ISBN: 978-981-18-1791-5
2021 the 11th International Workshop on Computer Science and Engineering (WCSE 2021)
doi: 10.18178/wcse.2021.06.039

Improved State-Recovery Attacks on Modified KETJE JR

Guo-Shuang Zhang [1, 2 +], Yin Li [3], Xiao Chen [1, 2]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

[3] Dongguan University of Technology, Dongguan 523000, China

Abstract. KETJE, a lightweight authenticated encryption cipher is a third-round candidate of CAESAR competition whose design principles are similar to SHA-3 hash function. Fuhr et al. studied the security of KETJE JR against divide-and-conquer attacks and proposed state-recovery attacks on modified KETJE JR. In this paper, we study the relations among the algebraic representations of internal state bits, and find new guessing strategies based on Fuhr et al.'s method. With the usage of new guessing strategies, we improve the state-recovery attacks on KETJE JR v1 when r=40 and r=32. Compared with Fuhr et al.'s work, our results are more efficient. The relults do not threaten the security of KETJE in practice, but provide evidence for improving the efficiency of KETJE by increasing the rate.

Keywords: lightweight authenticated encryption cipher, cryptanalysis, KETJE JR, state-recovery attack, divide-and-conquer attack

1. Introduction

Authenticated encryption algorithm is a cryptograpic primitive that provides both confidentiality and authentication protection, e.g. the widely used AES-GCM [1]. However, the current works demonstrate that such an algorithm may not be as strong as we hope. In particular, when the message for encryption is overlong or the initial vector is reused, its security will be seriously threatened.

In 2013, International Association for Cryptologic Research (IACR) launched the CAESAR competition to solicit potentially proper authenticated encryption algorithms with associated data (AEAD) that are required to ensure the confidentiality and integrity of a message as well as the integrity of the so-called associated data. KETJE, designed by the KECCAK team, is one of the 15 candidates for the third-round of the ongoing CAESAR competition. Its design principle relies on the Monkey Wrap [2] mode of operation, which is derived from Sponge Structure [3]. The internal permutation of KETJE is a round-reduced version of KECCAK permutation. In KETJE v1 [2], there are two versions, KETJE JR and KETJE SR, with 200-bit and 400-bit internal states respectively. In the third round of the competition, the design team proposed KETJE v2 [4]. Compared to KETJE v1, KETJE v2 includes two new variants called KETJE MINOR and KETJE MAJOR, with larger internal state sizes (800 bits and 1600 bits respectively) and a modified keystream extraction.

The candidates were opened by the competition to call for security evaluations. Since KETJE uses similar internal components as KECCAK, it has received widespread attentions from the very beginning. So far, there are many attacks on KETJE. In 2017, Dong et al. [5] described key recovery attacks on 6/7-round KETJE SR using cube-like attack. In the same year, Li et al. [6] proposed key recovery attacks on round-reduced KETJE MINOR and KETJE MAJOR based on conditional cube attack, and evaluated the security of KETJE MINOR and KETJE MAJOR with initial vectors of different length against cube attack. Song et al. [7] proposed a new Mixed-Integer Linear Programming (MILP) model for finding conditional cubes, and provided key recovery attacks on 7-round KETJE MINOR and KETJE MAJOR. In 2018, Bi et al. [8] performed key recovery attacks on KETJE MINOR and KETJE MAJOR based on MILP mode of cube-like attack which require fewer initial vectors than Ref. [6]. Song et al. [9] improved the cube-like attacks on 5-round KETJE JR and 7-round KETJE

+ Corresponding author. Tel.: + 8613911771324.
E-mail address: zhangguoshuang@iie.ac.cn.

SR based on MILP techniques. In addition, by introducing the kernel quadratic term to overcome the problem of few degrees of freedom, Li et al. [10] further improved the results of conditional cube attacks on KETJE SR. Based on the same idea, Zhou et al. [11] gave a practical key recovery attack on 5-round KETJE JR. In addition, in 2018, Fuhr et al. [12] proposed a state-recovery attack on the modified KETJE JR with the usage of the divide-and-conquer method.

At the same time, KETJE's design team launched a challenge for the security analysis of KETJE, in order to encourage experts and scholars to participate in the cryptanalysis of various variants of KETJE as well as its weakened versions. And the team has shown a special interest in state-recovery attacks, including increases in the rate of the Monkey Wrap construction.

In this paper, we study state-recovery attacks on KETJE JR v1 when the bit rates are r=40 and r=32. The main contributions are as follows: When r=40, by analyzing the relationship among the algebraic representations of the state bits in $B^2[i,i,*]$, we propose a new guessing strategy for the state C^1. While reducing the number of bits to be guessed, we can still get enough algebraic relations that satisfy the divide-and-conquer attack conditions. When r=32, we take advantage of $B^1[i,i,*]$ to establish the linear relationship between the unknown bits of $A^2[*,0,*]$ and the state needed to be recovered, and significantly reduce the search complexity of the state-recovery attack combined with the new guessing strategy of C^1. The results are summarized in Table 1.

Table 1: State-recovery attacks on KETJE JR

Algorithm	Bit rate	Number of key blocks	Sieving method	Computational complexity	Search complexity	Memory complexity	Source
KETJE JR v1	r=32	n=4	Pg	2^{87}	2^{92}	$2^{65.82}$	[12]
KETJE JR v1	r=32	n=4	Pg	$\mathbf{2^{87}}$	$\mathbf{2^{86}}$	$\mathbf{2^{65.89}}$	**Section 5**
KETJE JR v1	r=40	n=4	Pg	2^{73}	2^{70}	$2^{63.87}$	[12]
KETJE JR v1	r=40	n=4	Pg	$\mathbf{2^{71}}$	$\mathbf{2^{70}}$	$\mathbf{2^{65.89}}$	**Section 4**
KETJE JR v1	r=40	n=4	Lm	$2^{71.5}$	2^{70}	$2^{63.87}$	[12]

Pg: Preliminary guessing; Lm: List merging.

The rest of the paper is organized as follows: we briefly summarize the KETJE JR algorithm in section 2. And then, we introduces the basic ideas of the state-recovery attack on KETJE JR based on the divide-and-conquer strategy in section 3. Section 4 gives an improved state-recovery attack on KETJE JR using four consecutive blocks when r=40, and Section 5 studies and gives an improved state-recovery attack on KETJE JR when r=32. Finally, some conclusions are drawn.

2. Introduction of KETJE JR

KETJE relies on the Monkey Wrap mode of operation and the round-reduced version of KECCAK permutation. This section provides a brief introduction to KETJE JR and focuses on the components involved in the following analysis.

