ordered sequences instead of multisets . The probability for a wrong guess to pass the test is $2^{440} \times 2^{-2040} = 2^{-1600}$. Since we try only $2^{224+64} = 2^{288}$ ordered sequences, only the right subkey should verify the test.

If we had opted for mutliset attack on Kalyna-256/512, the total possible admissible multisets would have been 2^{432} (as the parameter $X_0[31]$ would not have been required). Therefore, the probability for a wrong guess to pass the test would have been $2^{432} \times 2^{-467.6} = 2^{-35.6}$ (similar to that described in Section 4). As mentioned above, since we try 2^{288} multisets, we would have got mutliple candidates for the right subkey and unable to recover the secret key.

6.3 Recovering the remaining Subkey bytes

The remaining subkeys recovery process is similar to that discussed in Section 4.3. When a match with an ordered sequence is found using a given plaintext-ciphertext pair, we choose one of the ciphertexts and perform the following steps:

1. We already know the corresponding K_8 and K_9 and $U_7[6, 7, 8, 9, 18, 19, 28, 29]$.
2. We guess the remaining 24 bytes of U_7, and deduce the corresponding 2^{192} values of K_7 and K_6.
3. For each 2^{192} guesses of (K_7, K_6), from X_7 we compute X_5. We discard the key guesses for which $X_5[8, 9, 28, 29]$ does not match with the values of $X_5[8, 9, 28, 29]$ obtained from the corresponding matched ordered sequence in the pre-computation table.
4. For the remaining $2^{192-32} = 2^{160}$ guesses of (K_9, K_8, K_7, K_6), we guess 2^{256} values of K_5. We deduce X_4 and discard the key guesses for which $X_4[2, 3, 6, 7, 8, 9, 12, 13, 18, 19, 22, 23, 24, 25, 28, 29]$ does not match with the values obtained corresponding to the correct ordered sequence from the precomputation table. From a total of $2^{160+256} = 2^{416}$ key guesses, 2^{288} key guesses are expected to remain.
5. We deduce K_4 from K_5 for the remaining key guesses and compute X_3. We compare this to the value obtained from the precomputation table corresponding to the correct ordered sequence and discard those that do not match. 2^{32} values of (K_9, K_8, K_7, K_6, K_5, K_4) are expected to remain.
6. One value of K_3 and K_2 corresponding to the matching sequence is already known from the pre-computation table. We deduce X_1 for the remaining 2^{32} values of (K_9, K_8, K_7, K_6, K_5, K_4, K_3, K_2).
7. We guess 2^{256} values of K_1, deduce K_0 and compute $X_0[23]$ and plaintext. We compare this to $X_0[23]$ obtained from the precomputation table and to the plaintext corresponding to ciphertext being decrypted respectively. We are left with only 2^{24} values of (K_9, K_8, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0). We search these exhaustively to find the correct set of subkeys.

Complexities. The time complexity of the precomputation phase is $2^{447.83}$ Kalyna-128/256 encryptions. The time complexity of the online phase is dominated by step 2(part 1) and is $2^{224} \times 2^{256} \times 2^{-2.17} = 2^{477.83}$. The time complexity of the Subkey recovery phase is dominated by step 4 which is $2^{160} \times 2^{256} \times 2^{-3.17} = 2^{412.83}$. Clearly the time complexity of the whole attack is dominated by the time complexity of the online phase, i.e., $2^{477.83}$. It was shown in [5] that each 256-byte multiset requires 512-bits space. Hence, to store each entry in table T, we require 2048-bits to store the ordered sequence and $93 \times 8 = 744$-bits to store the 52-byte parameters, i.e., a total of 928-bits ($= 2^{11.45}$). Therefore, the memory complexity of this attack is $2^{440} \times 2^{11.45-8} = 2^{443.45}$ Kalyna 256-bit blocks. The data complexity of this attack is 2^{217} plaintexts.

