Cryptanalysis of the Round-Reduced Kupyna Hash Function

Jian Zou[1,2], Le Dong[3]

[1]Mathematics and Computer Science of Fuzhou University, Fuzhou, China, 350108
[2]Key Lab of Information Security of Network Systems (Fuzhou University), Fuzhou, China, China, 350108
[3] College of Mathematics and Information Science, Henan Normal University, Xinxiang, China, 453007
zoujian@fzu.edu.cn, dongle127@163.com

Abstract. The Kupyna hash function was selected as the new Ukrainian standard DSTU 7564:2014 in 2015. It is designed to replace the old Independent States (CIS) standard GOST 34.311-95. The Kupyna hash function is an AES-based primitive, which uses Merkle-Damgård compression function based on Even-Mansour design. In this paper, we show the first cryptanalytic attacks on the round-reduced Kupyna hash function. Using the rebound attack, we present a collision attack on 5-round of the Kupyna-256 hash function. The complexity of this collision attack is $(2^{120}, 2^{64})$ (in time and memory). Furthermore, we use guess-and-determine MitM attack to construct pseudo-preimage attacks on 6-round Kupyna-256 and Kupyna-512 hash function, respectively. The complexity of these preimage attacks are $(2^{250.33}, 2^{250.33})$ and $(2^{498.33}, 2^{498.33})$ (in time and memory), respectively.

Key words: Kupyna, preimage attack, collision attack, rebound attack, meet-in-the-middle, guess-and-determine

1 Introduction

Cryptographic hash functions are playing important roles in the modern cryptography. They have many important applications, such as authentication and digital signatures. In general, hash function must satisfy three security requirements: preimage resistance, second preimage resistance and collision resistance. In the last few years, the cryptanalysis of hash functions has been significantly improved. After the pioneering work of Wang [26–28], there is a strong need for a secure and efficient hash function. In 2015, the Kupyna hash function[17] was approved as the new Ukrainian standard DSTU 7564:2014. In addition, GOST R 34.11-2012 [18] was selected as the new Russian National hash function standard.

Ukraine had been using the Commonwealth of Independent States (CIS) standard GOST R 34.11-94 [19] as the main cryptographic hash function until 2015. GOST R 34.11-94 [19] was theoretically broken in 2008 [13, 12]. As a result, the Kupyna hash function is designed to replace the old Russian standard

GOST R 34.11-94 that no longer fits performance and security requirements. The Kupyna hash function was approved in 2015, which is widely used in Ukraine. Note that for the remainder of this article, we refer to the Kupyna hash function simply as Kupyna.

Kupyna is an iterated hash function based on the wide-pipe Merkle-Damgård design. It uses Davies-Meyer compression function based on Even-Mansour block cipher construction. The compression function of Kupyna employs an SPN structure following the AES design strategy. Kupyna supports output length from 8 bits to 512 bits. The recommended modes are Kupyna-256, Kupyna-384 and Kupyna-512.

With respect to the collision attack, the rebound attack proposed by Mendel *et al.* [14] is very effective with the differential attack against AES based structure. From then on, many techniques are proposed to improve the original rebound attack such as start-from-the-middle technique [11], linearized match-in-the-middle technique [11], Super-Sbox analysis [7, 5], and multiple-inbound technique [10]. At FES 2014, Mendel *et al.*[15] constructed a 5-round collision attack on Grøstl-256.

At FSE 2008, Leurent [8] proposed the preimage attack on the full MD4 hash function. From then on, many techniques are proposed to improve the preimage attacks. One of them is the meet-in-the-middle (MitM) preimage attack with the splice-and-cut technique. This method is first proposed by Aoki and Sasaki to attack MD4 [2]. The MitM attack preimage attacks have been applied to many hash function such as HAVAL-3/4 [20], MD4 [8, 2], MD5 [21], Tiger [6], RIPEMD [25], SHA-0/1 [3], and SHA-2 [1, 6]. In CRYPTO 2009, Aoki and Sasaki [3] combined the MitM attack with the guessing of carry bits technique to improve their preimage attack on SHA-0 and SHA-1. In FSE 2012, Wu *et al.* [29] improved its complexity and proposed the first pseudo preimage attack on Grøstl. Using the combination of the guess-and-determin and the MitM attack, Sasaki *et al.* [22] improved the preimage attacks on Whirlpool in AsiaCrypt 2012. In addition, Zou *et al.* [30] combined the guess-and-determin with the MitM attack to propose an improved pseudo-preimage attack on Grøstl.

In this paper, we combine the guess-and-determine technique with the MitM to propose a pseudo-preimage attack on 6-round Kupyna-256 and an 8-round pseudo-preimage attack on Kupyna-512. Then we construct a 5-round collision attack on Kupyna-256. Our cryptanalytic results of Kupyna are summarized in Table 1. We will explain these results in Section 3 and 4.

Outline of the paper The rest of the paper is organized as follows. We give a short description of Kupyna in Section 2. In Section 3, we show our preimage attacks on the round-reduced Kupyna. Then we present the collision attack on the round-reduced Kupyna-256 in Section 4. Section 5 concludes the paper.