2.1. The Monkey Wrap Mode of Operation

The Monkey Wrap mode of operation is a mode for authentication encryption with associated data based on the sponge construction. It consists of four parts: initialization, association data processing, plaintext processing and tag extraction, which enables the authentication encryption of multiple plaintexts and associated data. For more details, please refer to Ref. [4].

2.2. KECCAK-p Permutation

KECCAK-p permutations are derived from KECCAK-f permutations [13] and have a tunable number of rounds. According to its width b and its number of rounds n_r, KECCAK-p permutation is usually denoted as KECCAK-$p[b, n_r]$, and KECCAK-p for short. For KETJE JR, KECCAK-p permutation is described as a sequence of operations on a state A of 200 bits that can be regarded as a 5×5×8 three-dimensional array on $GF(2)$. $A[x, y, z]$ represents the bit at the coordinate $[x, y, z]$, corresponding to the $(40y + 8x + z)$-th bit of the internal state. For simplicity, we use the same terms as the authors did in Ref. [14].

KECCAK-p is an iterated permutation, consisting of a sequence of n_r rounds F. F consists of five steps : $F = \iota \circ \chi \circ \pi \circ \rho \circ \theta$, where

θ: $A[x,y,z] = A[x,y,z] \oplus \sum_{i=0}^{4} A[x-1,i,z] \oplus \sum_{i=0}^{4} A[x+1,i,z-1]$,

ρ: $A[x,y,z] = A[x,y,z-r[x,y]]$,

π: $A[y,2x+3y,*] = A[x,y,*]$,

χ: $A[x,y,z] = A[x,y,z] \oplus (A[x+1,y,z] \oplus 1) A[x+2,y,z]$,

ι: $A[0,0,*] = A[0,0,*] \oplus RC[i_r]$.

θ is a linear transformation. Each bit $A[x,y,z]$ is XORed with all bits in two adjacent columns $A[x-1,*,z]$ and $A[x+1,*,z-1]$. ρ provides diffusion between the slices of the state. It consists of different cyclic shifts. The cyclic shift constants can refer to Ref. [14]. π is a substitution operation on the coordinate position (x, y). χ is the only nonlinear operation of the round function F, which can be regarded as a nonlinear transformation for each row of the state A. ι is a constant addition to add a round constant to $A[0,0,*]$. The values of the round constants $RC[i_r]$ are given in Ref. [2], which will not be detailed here.

2.3. Keystream Extraction

According to the construction of Monkey Wrap, the first r bits of the internal state are extracted as the keystream after N_{step} rounds of KECCAK-p permutation, and r is called bit rate. In addition, according to the correspondence between internal state bits and coordinates $[x,y,z]$, for KETJE JR v1, the keystream bits are located on the same plane $A[*,0,*]$ when $r \leq 40$.

This paper stipulates that all counts start from 0, the left side of the data is the least significant bit, and the right side is the most significant bit. In addition, unless otherwise specified below, KETJE JR refers to KETJE JR v1.

3. State-recovery Attacks on KETJE JR based on Divide-and-conquer Strategy

KETJE JR supports a key K of variable length up to 182 bits, and the recommended key length is 96 bits. The security claimed by the authors for KETJE JR is determined by $min(96,k)$ for a key of size k. Based on the idea of meet-in-the-middle and the divide-and-conquer strategy, Ref. [12] gives state-recovery attacks on modified KETJE JR whose computation complexities are lower than exhaustive attack by using several consecutive blocks of keystream. The following is a brief introduction to the basic idea of this attack. For details, please refer to Ref. [12]. Assume that the size of each keystream block is 40 bits, i.e. $r = 40$.

3.1. Divide-and-conquer Framework

Divide-and-conquer is a common technique in cryptanalysis [15-18]. Its main idea is to decompose a problem that is difficult to solve directly into some smaller sub-problems. These sub-problems are independent of each other and have the same form as the original problem. These sub-problems can be solved in turn, then the solutions of these sub-problems are combined to obtain the solution of the original problem.

The state-recovery attack on KETJE JR based on the divide-and-conquer strategy can be formalized as a such problem: given a set $\mathcal{U}$ and a set $\mathcal{W}$, denote $\mathcal{H}: (\mathcal{U},\mathcal{W}) \longrightarrow GF(2)^d$ as a function defined on $(\mathcal{U},\mathcal{W})$, find all $u \in \mathcal{U}$ and $w \in \mathcal{W}$ such that $\mathcal{H}(u,w) = t$. For this problem, it is usually necessary to traverse all possible $u \in \mathcal{U}$ and $w \in \mathcal{W}$. Hence the corresponding computational complexity is $O(|\mathcal{U}| \cdot |\mathcal{W}|)$. If the function $\mathcal{H}(u,w)$ is in the form of $\mathcal{H}(u,w) = f(u) \oplus g(w)$, correspondingly the problem is to find the solution of $\mathcal{H}(u,w) = f(u) \oplus g(w) = t$, one can use the divide-and-conquer strategy. First, traverse all possible $u \in \mathcal{U}$ to compute $f(u)$ and store $(u, f(u))$ in a list sorted according to the value of $f(u) \oplus t$. Then, traverse all possible $w \in \mathcal{W}$ to compute $g(w)$ and search the list to find all $u \in \mathcal{U}$ such that $f(u) \oplus t = g(w)$. Such (u,w) is a solution of $\mathcal{H}(u,w) = t$.

The computational complexity of the above process based on the divide-and-conquer strategy is $|\mathcal{U}| + |\mathcal{W}|$, and the memory complexity is $|\mathcal{U}|(\log_2(|\mathcal{U}|) + d)$, where $|\mathcal{U}|$ is the number of elements in $\mathcal{U}$ and

$\log_2(|\mathcal{U}|)$ represents the bit size of $|\mathcal{U}|$. The search complexity is $|\mathcal{U}| \times |\mathcal{W}| \times 2^{-d}$, which is determined by the number of possible solutions.

3.2. Divide-and-conquer Attack on KETJE JR

The basic idea of the state-recovery attack on KETJE JR based on the divide-and-conquer strategy is as follows: The internal state of KETJE JR to be restored is divided into two parts. According to the transformations in KETJE JR, algebraic equations of the internal state satisfying the divide-and-conquer attack can be constructed from the known output blocks of keystream. By using the divide-and-conquer strategy, all possible values of two parts of the internal state are traversed separately, and two lists are constructed correspondingly. Search the lists to find the elements satisfying the above algebraic equations, and combine the corresponding two partial state values as a candidate. Then, all the candidates are tested in turn until the correct internal state is obtained. The state-recovery attack on KETJE JR using 3 known consecutive blocks of keystream is shown in Fig. 1.