7 Conclusions

In this work, we utilize multiset attacks to launch key recovery attack on Kalyna-128/256 and Kalyna-256/512. We improve the previous 7-round attack on both the variants to demonstrate the first 9-round attacks on the same. Our attacks on Kalyna-256/512 even improve upon the previous 7-round attack on the same variant in terms of time and data complexities. We obtain these results by constructing new 6-round distinguishers on Kalyna and applying MITM attack on the rest of the rounds. Currently, this line of attack only works on Kalyna-b/2b variants and Kalyna variants in which block size and key size are equal appear to be safe. It would be an interesting problem to try applying multiset attacks on Kalyna-b/b. Presently, all five variants of Kalyna have been included in the Ukranian standard. However, our results as well as the previous 7-round attack show that compared to Kalyna-b/2b variants, Kalyna-b/b variants appear to be more robust.

References

1. Riham AlTawy, Ahmed Abdelkhalek, and Amr M. Youssef. A Meet-in-the-Middle Attack on Reduced-Round Kalyna-b/2b. *IACR Cryptology ePrint Archive*, 2015:762, 2015. `http://eprint.iacr.org/2015/762`.
2. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Information Security and Cryptography. Springer, 2002.
3. Joan Daemen and Vincent Rijmen. Understanding two-round differentials in AES. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, volume 4116 of *Lecture Notes in Computer Science*, pages 78–94. Springer, 2006.
4. Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008.
5. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 371–387. Springer, 2013.
6. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. *J. Cryptology*, 28(3):397–422, 2015.
7. Kishan Chand Gupta and Indranil Ghosh Ray. On constructions of MDS matrices from companion matrices for lightweight cryptography. In Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar R. Weippl, Lida Xu, Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar R. Weippl, and Lida Xu, editors, *Security Engineering and Intelligence Informatics - CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings*, volume 8128 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2013.
8. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full whirlpool compression function. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, volume 5912 of *Lecture Notes in Computer Science*, pages 126–143. Springer, 2009.
9. Leibo Li, Keting Jia, and Xiaoyun Wang. Improved single-key attacks on 9-round AES-192/256. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 127–146. Springer, 2014.
10. Roman Oliynykov. Next Generation of Block Ciphers Providing High-Level Security, June 2015. `http://http://www.slideshare.net/oliynykov/next-generation-ciphers/`.
11. 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. *IACR Cryptology ePrint Archive*, 2015:650, 2015. `http://eprint.iacr.org/2015/650`.
12. Li Rongjia and Jin Chenhui. Meet-in-the-middle attacks on 10-round AES-256. *Designs, Codes and Cryptography*, pages 1–13, 2015.

A One round of Kalyna when MixColumn and Key Xoring operations are interchnaged

Here, we show how one internal round of Kalyna works when *MixColumn* and *AddRoundKey* operations are interchanged.

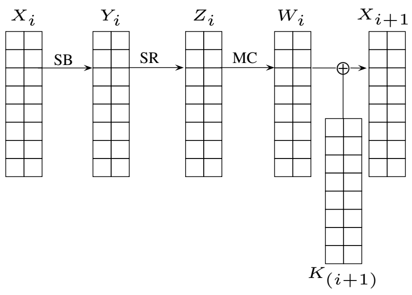

Fig. 7. Normal one round of Kalyna-128/128.

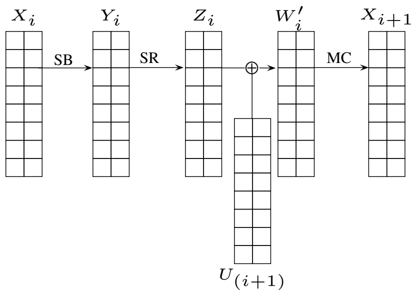

Fig. 8. One round of Kalyna-128/128 with swapped MC and ARK operation. Here, $W'_i = Z_i \oplus U_{i+1}$.