Table 1. Summary of Attack Results

Algorithm	Target	Attack Type	Rounds	Time	Memory	Source
Kupyna-256	Hash Function	Collision	5	2^{120}	2^{64}	Section 4
	Output Transformation	Collision	5	2^{240}	2^{64}	Section 3.2
	Hash Function	Pseudo Preimage	6	$2^{250.33}$	$2^{250.33}$	Section 3.3
Kupyna-512	Output Transformation	Preimage	8	2^{472}	2^{120}	Section 3.4
	Hash Function	Pseudo Preimage	8	$2^{498.33}$	$2^{498.33}$	Section 3.5

2 Description of Kupyna

Kupyna is an iterated hash function with an SPN structure following the AES design strategy. Kupyna supports output length from 8 bits to 512 bits. The recommended modes are Kupyna-256, Kupyna-384 and Kupyna-512. Assume a message is padded and divided into message blocks $M_0, M_1, \ldots, M_{k-1}$ of l-bit length, which is defined as

$$l = \begin{cases} 512, \ if \ 8 \leq n \leq 256, \\ 1024, \ if \ 256 < n \leq 512. \end{cases}$$

We can process as follows to generate a hash value h:

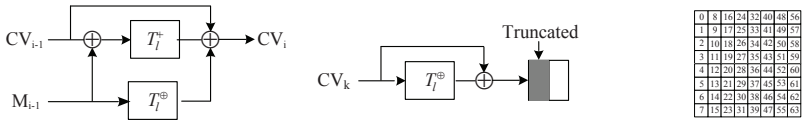
$$\begin{cases} CV_0 \leftarrow IV \\ CV_{i+1} \leftarrow CF(CV_i, M_i) \text{ for } i = 0, 1, \ldots, k-1 \\ h = Trunc(T_l^{\oplus}(CV_k) \oplus CV_k) \end{cases}$$

Here the IV is the initial value(If $l = 512$, IV=1 $\ll 510$, else IV=1 $\ll 1024$.), and $CF(CV_i, M_i)$ is the compression function of Kupyna. $T_l^{\oplus}(CV_k)$ is a permutation that is a component of the compression function and will be defined later.

Before applying the compression function, the input message M is processed to be a multiple of l bits by the padding procedure. According to the padding procedure, a single bit '1' and len_0 '0's are put at the end of the message M. Here len_0 satisfies the following equation $len_M + 1 + len_0 \equiv 96 \mod 512$ (len_M and len_0 are short for the length of M and the number of '0' respectively). After the above step, we put another 96 bits including the length of the message at the end of the padding. Then the padded message M^* is divided into l bits blocks M_i $(i = 0, 1, \ldots, k-1)$.

The compression function uses two permutations $T_l^{\oplus}$ and T_l^{+}, and computes as follows(see also Fig.1):

$$CF_l(CV_i, M_i) = T_l^{\oplus}(CV_i \oplus M_i) \oplus T_l^{+}(M_i) \oplus CV_i.$$



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

Fig. 1. Compression function, output transformation of Kupyna, and the byte positions for Kupyna-256.

Here, CV_i is the chaining value and M_i is a message block. Note that, the compression function CF_l has only two forms CF_{512} (for $8 \leq n \leq 256$) and CF_{1024} (for $256 < n \leq 512$). In the following, we only present Kupyna-256 and Kupyna-512 for simplicity. $T_l^{\oplus}()$ and $T_l^{+}()$ are AES-like permutation with 8×8 and 8×16 sized state for Kupyna-256 and Kupyna-512 respectively. As shown in Fig. 1, byte positions in a state S for Kupyna-256/512 are denoted by $G = (g_{i,j})$, $g_{i,j} \in GF(2^8)$, where $i = 0, 1, ..., 7$, $j = 0, 1, ..., c-1$ (if $l = 512, c = 8$; if $l = 1024, c = 16$). Kupyna-256 adopts 10-round $T_{512}^{\oplus}()$ and $T_{512}^{+}()$. Kupyna-512 uses 14-round $T_{1024}^{\oplus}()$ and $T_{1024}^{+}()$. The round function of the two permutations consist of the following four operations:

- **AddConstant:** The AddConstant operation adds a round-dependent constant to the state.
- **SubBytes:** The SubBytes transformation applies an S-box to each cell of the state.
- **ShiftRows:** The ShiftRows transformation cyclically rotates the cells of the i-th row leftwards by shift vector (define later).
- **MixColumns:** In the MixColumns operation, each column of the matrix is multiplied by an MDS matrix.

We use AC, SB, SR and MC to denote these transformations for short. The SB function transforms each cell $g_{i,j}$ of the state matrix $G = (g_{i,j})$ by $SB_{i \ mod \ 4}(g_{i,j})$, where $SB_s : GF(2^8) \rightarrow GF(2^8), s \in 0, 1, 2, 3)$. We omitted the detail of the SB, since its not important in our attack. The shift vectors used in $T_l^{\oplus}()$ and $T_l^{+}()$ are different. $T_{512}^{\oplus}()$ and $T_{512}^{+}()$ in Kupyna-256 uses (0,1,2,3,4,5,6,7), while $T_{1024}^{\oplus}()$ and $T_{1024}^{+}()$ in Kupyna-512 uses (0,1,2,3,4,5,6,11). For a detailed explanation, we refer to the original paper [17]. Since the design of Kupyna is similar to Grøstl, we can apply some attacks on Grøstl to Kupyna.

3 Preimage Attack on Kupyna

In this section, we will show how to construct a preimage attack on the Kupyna hash function. Firstly, we present the previous preimage attacks on Grøstl. Secondly, we show how to construct the preimage attacks on Kupyna. Note that, if we make $CV_i^{'} = CV_i \oplus M_i$, then compression function can be rewritten as

$$CF(CV_i, M_i) = T_l^{\oplus}(CV_i^{'}) \oplus CV_i^{'} \oplus T_l^{+}(M_i) \oplus M_i.$$

This important property will be used in our preimage attack on Kupyna.