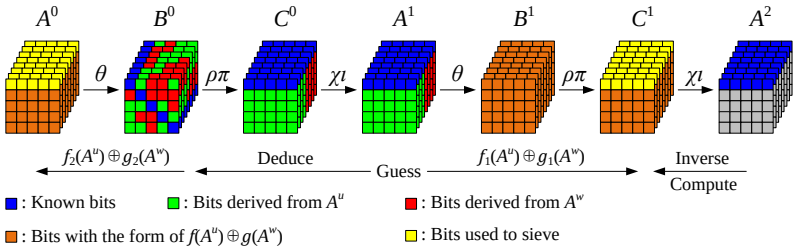

Fig. 1: Schematic diagram of the state-recovery attack on KETJE JR using 3 consecutive keystream blocks.

$A^0[*,0,*], A^1[*,0,*], A^2[*,0,*]$ are three known consecutive blocks of keystream. A^0 is the state containing the first known keystream block. B^0 is the value of the state after applying θ to A^0 , and C^0 is the state after applying ρ and π to B^0. A^1 represents the state after applying χ and ι to C^0, which contains the second keystream block. B^1, C^1, A^2 can be deduced by analogy.

The basic idea of recovering the full state A^1 based on the divide-and-conquer strategy is to guess the front half state $A^u(A^1[*,*,0], A^1[*,*,1], A^1[*,*,2], A^1[*,*,3])$ and the back half state $A^w(A^1[*,*,4], A^1[*,*,5], A^1[*,*,6], A^1[*,*,7])$ of the state A^1 respectively. It is worthy of note that if the keystream block $A^2[*,0,*]$ is known, $C^1[*,0,*]$ can be deduced. Since θ, ρ, π are all linear permutations, each bit of $C^1[*,0,*]$ can be expressed as a linear function of A^1, that is $C^1[*,0,*] = f_1(A^u) \oplus g_1(A^w)$. In addition, given A^u and A^w, the front half state C^u and the back half state C^w of the state C^0 can be determined correspondingly. According to the inverses of θ, ρ, π, each bit of $A^0[*,0,*]$ can be expressed as a linear function of C^0, that is $A^0[*,0,*] = f_2(C^u) \oplus g_2(C^w)$. We then define

$$f(A^u) = (f_1(A^u), f_2(C^u)),\ g(A^w) = (g_1(A^w), g_2(C^w)).$$

The state-recovery attack on A^1 based on the divide-and-conquer strategy is to find all A^u and A^w that satisfy

$$(C^1[*,0,*], A^0[*,0,*]) = f(A^u) \oplus g(A^w).$$

In the above state-recovery attack, since the keystream block $A^1[*,0,*]$ is known, the bits that need to be guessed in the front half A^u and the back half A^w of the state A^1 are 80. The corresponding computational complexity is 2^{80} evluations of f and g, the memory complexity is $2^{80} \times 160$, and the search complexity is $2^{80} \times 2^{80} \times 2^{-80} = 2^{80}$.

4. Improved State-recovery Attack on KETJE JR when *r*=40

When *r*=40, the number of possible internal state values is 2^{40} if there are 4 consecutive keystream blocks that are known. At this time, the state-recovery attack on KETJE JR based on the divide-and-conquer strategy is similar to section 3.2. The difference is that we need to use $A^0[*,0,*]$ and $B^2[i,i,*]$ $(i \in \{0,1,2,3,4\})$ to establish algebraic equations. Since each bit of $B^2[i,i,*]$ is a nonlinear function of A^1, in order to meet the conditions of divide-and-conquer, Ref. [12] gives a method called *preliminary guessing*. By guessing partial values of C^1, the representations of some bits in $B^2[i,i,*]$ can be derived from the function

of front half and the function of back half of A^1, and then we can combine $A^0[*,0,*]$ to sieve the guessed internal state values. We first briefly analyze the complexity of this state-recovery attack on KETJE JR (see Ref. [12] for details), and then give an improved guessing strategy combined with its complexity analysis.

4.1. Complexity Analysis of the State-recovery Attack on KETJE JR Based on Preliminary Guessing

Fig. 2 is a schematic diagram of 3-round KETJE JR with r=40. The parts are colored in yellow, blue, purple, and black representing the known state bits after different transformations. $A^0[*,0,*], A^1[*,0,*], A^2[*,0,*], A^3[*,0,*]$ are 4 known consecutive blocks of keystream. Assuming that A^1 is the internal state needed to be restored, based on the idea of divide-and-conquer, the unknown bits in the front half A^u and the back half A^w of A^1 are guessed separately. Then, the guessed state values are sieved by the known bits in A^1, A^2, A^3. In order to establish the algebraic equations of A^u and A^w using $B^2[i,i,*](i \in \{0,1,2,3,4\})$ that satisfies the conditions of divide-and-conquer, Ref. [12] guesses some bits of C^1. Assuming that m bits of C^1 are guessed, the number of algebraic equations that can be established by $B^2[i,i,*]$ is R, then the computational complexity of the state-recovery attack based on the divide-and-conquer strategy is $2^{65+\frac{m}{2}}$. The search complexity is 2^{80-R}. The memory complexity is $(115+R-m/2)\times 2^{65-\frac{m}{2}}$.

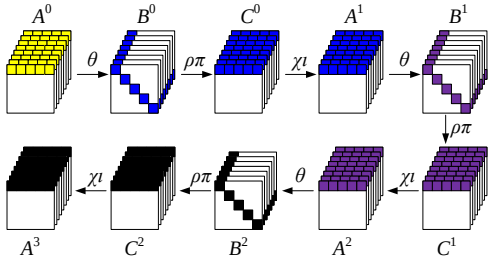

Fig. 2: Schematic diagram of 3-round KETJE JR when r=40.

In Ref. [12], by guessing $m = 16$ bits of C^1, the number of algebraic equations that can be established to satisfy the conditions of divide-and-conquer is $R = 10$. The corresponding computational complexity is 2^{73}, the search complexity is 2^{70}, and the memory complexity is $117 \times 2^{65-8} \approx 2^{63.87}$.

4.2. Improved State-recovery Attack on KETJE JR Based on Preliminary Guessing When r=40

Notice that the computational complexity and the search complexity in the state-recovery attack above are bottlenecks of the entire process. In Ref. [12], in order to express the bits of $B^2[i,i,*]$ as the sum of the front half and the back half of A^1, it is required that each state bit in two consecutive slices of A^2 is a function of A^u or A^w. Thus they need to guess all state bits involved in the corresponding slice of C^1. However, this guessing strategy does not take full advantage of the relationships between the algebraic representations of the bits in $B^2[i,i,*]$ and the algebraic equations derived from them. Based on this consideration, the following gives an improved state-recovery attack on KETJE JR using 4 keystream blocks when r=40.