B Derivation of Eq. 3 defined in Section 3

$$\begin{pmatrix} AD_x & 95_x & 76_x & A8_x & 2F_x & 49_x & D7_x & CA_x \\ CA_x & AD_x & 95_x & 76_x & A8_x & 2F_x & 49_x & D7_x \\ D7_x & CA_x & AD_x & 95_X & 76_x & A8_x & 2F_x & 49_x \\ 49_x & D7_x & CA_x & AD_x & 95_x & 76_x & A8_x & 2F_x \\ 2F_x & 49_x & D7_x & CA_x & AD_x & 95_x & 76_x & A8_x \\ A8_x & 2F_x & 49_x & D7_x & CA_x & AD_x & 95_x & 76_x \\ 76_x & A8_x & 2F_x & 49_x & D7_x & CA_x & AD_x & 95_x \\ 95_x & 76_x & A8_x & 2F_x & 49_x & D7_x & CA_x & AD_x \end{pmatrix} \times \begin{pmatrix} W_j[8] \\ W_j[9] \\ W_j[10] \\ W_j[11] \\ Uk_1 \\ W_j[13] \\ W_j[14] \\ W_j[15] \end{pmatrix} = \begin{pmatrix} Uk_2 \\ Uk_3 \\ Uk_4 \\ Uk_5 \\ Z_j[12] \\ Z_j[13] \\ Z_j[14] \\ Z_j[15] \end{pmatrix}$$

Using Inverse Mix Column operation, $Z_j[12]$, $Z_j[13]$, $Z_j[14]$ and $Z_j[15]$ can be written as:

$$2F_x \cdot W_j[8] \oplus 49_x \cdot W_j[9] \oplus D7_x \cdot W_j[10] \oplus CA_x \cdot W_j[11] \oplus AD_x \cdot Uk_1 \oplus 95_x \cdot W_j[13] \oplus 76_x \cdot W_j[14] \oplus A8_x \cdot W_j[15] = Z_j[12] \quad (22)$$

$$A8_x \cdot W_j[8] \oplus 2F_x \cdot W_j[9] \oplus 49_x \cdot W_j[10] \oplus D7_x \cdot W_j[11] \oplus CA_x \cdot Uk_1 \oplus AD_x \cdot W_j[13] \oplus 95_x \cdot W_j[14] \oplus 76_x \cdot W_j[15] = Z_j[13] \quad (23)$$

$$76_x \cdot W_j[8] \oplus A8_x \cdot W_j[9] \oplus 2F_x \cdot W_j[10] \oplus 49_x \cdot W_j[11] \oplus D7_x \cdot Uk_1 \oplus CA_x \cdot W_j[13] \oplus AD_x \cdot W_j[14] \oplus 95_x \cdot W_j[15] = Z_j[14] \quad (24)$$

$$95_x \cdot W_j[8] \oplus 76_x \cdot W_j[9] \oplus A8_x \cdot W_j[10] \oplus 2F_x \cdot W_j[11] \oplus 49_x \cdot Uk_1 \oplus D7_x \cdot W_j[13] \oplus CA_x \cdot W_j[14] \oplus AD_x \cdot W_j[15] = Z_j[15] \quad (25)$$

If we combine the above equations in the following way:

$$CA_x \cdot (Eq\ 22) \oplus AD_x \cdot (Eq\ 23) \oplus 49_x \cdot (Eq\ 24) \oplus D7_x \cdot (Eq\ 25) \quad (26)$$

We can eliminate the unknown variable Uk_1 and obtain:

$$\begin{aligned} 94_x \cdot W_j[8] \oplus B4_x \cdot W_j[9] \oplus 4E_x \cdot W_j[10] \oplus 7E_x \cdot W_j[11] &= CA_x \cdot Z_j[12] \oplus AD_x \cdot Z_j[13] \\ \oplus C0_x \cdot W_j[13] \oplus DA_x \cdot W_j[14] \oplus C5_x \cdot W_j[15] \quad &\oplus 49_x \cdot Z_j[14] \oplus D7_x \cdot Z_j[15] \end{aligned} \quad (27)$$