3.1 Previous Works

In FSE 2012, Wu *et al.*[29] proposed the first pseudo preimage attack on Grøstl hash function. In [30], Zou *et al.* found out the attack of Wu *et al.* could be divided into two-phase MitM attacks, then they could use the subspace preimage to improve the complexity. Subspace preimage attack can be defined by a linear subspace (with a linear function $L(\cdot)$). In subspace preimage attack process, the attacker should store the $L(A)$ in the lookup table L_{Sub}, and check if there exists an entry in L_{Sub} that $L(A)$ is equal to $L(B)$. If they find one, $L(A \oplus B)$ is equal to 0, that means $A \oplus B$ is in the linear subspace. Compared with b-bit partial preimage, the subspace preimage could achieve some balance between the two-phase MitM attacks, and then reduce the overall complexity.

The idea of Zou*et al.* can be summarized as below. Suppose the hash output is n-bit and the state size is $2n$-bit. To find a pseudo preimage (H, M) of Grøstl, it is desirable to invert the output transformation of Grøstl. Let $X = CF(H, M)$, then X is the preimage of the output transformation. With $H^{'} = H \oplus M$, they get

$$(P(H^{'}) \oplus H^{'}) \oplus (Q(M) \oplus M) \oplus X = 0.$$

If enough candidates for $P(H^{'}) \oplus H^{'}$, $Q(M) \oplus M$, and X have been collected, the pseudo preimage attack turns into a three-sum problem. The attack process is similar to the generalized birthday attack[24]. Using four parameters x_1, x_2, x_3 and b, the attack process can be described as follow:

1. Find 2^{x_1} preimages X of the output transformation and store them in a lookup table L_1.
2. Find 2^{x_3} subspace preimages of $P(H^{'}) \oplus H^{'}$. Then store $P(H^{'}) \oplus H^{'}$ in a lookup table L_2.
3. Find 2^{x_2} random M with the correct padding and calculate $Q(M) \oplus M$. Check if there exists an entry in L_1 that $Q(M) \oplus M \oplus X$ is in the same subspace of $P(H^{'}) \oplus H^{'}$. Then $2^{x_1+x_2-b}$ partial matches $Q(M) \oplus M \oplus X$ are expected to remain (Here the dimension of the linear equation $L(x) = t_0$ is $n - b$).
4. For each of the $2^{x_1+x_2-b}$ $Q(M) \oplus M \oplus X$ remained in step 3, check whether there exists an entry in L_2 that matches the remaining $(2n - b)$ bits. Once a full match is found, a pseudo preimage of Grøstl hash function is found.

Suppose for Grøstl with $2n$-bit state, it needs $2^{C_1(2n,n)}$ computations to find a fixed position n-bit partial preimage of X and it needs $2^{C_2(2n,b)}$ computations to find a subspace preimage of $P(X) \oplus X$. Then the overall complexity to compute the pseudo preimage of Grøstl is:

$$2^{x_1+C_1(2n,n)} + 2^{x_3+C_2(2n,b)} + 2^{x_2-1} + 2^{x_1+x_2-b} \cdot C_{TL}, \tag{1}$$

with memory requirement of $2^{x_1} + 2^{x_3}$. Here C_{TL} is chosen as 1/640 and 1/2048 for 5-round Grøstl-256 and 8-round Grøstl-512 respectively. For a detailed explanation of the attack, we refer to the original paper [18]. Note that, we will use the same C_{TL} in the preimage attacks on Kupyna.

3.2 Preimage Attack on 6-round Kupyna-256 Output Transformation

Using the combination of the guess-and-determine technique and the MitM attack, we can construct 6-round preimage attacks on the compression function of Kupyna-256. The 6-round chunk separation of the state update transformation is illustrated in Fig. 2. By guessing some unknown bytes, all the possible values of the guessed bytes are used as extra freedom degrees in the MitM preimage attack. As a result, more matching points can be obtained. Note that the guessed bytes are extra constraints. After a partial match is found, we should check if the guessed value produces a valid preimage. More details about the guessing technique will be shown in the following attack algorithm.

Parameters for the Guess-and-Determine MitM Attack As shown in Fig. 2, we use the purple bytes as the guessed bytes. The red/blue color means neutral message. They are independent from each other. The white color stands for the bytes whose values are affected by red bytes and blue bytes both, and we can't determine their values until a partial match is found. The gray bytes are constants that come from the hash value or the initial structure. In order to evaluate the complexity for the attack, we should define these parameters: freedom degrees in red and blue bytes (d_r, d_b), the guessing red and blue bytes (D_{gr}, D_{gb}), the bits of the matching point b_m.