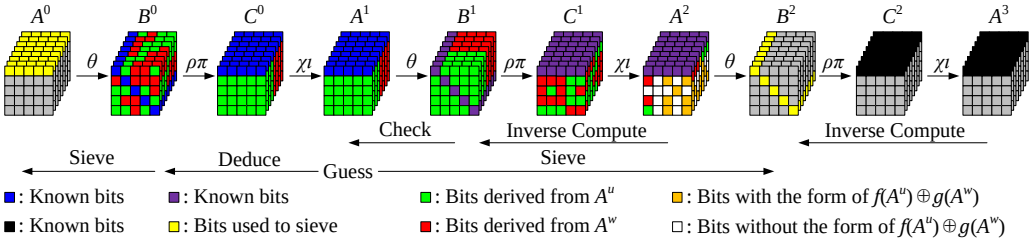

Fig. 3: Schematic diagram of the state-recovery attack on KETJE JR using 4 keystream blocks (r=40).

Suppose that A^1 is the internal state needed to be restored. Following up the definitions in Ref. [12], A^u is defined as a 105-bit state composed of the front half of A^1 and 5 parity bits

$$P^1[7] = (P^1[0,7], P^1[1,7], P^1[2,7], P^1[3,7], P^1[4,7]),$$

A^w is defined as a 105-bit state composed of the back half of A^1 and 5 parity bits

$$P^1[3] = (P^1[0,3], P^1[1,3], P^1[2,3], P^1[3,3], P^1[4,3]),$$

where $P^1[x,z] = \sum_{i=0}^{4} A[x,i,z]$. Then each bit in B^1 can be expressed as a linear function of A^u or A^w.

We need to guess the unknown bits in A^u and A^w respectively. Since $A^1[*,0,*]$ is known, the number of unknown bits in A^u and A^w are both 85. In addition, given $A^2[*,0,*]$, $B^1[i,i,*](i \in \{0,1,2,3,4\})$ can be obtained as is shown in Fig. 3. Thus, 20 linear check relations with respect to 85 unknown bits can be established, and the final number of free unknown bits in A^u and A^w is 65 by using Gaussian elimination.

From the above discussion, if A^u and A^w are known, the front half $C^u(C^0[*,*,0], C^0[*,*,1], C^0[*,*,2], C^0[*,*,3])$ and the back half $C^w(C^0[*,*,4], C^0[*,*,5], C^0[*,*,6], C^0[*,*,7])$ of C^0 can be determined correspondingly. According to the inverse transformations of θ, ρ, π, each bit of $A^0[*,0,*]$ can be expressed as a linear function of C^0. Without loss of generality, denote

$$A^0[*,0,*] = f_1(C^u) \oplus g_1(C^w). \tag{1}$$

In addition, from $P^1[x,z] = \sum_{i=0}^{4} A[x,i,z]$, we can easily deduce

$$P^1[3] = f_2(A^u),\ P^1[7] = g_3(A^w). \tag{*}$$

From the definition of A^u and A^w, $P^1[7]$ and $P^1[3]$ are contained in A^u and A^w respectively. Let $P^1[3] = g_2(A^w)$ and $P^1[7] = f_3(A^u)$, and substitute them in the formula (*). Then we have

$$0^5 = f_2(A^u) \oplus g_2(A^w), \tag{2}$$

$$0^5 = f_3(A^u) \oplus g_3(A^w). \tag{3}$$

0^5 represents the concatenation of 5 bits 0.

In summary, from the known keystream block $A^0[*,0,*]$ and the parity bits $P^1[7]$, $P^1[3]$, 50 linear equations with respect to A^u and A^w can be established to sieve the guesses of A^u and A^w. So far, for the 4 known consecutive keystream blcoks--$A^0[*,0,*], A^1[*,0,*], A^2[*,0,*], A^3[*,0,*]$, only $A^3[*,0,*]$ has not been used. Actually, given $A^3[*,0,*]$, $B^2[i,i,*]$ ($i \in \{0,1,2,3,4\}$) can be obtained by the inverse transformations. Nevertheless, due to the existence of χ, many bits of A^2 cannot be expressed in the form of $f(A^u) \oplus g(A^w)$. That is to say, there exists the unwanted term $f(A^u)g(A^w)$ in their algebraic equations. So that, after applying θ to A^2, even if $B^2[i,i,*]$ and its algebraic equation are known, $B^2[i,i,*]$ cannot be used for sieving since the conditions of the divide-and-conquer attack are not satisfied. For this reason, we give a new guessing strategy for C^1. Compared with Ref. [12], by using the relationship between the algebraic representations of bits in $B^2[i,i,*]$, it is possible to obtain enough algebraic equations that satisfy the conditions of divide-and-conquer attack while reducing the guessed bits. We expand the transition from A^1 to A^3 in Fig. 3 in terms of slice (as shown in Fig. 4).

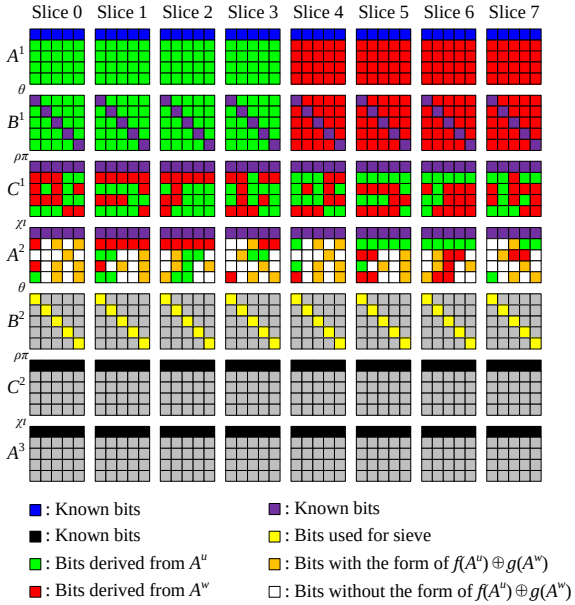

Fig. 4: Expanded diagram of state transition from A^1 to A^3.

In order to establish enough algebraic equations that satisfy the conditions of divide-and-conquer by B^2, we guess 6 bits of C^u

$$C^1[1,2,0],C^1[3,2,0],C^1[1,2,2],C^1[0,4,2],C^1[1,3,3],C^1[3,3,3],$$

and 6 bits of C^w

$$C^1[1,2,4],C^1[3,2,4],C^1[1,2,6],C^1[0,4,6],C^1[1,3,7],C^1[3,3,7],$$

respectively, totaling 12 bits (as shown in Fig. 5). Then the form of the algebraic equation of each bit in A^2 is shown in Fig. 5.

In Fig. 5, each small square colored in white of A^2 indicates that the quadratic term $f(A^u)g(A^w)$ is contained in its algebraic equation. For example, $A^2[1,1,0] = C^1[1,1,0] \oplus (C^1[2,1,0] \oplus 1)C^1[3,1,0]$.