C Derivation of Eq. 13 defined in Section 5

$$\begin{pmatrix} AD_x & 95_x & 76_x & A8_x & 2F_x & 49_x & D7_x & CA_x \\ CA_x & AD_x & 95_x & 76_x & A8_x & 2F_x & 49_x & D7_x \\ D7_x & CA_x & AD_x & 95_X & 76_x & A8_x & 2F_x & 49_x \\ 49_x & D7_x & CA_x & AD_x & 95_x & 76_x & A8_x & 2F_x \\ 2F_x & 49_x & D7_x & CA_x & AD_x & 95_x & 76_x & A8_x \\ A8_x & 2F_x & 49_x & D7_x & CA_x & AD_x & 95_x & 76_x \\ 76_x & A8_x & 2F_x & 49_x & D7_x & CA_x & AD_x & 95_x \\ 95_x & 76_x & A8_x & 2F_x & 49_x & D7_x & CA_x & AD_x \end{pmatrix} \times \begin{pmatrix} W_j[8] \\ W_j[9] \\ W_j[10] \\ W_j[11] \\ W_j[12] \\ W_j[13] \\ W_j[14] \\ W_j[15] \end{pmatrix} = \begin{pmatrix} Z_j[8] \\ Z_j[9] \\ Uk_1 \\ Uk_2 \\ Z_j[12] \\ Z_j[13] \\ Uk_3 \\ Uk_4 \end{pmatrix}$$

Using Inverse Mix Column operation, $Z_j[8]$, $Z_j[9]$, $Z_j[12]$ and $Z_j[13]$ can be written as:

$$AD_x \cdot W_j[8] \oplus 95_x \cdot W_j[9] \oplus 76_x \cdot W_j[10] \oplus A8_x \cdot W_j[11] \oplus 2F_x \cdot W_j[12] \oplus 49_x \cdot W_j[13] \oplus D7_x \cdot W_j[14] \oplus CA_x \cdot W_j[15] = Z_j[8]$$

$$CA_x \cdot W_j[8] \oplus AD_x \cdot W_j[9] \oplus 95_x \cdot W_j[10] \oplus 76_x \cdot W_j[11] \oplus A8_x \cdot W_j[12] \oplus 2F_x \cdot W_j[13] \oplus 49_x \cdot W_j[14] \oplus D7_x \cdot W_j[15] = Z_j[9]$$

$$25_x \cdot W_j[8] \oplus 49_x \cdot W_j[9] \oplus D7_x \cdot W_j[10] \oplus CA_x \cdot W_j[11] \oplus AD_x \cdot W_j[12] \oplus 95_x \cdot W_j[13] \oplus 76_x \cdot W_j[14] \oplus A8_x \cdot W_j[15] = Z_j[12]$$

$$A8_x \cdot W_j[8] \oplus 25_x \cdot W_j[9] \oplus 49_x \cdot W_j[10] \oplus D7_x \cdot W_j[11] \oplus CA_x \cdot W_j[12] \oplus AD_x \cdot W_j[13] \oplus 95_x \cdot W_j[14] \oplus 76_x \cdot W_j[15] = Z_j[13]$$

If we xor together the above four equations, then we get

$$Z_j[8] \oplus Z_j[9] \oplus Z_j[12] \oplus Z_j[13] = EA_x \cdot W_j[8] \oplus 54_x \cdot W_j[9] \oplus 7D_x \cdot W_j[10] \oplus C3_x \cdot W_j[11] \oplus E0_x \cdot W_j[12] \oplus 5E_x \cdot W_j[13] \oplus 7D_x \cdot W_j[14] \oplus C3_x \cdot W_j[15]$$