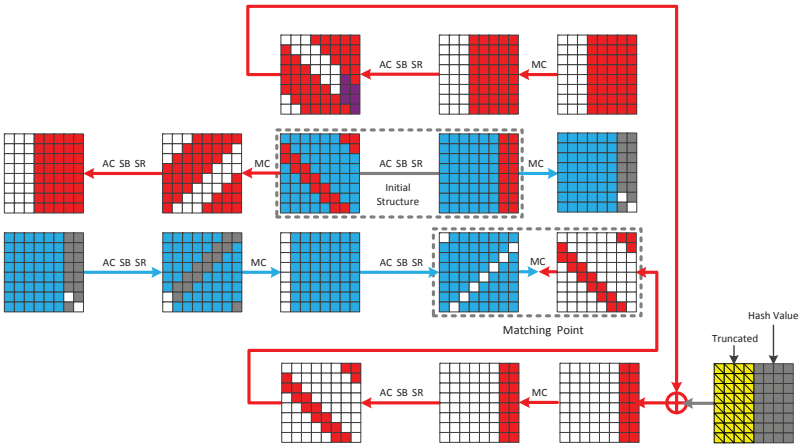


Fig. 2. Preimage attack on the Output Transformation of Kupyna-256

The Attack Algorithm and Complexity The guess-and-determine MitM attack algorithm can be described as follows:

1. Set random values to constants in the initial structure.
2. For all possible values 2^{d_r} of the red bytes and $2^{D_{gr}}$ of the guessing red bytes, compute backward from the initial structure and obtain the value at the matching point. Store the values in a lookup table L_{comp}.
3. For all possible values 2^{d_b} of the blue bytes and $2^{D_{gb}}$ of the guessing blue bytes, compute forward from the initial structure and obtain the value at the matching point. Check if there exists an entry in L_{comp} that matches the result at the matching point. Expected number of the partial matches is $2^{d_r+d_b+D_{gr}+D_{gb}-b_m}$.
4. Once a partial match is found, compute and check if the guessed value is right. The probability of the validity is $2^{-D_{gr}-D_{gb}}$. There are $2^{d_r+d_b-b_m}$ valid partial matches left. Then we continue the computation and check the full match. The probability that a partial match is a full match is $2^{-(n-b_m)}$.
5. The success probability for the above steps is $2^{d_r+d_b+D_{gr}+D_{gb}-b_m} \cdot 2^{-(D_{gb}+D_{gr})} \cdot 2^{-(n-b_m)} = 2^{d_r+d_b-n}$. Then repeat the above steps for $2^{n-d_b-d_r}$ to find one full match.

The complexity for each step can be calculated as follows:

1. In Step 2, building the the lookup table L_{comp} takes $2^{d_r+D_{gr}}$ computations and memory.
2. In Step 3, it takes $2^{d_b+D_{gb}}$ computations to find the partial matches. Expected number of the partial matches is $2^{d_b+D_{gb}+d_r+D_{gr}-b_m}$.
3. In Step 4, testing all the partial matches in step 3 needs $2^{d_b+D_{gb}+d_r+D_{gr}-b_m}$ computations. The probability of the validity is $2^{-D_{gb}-D_{gr}}$, and there are $2^{d_b+d_r-b_m}$ valid partial matches left.
4. In Step 5, repeat the above four step for $2^{n-d_b-d_r}$ times.

Then the complexity of the above attack algorithm is:

$$\begin{aligned} & 2^{n-d_b-d_r} \cdot (2^{d_r+D_{gr}} + 2^{d_b+D_{gb}} + 2^{d_b+D_{gb}+d_r+D_{gr}-b_m}) \\ & = 2^n \cdot (2^{D_{gr}-d_b} + 2^{D_{gb}-d_r} + 2^{D_{gb}+D_{gr}-b_m}). \end{aligned} \tag{2}$$

As shown in Fig. 2, the parameters for the attack on the 6-round compression function of Kupyna-256 are as follow: $d_r = 16$, $d_b = 64$, $D_{gr} = 48$, $D_{gb} = 0$, $b_m = 64$ and $n = 256$. According to equation (1), the overall complexity is $2^{256} \cdot (2^{48-64} + 2^{0-16} + 2^{48-64}) \approx 2^{240}$ compression function calls. Only Step 2 requires $2^{16+48} = 2^{64}$ memory.

3.3 Subspace Preimage Attack on $T^{\oplus}_{512}(H') \oplus H'$

In [30], Zou *et al.* has shown that the guess-and-determine technique is not fit for the b-bit subspace preimage attack when b is small. Our preimage attack on $T^{\oplus}_{512}(H') \oplus H'$ adopts the technique used in [30]. The attack is obtained by combining the MitM attack with a complicated initial structure. The detail of the initial structure is shown as follows (also in Fig. 3):

1. Randomly set the State value #6[1,3,5,7,8,10,12,14,17,19,21,23,24,26,28,30](gray).
2. Set the value of the State #4[0,5,6,7,8,9,14,15,16,17,18,23,24,25,26,27] (blue) so that the chosen 4 bytes at State #3[7,14,21,28] (red) can be achieved through the Inverse-MixColumns operation.
3. For each value of the State #4[0,5,6,7,8,9,14,15,16,17,18,23,24,25,26,27] (blue), we can calculate through the SubBytes and ShiftRows operations and get the corresponding values of the State #5[0,1,2,3,8,9,10,15,16,17,22,23,24,29,30,31] (blue).
4. With the known values of State #5 (blue) and State #6 (gray), we calculate the rest blue bytes of State #5 and State #6 through the MixColumns operation.
5. With the calculated bytes #5[4,5,6,7,11,12,13,14,18,19,20,21,25,26,27,28], we compute the values of #4[33,34,35,36,42,43,44,45,51,52,53,54,60,61,62,63] through the Inverse-ShiftRows and Inverse-SubBytes operations. We check whether the values of the State #4[33,34,35,36,42,43,44,45,51,52,53,54,60,61,62,63] satisfy the linear relationship so that the chosen 4 bytes at #3[35,42,49,56] can be achieved through the Inverse-Mix-Columns operation