Let $C^1[1,1,0] = f_1(A^u)$, $C^1[2,1,0] \oplus 1 = f_2(A^u)$ and $C^1[3,1,0] = g_1(A^w)$, then

$$A^2[1,1,0] = f_1(A^u) \oplus f_2(A^u)g_1(A^w).$$

For convenience, let $q_1 = A^2[1,1,0], q_2 = A^2[3,1,0], \cdots, q_{40} = A^2[3,4,7]$, and denote the known bits of each slice in B^2 as $(a_0, b_0, c_0, d_0, e_0), (a_1, b_1, c_1, d_1, e_1), \cdots, (a_7, b_7, c_7, d_7, e_7)$, then:

$$\begin{aligned}
a_0 &= q_{36} \oplus q_{39} \oplus \mathcal{L}_{a_0}, \\
b_0 &= q_1 \oplus \mathcal{L}_{b_0}, \\
c_0 &= q_1 \oplus q_3 \oplus q_5 \oplus q_{40} \oplus \mathcal{L}_{c_0}, \\
d_0 &= q_4 \oplus q_{38} \oplus \mathcal{L}_{d_0}, \\
e_0 &= q_2 \oplus q_4 \oplus q_6 \oplus q_{35} \oplus q_{37} \oplus \mathcal{L}_{e_0},
\end{aligned}$$

$$\begin{aligned}
a_1 &= q_1 \oplus q_3 \oplus q_5 \oplus \mathcal{L}_{a_1}, \\
b_1 &= \mathcal{L}_{b_1}, \\
c_1 &= q_7 \oplus q_9 \oplus q_2 \oplus q_4 \oplus q_6 \oplus \mathcal{L}_{c_1}, \\
d_1 &= q_7 \oplus q_{10} \oplus q_{11} \oplus \mathcal{L}_{d_1}, \\
e_1 &= q_8 \oplus q_{10} \oplus q_{12} \oplus \mathcal{L}_{e_1},
\end{aligned}$$

$$\begin{aligned}
a_2 &= q_9 \oplus \mathcal{L}_{a_2}, \\
b_2 &= q_{13} \oplus q_7 \oplus q_{11} \oplus \mathcal{L}_{b_2}, \\
c_2 &= q_8 \oplus q_{10} \oplus q_{12} \oplus \mathcal{L}_{c_2}, \\
d_2 &= q_{14} \oplus \mathcal{L}_{d_2}, \\
e_2 &= q_{14} \oplus \mathcal{L}_{e_2},
\end{aligned}$$

$$\begin{aligned}
a_3 &= q_{18} \oplus \mathcal{L}_{a_3}, \\
b_3 &= q_{16} \oplus q_{15} \oplus q_{17} \oplus \mathcal{L}_{b_3}, \\
c_3 &= q_{16} \oplus q_{19} \oplus q_{14} \oplus \mathcal{L}_{c_3}, \\
d_3 &= \mathcal{L}_{d_3}, \\
e_3 &= q_{20} \oplus q_{13} \oplus \mathcal{L}_{e_3}.
\end{aligned}$$

$$\begin{aligned}
a_4 &= q_{16} \oplus q_{19} \oplus \mathcal{L}_{a_4}, \\
b_4 &= q_{21} \oplus \mathcal{L}_{b_4}, \\
c_4 &= q_{21} \oplus q_{23} \oplus q_{25} \oplus q_{20} \oplus \mathcal{L}_{c_4}, \\
d_4 &= q_{24} \oplus q_{18} \oplus \mathcal{L}_{d_4}, \\
e_4 &= q_{22} \oplus q_{24} \oplus q_{26} \oplus q_{15} \oplus q_{17} \oplus \mathcal{L}_{e_4},
\end{aligned}$$

$$\begin{aligned}
a_5 &= q_{21} \oplus q_{23} \oplus q_{25} \oplus \mathcal{L}_{a_5}, \\
b_5 &= \mathcal{L}_{b_5}, \\
c_5 &= q_{27} \oplus q_{29} \oplus q_{22} \oplus q_{24} \oplus q_{26} \oplus \mathcal{L}_{c_5}, \\
d_5 &= q_{27} \oplus q_{30} \oplus q_{31} \oplus \mathcal{L}_{d_5}, \\
e_5 &= q_{28} \oplus q_{30} \oplus q_{32} \oplus \mathcal{L}_{e_5},
\end{aligned}$$

$$\begin{aligned}
a_6 &= q_{29} \oplus \mathcal{L}_{a_6}, \\
b_6 &= q_{33} \oplus q_{27} \oplus q_{31} \oplus \mathcal{L}_{b_6}, \\
c_6 &= q_{28} \oplus q_{30} \oplus q_{32} \oplus \mathcal{L}_{c_6}, \\
d_6 &= q_{34} \oplus \mathcal{L}_{d_6}, \\
e_6 &= q_{34} \oplus \mathcal{L}_{e_6},
\end{aligned}$$

$$\begin{aligned}
a_7 &= q_{38} \oplus \mathcal{L}_{a_7}, \\
b_7 &= q_{36} \oplus q_{35} \oplus q_{37} \oplus \mathcal{L}_{b_7}, \\
c_7 &= q_{36} \oplus q_{39} \oplus q_{34} \oplus \mathcal{L}_{c_7}, \\
d_7 &= \mathcal{L}_{d_7}, \\
e_7 &= q_{40} \oplus q_{33} \oplus \mathcal{L}_{e_7}.
\end{aligned}$$

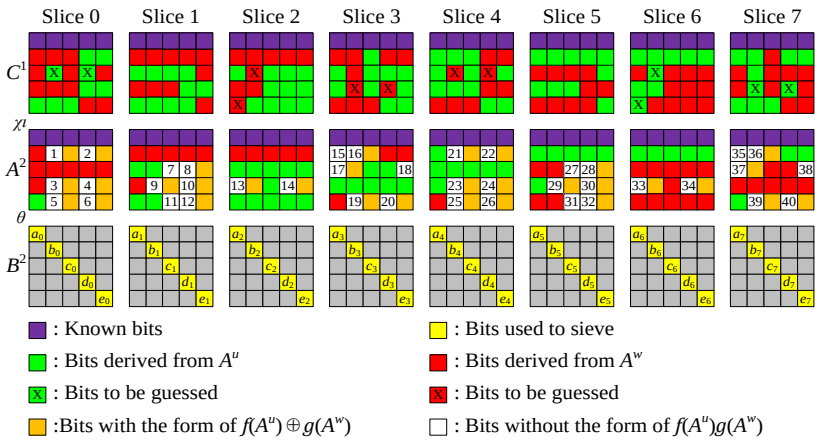

Fig. 5: Expanded diagram of state transition of C^1 based on preliminary guessing.

In above algebraic equations, $\mathcal{L}_\Lambda$ denotes $\mathcal{L}_\Lambda = f_\Lambda(A^u) \oplus g_\Lambda(A^w)$, $\Lambda \in \{a_0, b_0, c_0, d_0, e_0, \cdots, a_7, b_7, c_7, d_7, e_7\}$. Through Gaussian elimination, it can be known that the rank of the above system with 40 variables $q_1, q_2, \cdots, q_{40}$ is 30, and

$$b_1 = \mathcal{L}_{b_1}, \quad b_5 = \mathcal{L}_{b_5},$$
$$d_3 = \mathcal{L}_{d_3}, \quad d_7 = \mathcal{L}_{d_7},$$
$$e_1 \oplus c_2 = \mathcal{L}_{e_1} \oplus \mathcal{L}_{c_2}, \quad e_5 \oplus c_6 = \mathcal{L}_{e_5} \oplus \mathcal{L}_{c_6},$$
$$d_2 \oplus e_2 = \mathcal{L}_{d_2} \oplus \mathcal{L}_{e_2}, \quad d_6 \oplus e_6 = \mathcal{L}_{d_6} \oplus \mathcal{L}_{e_6},$$
$$a_0 \oplus e_6 \oplus c_7 = \mathcal{L}_{a_0} \oplus \mathcal{L}_{e_6} \oplus \mathcal{L}_{c_7}, \quad a_4 \oplus c_3 \oplus e_2 = \mathcal{L}_{a_4} \oplus \mathcal{L}_{c_3} \oplus \mathcal{L}_{e_2}.$$

Let $\Gamma = (b_1, b_5, d_3, d_7, e_1 \oplus c_2, e_5 \oplus c_6, d_2 \oplus e_2, d_6 \oplus e_6, a_0 \oplus e_6 \oplus c_7, a_4 \oplus c_3 \oplus e_2)$, then Γ can be expressed as the following form:

$$\Gamma = f_4(A^u) \oplus g_4(A^w), \tag{4}$$

Create the simultaneous equations from formula (1) ~ formula (4) and define:

$$f(A^u) = (f_1(A^u), f_2(A^u), f_3(A^u), f_4(A^u)),$$
$$g(A^w) = (g_1(A^w), g_2(A^w), g_3(A^w), g_4(A^w)).$$

Then, we have $(A^0[*,0,*], 0^5, 0^5, \Gamma) = f(A^u) \oplus g(A^w)$.

In summary, when r=40, the number of algebraic equations that can be established in the improved state-recovery attack on KETJE JR based on the preliminary guessing is 10 by guessing 12 bits of C^1. The corresponding computational complexity is $2^{65+\frac{12}{2}} = 2^{71}$, the search complexity is $2^{80-10} = 2^{70}$, and the memory complexity is $119 \times 2^{65-\frac{12}{2}} \approx 2^{65.89}$.

5. Improved State-recovery Attack on KETJE JR when r=32

From the above discussion, the number of possible internal state values is 2^{72} when 4 consecutive 32-bit keysteam blocks are known. In this section, we propose an improved state-recovery attack on KETJE JR when $r = 32$ (as shown in Fig. 6).

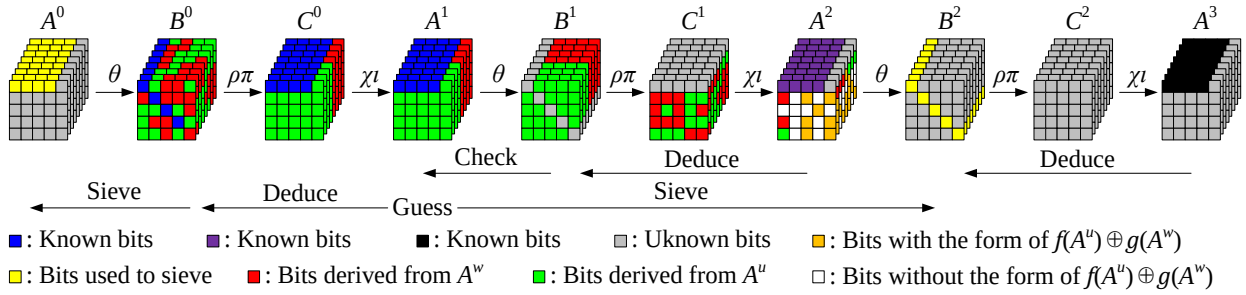

Fig. 6: Schematic diagram of the state-recovery attack on KETJE JR using 4 keystream blocks (r=32).

Through analysis, we find that when $r = 32$, there are 4 bits known in each row of $A^2[*,0,*]$, and only 1 bit is unknown. According to Ref. [19], each bit in the row of $C^1[*,0,*]$ can be expressed as a function of unknown bit in the corresponding row of $A^2[*,0,*]$. In addition, since ρ and π are transpositions of state bits, each bit of $B^1[i,i,*](i \in \{0,1,2,3,4\})$ is a function with respect to an unknown bit of $A^2[*,0,*]$. Suppose that A^1 is the internal state needed to be restored. Similar to section 4, A^u is defined as a 105-bit state composed of the front half of A^1 and the parity bit $P^1[7]$, and A^w is defined as a 105-bit state composed of the back half of A^1 and the parity bit $P^1[3]$. Then each bit in B^1 can be expressed as a linear function of A^u or A^w. In this way, linear equations of unknown bits in $A^2[*,0,*]$ and A^u (or A^w) can be established by using $B^1[i,i,*](i \in \{0,1,2,3,4\})$. On the one hand, we can derive the linear check relations between A^u and A^w which would reduce the computational complexity; on the other hand, the algebraic expressions of unknown bits in $A^2[*,0,*]$ with respect to A^u and A^w can be obtained.

Since 32 bits of $A^1[*,0,*]$ are known, the number of unknown bits needed to be guessed in A^u and A^w are both 89. In addition, 40 linear equations with respect to the unknown bits of $A^2[*,0,*]$ and A^u (or A^w) can be established by $B^1[i,i,*](i \in \{0,1,2,3,4\})$. Assuming that 20 linear equations established by the front 20 bits $(B^1[i,i,*], i \in \{0,1,2,3,4\}, * \in \{0,1,2,3\})$ of $B^1[i,i,*]$ contain 4 different unknown bits of $A^2[*,0,*]$.

Through Gaussian elimination, 16 linear check relations with respect to 89 unknown bits of A^u can be obtained, and each unknown bit can be expressed as a function of A^u. Similarly, assuming that 20 linear equations established by the back 20 bits ($B^1[i,i,*], i \in \{0,1,2,3,4\}, * \in \{4,5,6,7\}$) of $B^1[i,i,*]$ contain the other 4 different unknown bits of $A^2[*,0,*]$. Correspondingly, 16 linear check relations with respect to 89 unknown bits of A^w can be obtained, and each unknown bit can be expressed as a function of A^w. Finally, the number of free unknown bits of A^u and A^w are both 73.