For Kupyna-256 compression function, we suppose that it needs 2^{C_r} computations to find a suitable initial structure for the forward direction (from #6 to #8) and it needs 2^{C_b} computations to find a suitable initial structure for the backward direction (from #6 to #4). Then the complexity to find a suitable initial structure can be written as follows:

$$\begin{aligned} & 2^{n-d_b-d_r} \cdot (2^{d_r+C_r} + 2^{d_b+C_b} + 2^{d_b+d_r-m}) \\ & = 2^n \cdot (2^{C_r-d_b} + 2^{C_b-d_r} + 2^{-m}). \end{aligned} \tag{3}$$

As shown in Fig. 3, the parameters for this attack are as follows: $C_r = C_b = 32$ bits, $d_r = d_b = 40$ bits, m= 8 bits. Then $2^8 (= 2^{40+40-32-32}/2^8)$ 8-bit subspace preimages can be found with a complexity of 2^8. It means that an 8-bit subspace preimage can be found with a complexity of $2^0 (= 2^8/2^8)$ and the complexity $2^{C_2(512,8)} = 2^0$. According to Eq. (1) and $2^{C_1(512,256)} = 2^{240}$, $2^{C_2(512,8)} = 2^0$.

The Complexity of Pseudo Preimage Attack on 6-round Kupyna-256 Hash Function The overall complexity can be expressed as $2^{x_1+240} + 2^{x_3} + 2^{x_2-1} + 2^{x_1+x_2-8} \cdot C_{TL}$. When $x_1 = 10.33, x_2 = 251.33, x_3 = 250.33, b = 8$, the minimum complexity is $2^{250.33}$ and the memory requirement is $2^{250.33}$.

3.4 Preimage Attack on 8-round Kupyna-512 Output Transformation

Our 8-round preimage attack on the output transformation of Kupyna-512 combines the guess-and-determine technique with the MitM attack. Since this attack is similar to the preimage attack on 6-round Kupyna-256 Output Transformation, we omit the details of the attack and only show the chunk separation of the 8-round preimage attack in Fig. 4.

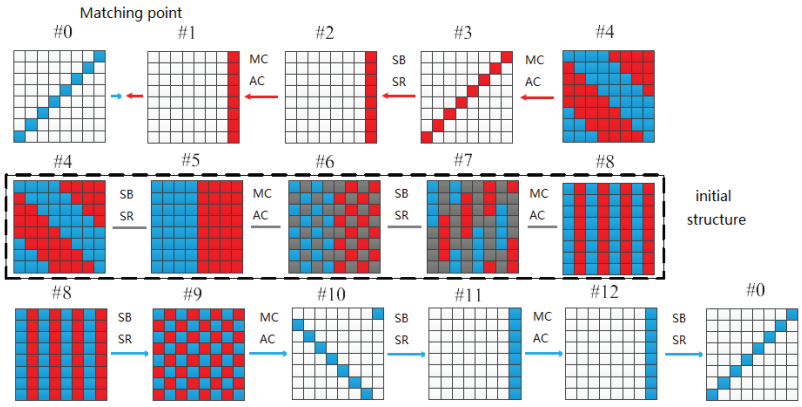


Fig. 3. Chunk separation of subspace preimage attack on 6-round Kupyna-256

In Fig. 4, the parameters for the attack on the output transformation of 8-round Kupyna-512 are as follows: $d_r = d_b = 80$ bits, $D_{gr} = D_{gb} = 40$ bits, $m = 144$ bits and $n = 512$ bits. According to Eq. (2), the complexity of our attack can be calculated as $2^{C_1(1024,512)} = 2^{512} \cdot (2^{40-80} + 2^{40-80} + 2^{40+40-144}) \approx 2^{472}$ compression function calls. The memory requirement is $2^{40+80} = 2^{120}$.

3.5 Subspace Preimage Attack on $T^{\oplus}_{1024}(H') \oplus H'$

We show the chunk separation of subspace preimage attack on $T^{\oplus}_{1024}(H') \oplus H'$ in Fig. 5. Here we also do not adopt the guess-and-determine technique here, because we cannot find a good guess-and-determine way to make the complexity better than the simple MitM attack.

The parameters for the MitM attack: the freedom degrees $d_r = 16$ bits, $d_b = 16$ bits. The size of matching point b_{max} is 32 bits. We set $d_r = d_b = b_{best} = 16$ bits, and then we can find $2^{16} (= 2^{16+16}/2^{16})$ 16-bit subspace preimages with the complexity of 2^{16}. It means that a 16-bit subspace preimage is found with the complexity of $2^0 (= 2^{16}/2^{16})$. The complexity is $2^{C2(1024,16)} \approx 2^0$.

The Complexity of Pseudo Preimage Attack on 8-round Kupyna-512 Hash Function The overall complexity can be expressed as $2^{x_1+472} + 2^{x_3} + 2^{x_2-1} + 2^{x_1+x_2-16} \cdot C_{TL}$. When $x_1 = 26.33, x_2 = 499.33, x_3 = 498.33, b = 16$, the minimum complexity is $2^{498.33}$ and the memory requirement is $2^{498.33}$.

4 Collision Attack on 5-round Kupyna-256

In a collision attack, we want to find, for a given initial value IV, two messages m and m' such that $H(IV, m) = H(IV, m')$. In this section, we will use the freedom of the message block to control the differential propagation in $T^{\oplus}_l()$,

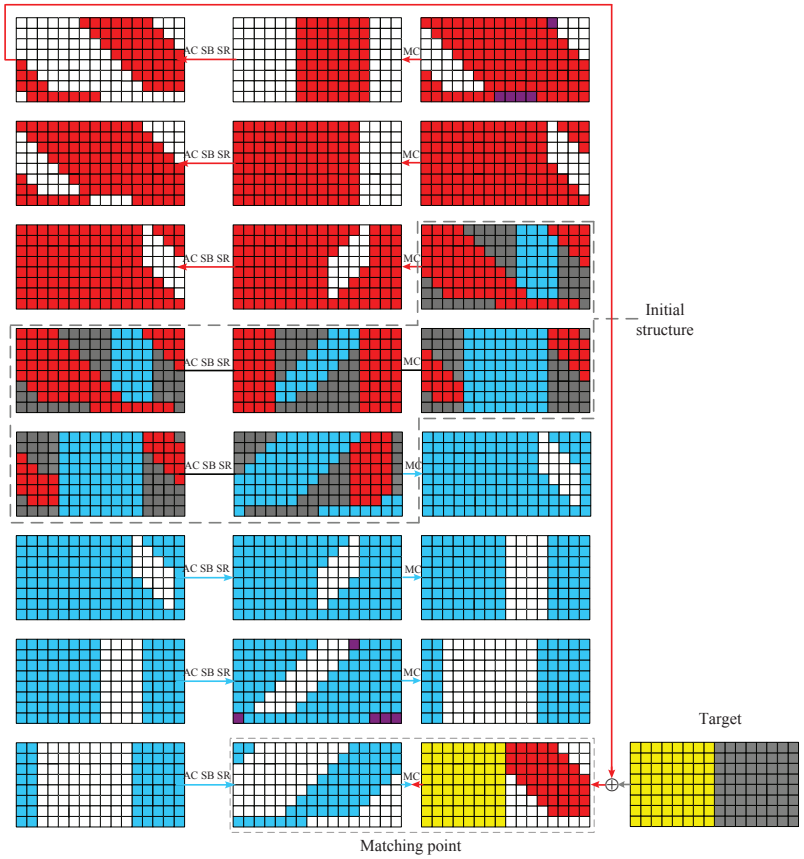


Fig. 4. Chunk separation of preimage attack on 8-round Kupyna-512 output transformation

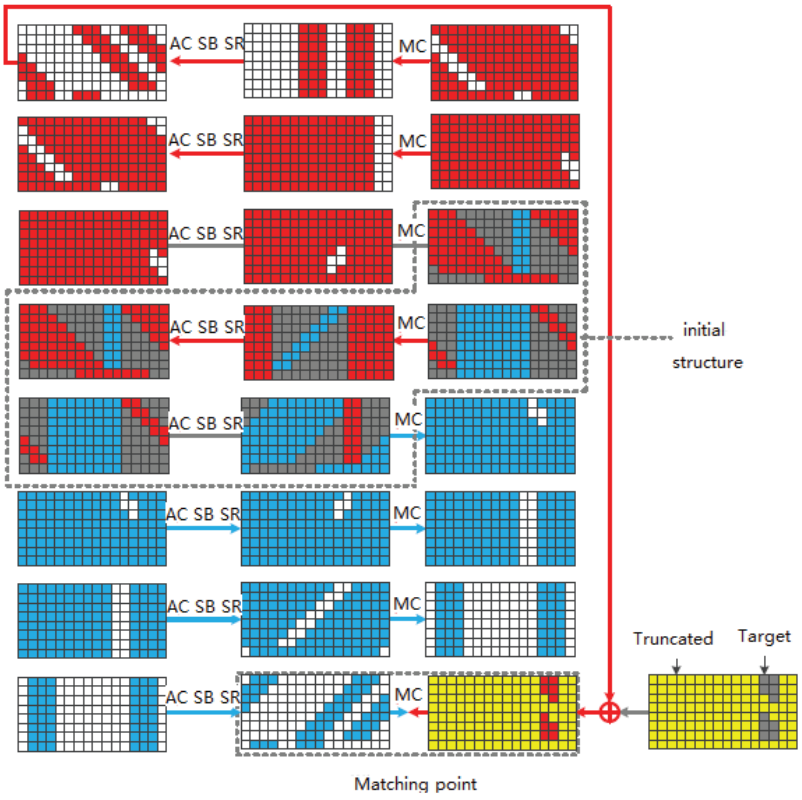


Fig. 5. Chunk separation of 8-round Subspace Preimage Attack on $T^{\oplus}_{1024}(H^{'}) \oplus H^{'}$

by setting on differences in $T_l^+()$. Based on these attacks, we then present the collision attacks on the hash function of Kupyna-256.

4.1 Details of the Attack

In order to simplify the attack process, we use the alternative description of Kupyna in the following way. Let $T_l^{\oplus'}()$ and $T_l^{+'}()$ denote the permutation $T_l^{\oplus}()$ and $T_l^{+}()$ without the last application of MC. Then, we can get an equivalent description of Kupyna, by setting:

$$\begin{cases} h_0 \leftarrow MC^{-1}(IV) \\ h_i \leftarrow T_l^{\oplus'}(h_{i-1} \oplus m_{i-1}) \oplus T_l^{+'}(m_{i-1}) \oplus h'_{i-1} \ for\ i = 1, \ldots, k \\ h = Trunc(MC(h'_k)) \end{cases}$$

We show the collision attacks on the hash function of Kupyna-256 by using the Super-Sbox technique. The Super-Sbox rebound technique was independently proposed by Lamberger *et al.* at Asiacrypto 2009 [7] and by Gilbert and Peyrin [5] at FSE 2010. The Super-Sbox consists of 8 parallel S-boxes S, followed by one MixBytes operation L and another 8 parallel S-boxes S: S-L-S.