From the above discussion, given A^u and A^w, the front half C^u and the back half C^w of C^0 can be determined respectively. According to the inverse transformations of θ, ρ, π, 32 known bits -- $A^{0,32}(A^{0,32} = A^0[0,0,*], A^0[1,0,*], A^0[2,0,*], A^0[3,0,*])$ of $A^0[*,0,*]$ can all be expressed as a linear function of C^0. Let

$$A^{0,32} = f_1(C^u) \oplus g_1(C^w),$$

similar to section 4, it holds that

$$0^5 = f_2(A^u) \oplus g_2(A^w),\ 0^5 = f_3(A^u) \oplus g_3(A^w).$$

To sum up, from the known keystream block $A^0[*,0,*]$ and the parity bits $P^1[7]$, $P^1[3]$, 42 linear equations with respect to A^u and A^w can be established to sieve the guesses of A^u and A^w. In addition, when $r = 32$, for each row of $A^3[*,0,*]$, there is only 1 bit that is unknown. So each bit of $B^2[i,i,*]$ can be expressed as a function of an unknown bit of $A^3[*,0,*]$. Meanwhile, each bit of $B^2[i,i,*]$ can be expressed as a function of A^u and A^w. Thus, by using $B^2[i,i,*]$, we can establish the algebraic expressions with respect to the unkown bits of $A^3[*,0,*]$ and A^u, A^w. If these algebraic expressions satisfy the conditions of divide-and-conquer, the guesses of A^u and A^w can be sieved based on these algebraic expressions.

In order to obtain enough algebraic expressions satisfying the conditions of divide-and-conquer from $B^2[i,i,*]$, 14 bits of C^u

$$C^1[4,2,1], C^1[3,3,1], C^1[4,3,1], C^1[4,4,1], C^1[1,2,2], C^1[0,3,2], C^1[1,3,2],$$
$$C^1[0,4,2], C^1[2,1,3], C^1[1,2,3], C^1[1,3,3], C^1[3,3,3], C^1[3,4,3], C^1[4,4,3],$$

and 14 bits of C^w

$$C^1[4,2,5], C^1[3,3,5], C^1[4,3,5], C^1[4,4,5], C^1[1,2,6], C^1[0,3,6], C^1[1,3,6],$$
$$C^1[0,4,6], C^1[2,1,7], C^1[1,2,7], C^1[1,3,7], C^1[3,3,7], C^1[3,4,7], C^1[4,4,7],$$

28 bits in all, are guessed respectively (as shown in Fig. 7). The form of the algebraic equation of each bit in A^2 is shown in Fig. 7.

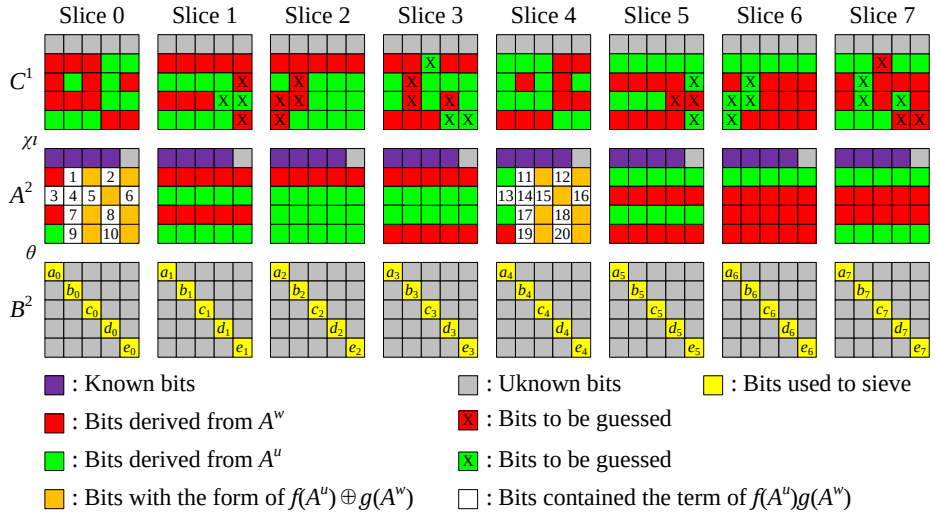

Fig. 7: Expanded diagram of state transition of C^1 based on preliminary guessing ($r = 32$)

Similar to section 4, let $q_1 = A^2[1,1,0], q_2 = A^2[3,1,0], \cdots, q_{20} = A^2[3,4,4]$, and denote bits in each slice of B^2 used for sieving as $(a_0, b_0, c_0, d_0, e_0), (a_1, b_1, c_1, d_1, e_1), \cdots, (a_7, b_7, c_7, d_7, e_7)$ (as shown in Fig. 7), then

$$
\begin{aligned}
a_0 &= q_6 \oplus \mathcal{L}_{a_0}, & a_4 &= q_{16} \oplus \mathcal{L}_{a_4},\\
b_0 &= q_1 \oplus q_3 \oplus \mathcal{L}_{b_0}, & b_4 &= q_{11} \oplus q_{13} \oplus \mathcal{L}_{b_4},\\
c_0 &= q_5 \oplus q_1 \oplus q_4 \oplus q_7 \oplus q_9 \oplus \mathcal{L}_{c_0}, & c_4 &= q_{15} \oplus q_{11} \oplus q_{14} \oplus q_{17} \oplus q_{19} \oplus \mathcal{L}_{c_4},\\
d_0 &= q_8 \oplus q_5 \oplus \mathcal{L}_{d_0}, & d_4 &= q_{18} \oplus q_{15} \oplus \mathcal{L}_{d_4},\\
e_0 &= q_2 \oplus q_8 \oplus q_{10} \oplus \mathcal{L}_{e_0}, & e_4 &= q_{12} \oplus q_{18} \oplus q_{20} \oplus \mathcal{L}_{e_4},\\
a_1 &= q_1 \oplus q_4 \oplus q_7 \oplus q_9 \oplus \mathcal{L}_{a_1}, & a_5 &= q_{11} \oplus q_{14} \oplus q_{17} \oplus q_{19} \oplus \mathcal{L}_{a_5},\\
b_1 &= q_5 \oplus \mathcal{L}_{b_1}, & b_5 &= q_{15} \oplus \mathcal{L}_{b_5},\\
c_1 &= q_2 \oplus q_8 \oplus q_{10} \oplus \mathcal{L}_{c_1}, & c_5 &= q_{12} \oplus q_{18} \oplus q_{20} \oplus \mathcal{L}_{c_5},\\
d_1 &= q_6 \oplus \mathcal{L}_{d_1}, & d_5 &= q_{16} \oplus \mathcal{L}_{d_5},\\
e_1 &= q_3 \oplus \mathcal{L}_{e_1}, & e_5 &= q_{13} \oplus \mathcal{L}_{e_5},\\
a_2 &= \mathcal{L}_{a_2}, & a_6 &= \mathcal{L}_{a_6},\\
b_2 &= \mathcal{L}_{b_2}, & b_6 &= \mathcal{L}_{b_6},\\
c_2 &= \mathcal{L}_{c_2}, & c_6 &= \mathcal{L}_{c_6},\\
d_2 &= \mathcal{L}_{d_2}, & d_6 &= \mathcal{L}_{d_6},\\
e_2 &= \mathcal{L}_{e_2}, & e_6 &= \mathcal{L}_{e_6},\\
a_3 &= \mathcal{L}_{a_3}, & a_7 &= \mathcal{L}_{a_7},\\
b_3 &= \mathcal{L}_{b_3}, & b_7 &= \mathcal{L}_{b_7},\\
c_3 &= \mathcal{L}_{c_3}, & c_7 &= \mathcal{L}_{c_7},\\
d_3 &= \mathcal{L}_{d_3}, & d_7 &= \mathcal{L}_{d_7},\\
e_3 &= \mathcal{L}_{e_3}, & e_7 &= \mathcal{L}_{e_7},
\end{aligned}
$$