If the differences in the message words are the same as in the output of the state update transformation, the differences cancel each other through the feed-forward. By using the freedom of the message blocks to cancel the differences of the chaining values successively, we can construct the collision attack on the Kupyna-256 hash function. Our differential path using the Super-Sbox rebound technique is shown in Fig. 6. The 5-round collision trail:

$$64 \rightarrow 64 \rightarrow 8 \rightarrow 1 \rightarrow 8 \rightarrow 8. \tag{4}$$

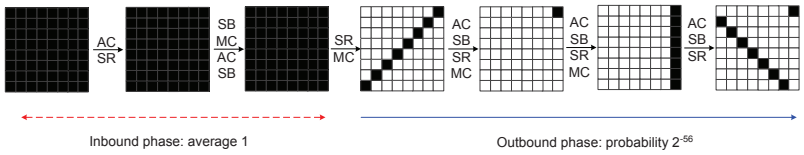


Fig. 6. Supersbox technique used in the 5-round truncated differential trail

The inbound phase is shown by dashed arrows and the outbound phase is shown by solid arrows. Note that, we should cancel the differences in 8 bytes in each iteration. The probability of this is 2^{-64}, so that we should generate 2^{64} pairs following the differential trail for a given difference. As shown in Fig. 7, the 5-round collision attack proceeds as follows:

1. Randomly choose message blocks m_0, m_0^* and compute h_1 until h_1 is fully active.

2. Use a right pair of message blocks m_1, m_1^* for the trail of (4) to cancel 8 bytes of the difference in h_1.
3. Use a right pair of message blocks m_2, m_2^* for the trail of to cancel 8 bytes of the difference in h_2.
4. Repeat steps 3-4 in 8 times until we have found a collision in h_9.

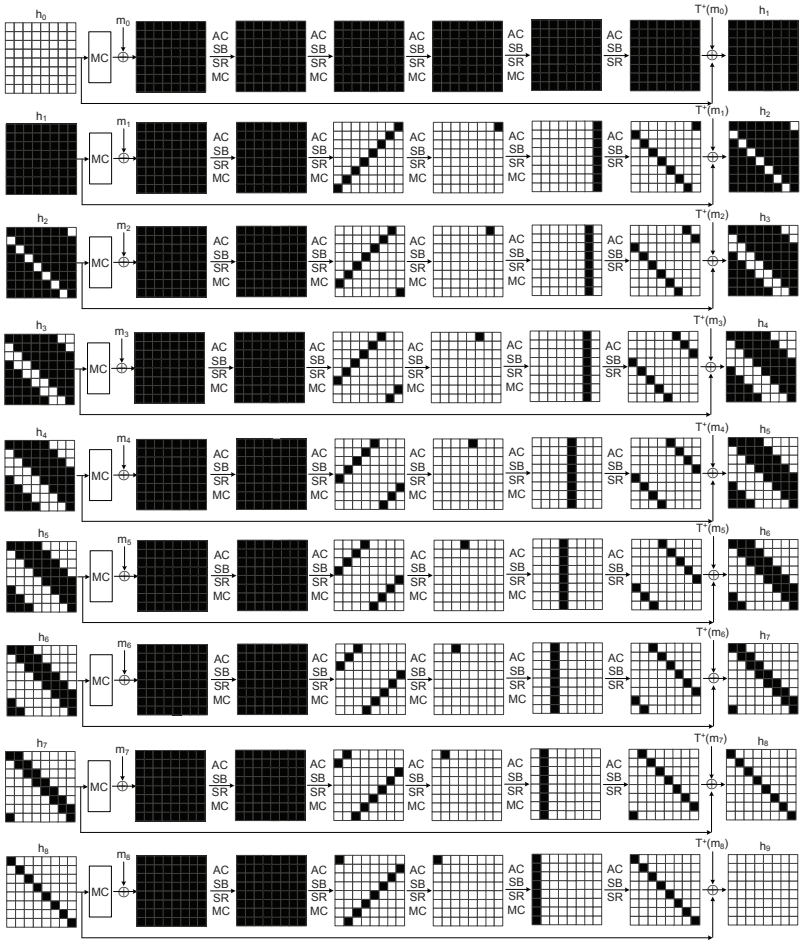


Fig. 7. Truncated differential trail used in the attack on 5 rounds

Using super-box matches, we can find 2^{64} pairs solutions for the inbound phase with a complexity of 2^{64} in time and memory. All in all, the average time complexity to generate an internal state pair that follows the differential path of the inbound phase is one. In the outbound phase, the state pairs of the

inbound phase are propagated outwards probabilistically. The transition from 8 active bytes to one active byte through the Mixcolumn transformation MC has a probability of 2^{-56}. As a result, there are only 2^8 pairs exist for this truncated differential trail. The freedom in finding right pairs for the 5-round trail is limited, more message blocks should be needed for the collision attack. As pointed in , this problem can be solved by using more message blocks in each step of the attack. The differences have to be canceled iteratively 8 times from h_2 to h_9. Then complexity of the attack is $8 \cdot 2^{64+56} = 2^{123}$, and we need the length of colliding message $8 \cdot 2^{56} = 2^{59}$. The memory requirement are 2^{64}. As noted in , we can use the tree-based approach to reduce the length of the colliding message pair to 65 message blocks. By using denser characteristics, the complexity of the 5-round collision attack can be slightly reduced to 2^{120}. Furthermore, we can construct collisions in the chosen-prefix setting with the same complexity due to the generic nature of our attack.

5 Conclusion

In this article, we propose the collision attacks and the preimage attacks on the Kupyna hash function. The design of Kupyna is similar to Grøstl, but some round constants are added with a modular addition. This design approach makes many known attacks harder to apply. The original MitM attack [2] is not suit to solve a pseudo-preimage of the Kupyna hash function due to the wide-pipe design. In addition, we can not construct the collision attack on the Kupyna hash function only by the original rebound attack. Our solution is the combination of some known attacks such as the guess-and-determine technique and the MitM preimage attack. To sum up, we present the first public security analysis of the Kupyna hash function. Firstly, we propose a pseudo-preimage attack on 6-round hash function of Kupyna-256 and a pseudo-preimage attack on 8-round hash function of Kupyna-512 by combining the guess-and-determine technique with the MitM attack. Secondly, we construct a collision attack on the 5-round Kupyna-256 hash function. However, our attacks do not threat any security claims of Kupyna.

References

1. Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for Step-Reduced SHA-2. In Matsui [9], pages 578–597.
2. Kazumaro Aoki and Yu Sasaki. Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *LNCS*, pages 103–119. Springer, 2008.
3. Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 70–89. Springer, 2009.

4. Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl – a SHA-3 candidate. Submission to NIST (Round 3), 2011.
5. Henri Gilbert and Thomas Peyrin. Super-sbox cryptanalysis: Improved attacks for aes-like permutations. In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *Lecture Notes in Computer Science*, pages 365–383. Springer, 2010.
6. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 56–75. Springer, 2010.
7. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full whirlpool compression function. In Matsui [9], pages 126–143.
8. Gaëtan Leurent. MD4 is Not One-Way. In Nyberg [16], pages 412–428.
9. 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*. Springer, 2009.
10. Krystian Matusiewicz, María Naya-Plasencia, Ivica Nikolic, Yu Sasaki, and Martin Schläffer. Rebound attack on the full lane compression function. In Matsui [9], pages 106–125.
11. Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved cryptanalysis of the reduced grøstl compression function, echo permutation and aes block cipher. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2009.
12. Florian Mendel, Norbert Pramstaller, and Christian Rechberger. A (second) preimage attack on the gost hash function. In Nyberg [16], pages 224–234.
13. Florian Mendel, Norbert Pramstaller, Christian Rechberger, Marcin Kontak, and Janusz Szmidt. Cryptanalysis of the gost hash function. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 162–178. Springer, 2008.
14. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced whirlpool and grøstl. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 260–276. Springer, 2009.
15. Florian Mendel, Vincent Rijmen, and Martin Schläffer. Collision attack on 5 rounds of grøstl. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 509–521, 2014.
16. 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*. Springer, 2008.
17. Gorbenko I. Kazymyrov O. Ruzhentsev V. Kuznetsov O. Gorbenko Y. Boiko A. Dyrda O. Dolgov V. Pushkaryov A. Oliynykov, R. A new standard of ukraine: The kupyna hash function. Cryptology ePrint Archive, Report 2015/885, 2015.
18. Information Protection and Special Communications of the Federal Security Service of the Russian Federation. Gost r 34.11.2012 information technology cryptographic date security hash-functions (in english). `http://tk26.ru/en/GOSTR3411-2012/GOST_R_34_11-2012_eng.pdf/`.

19. Information Protection and Special Communications of the Federal Security Service of the Russian Federation. Gost r 34.11.94 information technology cryptographic date security hash-functions (in russian).
20. Yu Sasaki and Kazumaro Aoki. Preimage Attacks on 3, 4, and 5-Pass HAVAL. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *LNCS*, pages 253–271. Springer, 2008.
21. Yu Sasaki and Kazumaro Aoki. Finding Preimages in Full MD5 Faster Than Exhaustive Search. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *LNCS*, pages 134–152. Springer, 2009.
22. Yu Sasaki, Lei Wang, Shuang Wu, and Wenling Wu. Investigating fundamental security requirements on whirlpool: Improved preimage and collision attacks. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 562–579. Springer, 2012.
23. Victor Shoup, editor. *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*. Springer, 2005.
24. David Wagner. A Generalized Birthday Problem. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.
25. Lei Wang, Yu Sasaki, Wataru Komatsubara, Kazuo Ohta, and Kazuo Sakiyama. (Second) Preimage Attacks on Step-Reduced RIPEMD/RIPEMD-128 with a New Local-Collision Approach. In Aggelos Kiayias, editor, *CT-RSA*, volume 6558 of *LNCS*, pages 197–212. Springer, 2011.
26. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In Shoup [23], pages 17–36.
27. Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.
28. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on sha-0. In Shoup [23], pages 1–16.
29. Shuang Wu, Dengguo Feng, Wenling Wu, Jian Guo, Le Dong, and Jian Zou. (pseudo) preimage attack on round-reduced grøstl hash function and others. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 127–145. Springer, 2012.
30. Jian Zou, Wenling Wu, Shuang Wu, and Le Dong. Improved (pseudo) preimage attack and second preimage attack on round-reduced grostl hash function. *J. Inf. Sci. Eng.*, 30(6):1789–1806, 2014.