Through Gaussian elimination, it can be known that the rank of the above system with 20 variables $q_1, q_2, \cdots, q_{20}$ is 14, and the number of algebraic equations that can be expressed in the form $f(A^u) \oplus g(A^w)$ is 26. Since each bit of $B^2[i,i,*]$ is a function of an unknown bit of $A^3[*,0,*]$, and there are 8 unknown bits in $A^3[*,0,*]$, at least 18 algebraic equations of A^u and A^w satisfying the conditions of divide-and-conquer can be obtained through Gaussian elimination.

In summary, when $r = 32$, in the improved state-recovery attack on KETJE JR using 4 consecutive keystream blocks, the number of algebraic equations that can be established is 18 by guessing 28 bits of C^1. The computational complexity is $2^{73+\frac{28}{2}} = 2^{87}$, the search complexity is $2^{104-18} = 2^{86}$, and the memory complexity is $119 \times 28 \approx 2^{65.89}$.

6. Conclusion

This paper studies the state-recovery attacks on the weakened version of KETJE JR v1 with the knowledge of several keystream blocks. Through analyzing the algebraic expressions of internal state bits in KETJE JR, new guessing strategies are proposed. By using new guessing strategies, we improve state-recovery attacks on KETJE JR v1 when r=40 and r=32 that reduce the complexities of known attacks. Due to the influence of π, the guessing strategies given in this paper can not be applied to analyze KETJE JR v2 directly. The state-recovery attack on KETJE JR v2 will be an important direction in our follow-up research.

7. References

[1] D. A. McGrew, J. Viega. The security and performance of the Galois/Counter Mode (GCM) of operation. In: A. Canteaut, et al(eds.). Progress in Cryptology - INDOCRYPT 2004, pp. 343–355.

[2] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, R. V. Keer. KETJE v1. Submission to Caesar competition, 2014. https://competitions.cr.yp.to/round1/ketjev1.pdf.

[3] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche. On the indifferentiability of the sponge construction. In EUROCRYPT 2008, volume 4965 of Lecture Notes in Computer Science. pp. 181–197.

[4] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, R. V. Keer. KETJE v2. Submission to Caesar competition, 2016. http://competitions.cr.yp.to/round3/ketjev2.pdf.

[5] X. Y. Dong, Z. Li, X. Y. Wang, L. Qin. Cube-like attack on round-reduced initialization of KETJE SR. IACR Trans.

Symmetric Cryptol. 2017(1): 259–280.

[6] Z. Li, W. Q. Bi, X. Y. Dong, X. Y. Wang. Improved Conditional Cube Attacks on Keccak Keyed Modes with MILP Method. In Advances in Cryptology - ASIACRYPT 2017, pp. 99-127.

[7] L. Song, J. Guo, D. P. Shi, S. Ling. New MILP Modeling: Improved Conditional Cube Attacks on Keccak-Based Constructions. In Advances in Cryptology - ASIACRYPT 2018, pp. 65-95.

[8] W. Q. Bi, X. Y. Dong, Z. Li, R. Zong, X. Y. Wang. MILP-aided Cube-Attack-Like Cryptanalysis on Keccak Keyed Modes. Designs, Codes and Cryptography. 2019(87): 1271–1296.

[9] L. Song, J. Guo. Cube-Attack-Like Cryptanalysis of Round-Reduced Keccak Using MILP. IACR Trans. Symmetric Cryptol. 2018(3):182-214.

[10] Z. Li, X. Y. Dong, W. Q. Bi, K. T. Jia, X. Y. Wang, W. Meier. New conditional cube attack on keccak keyed modes. IACR Trans. Symmetric Cryptol. 2019(2): 94-124.

[11] H. B. Zhou, Z. Li, X. Y. Dong, K. T. Jia, W. Meier. Practical Key-recovery Attacks on Round-Redude Ketje Jr, Xoodoo-AE and Xoodyak. Cryptology ePrint Archive, 2019/392.

[12] T. Fuhr, M. N. Plasencia, Y. Rotella. State-recovery attacks on modified KETJE JR. IACR Trans. Symmetric Cryptol. 2018(1): 29-56.

[13] The KECCAK reference. 2011, http://keccak.noekeon.org/.

[14] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche. The Keccak sponge function family. http://keccak.noekeon.org/.

[15] O. Dunkelman, G. Sekar, B. Preneel. Improved meet-in-the-middle attacks on reduced-round DES. In K. Srinathan, et al(eds.). Progress in Cryptology - INDOCRYPT 2007, pp. 86–100.

[16] Y. Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to whirlpool. In: A. Joux(eds.). Fast Software Encryption 2011, pp. 378–396.

[17] M. N. Plasencia. How to improve rebound attacks. In: P. Rogaway, et al(eds.). Advances in Cryptology - CRYPTO 2011, pp. 188–205.

[18] A. Canteaut, M. N. Plasencia, B. Vayssi ère. Sieve-in-the middle: Improved MITM attacks. In: R. Canetti, et al(eds.). Advances in Cryptology - CRYPTO 2013, pp. 222–240.

[19] J. Guo, M. C. Liu, L. Song. Linear structures: applications to cryptanalysis of round-reduced keccak. In: International Conference on the Theory and Application of Cryptology and Information Security. 2016, pp. 249-274.