Improved Conditional Cube Attacks on Keccak Keyed Modes with MILP Method*

Zheng Li[1], Wenquan Bi[1], Xiaoyang Dong[2**], and Xiaoyun Wang[1,2**]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, P. R. China,
{lizhengcn, biwenquan}@mail.sdu.edu.cn
[2] Institute for Advanced Study, Tsinghua University, P. R. China,
{xiaoyangdong, xiaoyunwang}@tsinghua.edu.cn.

Abstract. Conditional cube attack is an efficient key-recovery attack on Keccak keyed modes proposed by Huang *et al.* at EUROCRYPT 2017. By assigning bit conditions, the diffusion of a conditional cube variable is reduced. Then, using a greedy algorithm (Algorithm 4 in Huang *et al.*'s paper), Huang *et al.* find some ordinary cube variables, that do not multiply together in the $1st$ round and do not multiply with the conditional cube variable in the $2nd$ round. Then the key-recovery attack is launched. The key part of conditional cube attack is to find enough ordinary cube variables. Note that, the greedy algorithm given by Huang *et al.* adds ordinary cube variable without considering its bad effect, i.e. the new ordinary cube variable may result in that many other variables could not be selected as ordinary cube variable (they multiply with the new ordinary cube variable in the first round).

In this paper, we bring out a new MILP model to solve the above problem. We show how to model the CP-like-kernel and model the way that the ordinary cube variables do not multiply together in the $1st$ round as well as do not multiply with the conditional cube variable in the $2nd$ round. Based on these modeling strategies, a series of linear inequalities are given to restrict the way to add an ordinary cube variable. Then, by choosing the objective function of the maximal number of ordinary cube variables, we convert Huang *et al.*'s greedy algorithm into an MILP problem and the maximal ordinary cube variables are found.

Using this new MILP tool, we improve Huang *et al.*'s key-recovery attacks on reduced-round Keccak-MAC-384 and Keccak-MAC-512 by 1 round, get the first 7-round and 6-round key-recovery attacks, respectively. For Ketje Major, we conclude that when the nonce is no less than 11 lanes, a 7-round key-recovery attack could be achieved. In addition, for Ketje

** Both are corresponding authors.

Minor, we use conditional cube variable with 6-6-6 pattern to launch 7-round key-recovery attack.

1 Introduction

Nowadays, the cryptanalysis progress of symmetric-key ciphers heavily depends on automated evaluation tools. Providing a reliable security evaluation is the key point for a cipher to be accepted by industry. Recently, cryptographic communities found that many classical cryptanalysis methods could be converted to mathematical optimization problems which aim to achieve the minimal or maximal value of an objective function under certain constraints. Mixed-integer Linear Programming (MILP) is the most widely studied technique to solve these optimization problems. One of the most successful applications of MILP is to search differential and linear trails. Mouha *et al.* [26] and Wu *et al.* [31] first applied MILP method to count active Sboxes of word-based block ciphers. Then, at Asiacrypt 2014, by deriving some linear inequalities through the H-Representation of the convex hull of all differential patterns of Sbox, Sun *et al.* [30] extended this technique to search differential and linear trails. Another two important applications are to search integral distinguisher [32] and impossible differentials [28,8].

Keccak [3], designed by Bertoni *et al.*, has been selected as the new cryptographic hash function standard SHA-3. As one of the most important cryptographic standards, Keccak attracts lots of attention from the world wide researchers and engineers. Till now, many cryptanalysis results [7,10,11,18,19,21,24] and evaluation tools [9,14,23] have been proposed, including the recent impressive collision attacks [27,29]. Since the robust design of Keccak, the cryptanalysis progress of Keccak is still limited. It must be pointed out that the automatic evaluation tools for Keccak are still needed to be enriched urgently.

At Eurocrypt 2015, Dinur *et al.* [12] for the first time considered the security of the Keccak keyed modes aganist cube-attack-like cryptanalysis and give some key recovery attacks on reduced-round Keccak-MAC and Keyak [5]. At CT-RSA 2015, Dobraunig *et al.* [15] evaluate the security of Ascon [16] against the cube-like cryptanalysis. Later, Dong *et al.* [17] applied the cube-like method to Ketje Sr [4] which adopts smaller state size of Keccak-p permuation. At Eurocrypt 2017, Huang *et al.* [20] introduced a new type of cube-like attack, called *conditional cube attack*, which takes advantage of the large state freedom of Keccak to find a so-called *conditional cube variable* that do not multiply with all the other *cube variables* (called *ordinary cube variables*) in the first round and second round of Keccak, meanwhile, all *ordinary cube variables* do not multiply with each other in the first round. Thus, the degree of output polynomial of reduced-round Keccak over the cube variables is reduced by 1 and a *conditional cube*

tester is constructed. Then Li *et al.*[22] applied the *conditional cube attack* to reduced-round Ascon.

1.1 Our Contributions

For *conditional cube attack*, when the *conditional cube variable* is determined, the most important work is to find enough *ordinary cube variables* to launch the key recovery attack. In [20], Huang *et al.* gives the Algorithm 4 to search the *ordinary cube variables*. It is a greedy algorithm, it randomly selects a cube variable and adds to *ordinary cube variable* set, when the variable does not multiply with other *ordinary cube variables* in the set in the first round and does not multiply with *conditional cube variable* either in both the first and second round. The drawback is that it can hardly get the maximum number (optimal) of *ordinary cube variables*. Because, when a cube variable is added to *ordinary cube variable* set, many more variables which multiply with the new added cube variable in the first round will be discarded, which means that we add just one cube variable with the price that many variables lost the chance to be an *ordinary cube variable*. Actually, the search problem is an optimization problem. When the capacity of Keccak is large, the greedy algorithm is enough to find a proper *ordinary cube variable* set. However, when the capacity or the state size is small, the algorithm could hardly find enough *ordinary cube variables* and invalidate the *conditional cube attack*. In fact, for Keccak-MAC-512 and Keccak-MAC-384, only 5 round and 6 round attacks are achieved by Huang *et al.*'s algorithm. When the capacity is large or the internal state of Keccak sponge function is smaller than 1600-bit, e.g. 800-bit Ketje Minor, the number of *ordinary cube variables* is reduced significantly.

In this paper, we present a novel technique to search *ordinary cube variables* by using MILP method[3]. By modeling the relations between *ordinary cube variables* and *conditional cube variables* in the first and second round, modeling the so-called *CP-like-kernel* and *ordinary cube variables* chosen conditions, we construct a linear inequality system. The target object is the maximum number of *ordinary cube variables*. Based on this MILP tool, we improve Huang *et al.*'s attacks on Keccak-MAC and give some interesting results on Ketje Major and Minor, which are summarized in Table 1. In addition, we list our source code of the new MILP tool[4] in a public domain to enrich the automatic evaluation tools on Keccak and help the academic communities study Keccak much easier. The following are the main application results of the MILP tool.

1. It should be noted that, when the capacity reaches 768 or 1024, the cryptanalysis of Keccak becomes very hard. In fact, collision results on round-

[3] Note that, in Huang *et al.*'s paper, a small MILP model is also introduced, however, it could only find some distinguisher attacks on Keccak hash function.

[4] `https://github.com/lizhengcn/MILP_conditional_cube_attack`

reduced Keccak-384 or Keccak-512 that are better than the birthday bound could respectively reach 4/3-round, while the preimage attacks [25,19] on the two versions could reach only 4 rounds. Based on our MILP tool, for Keccak-MAC-384, we find more than 63 *ordinary cube variables* and improve Huang *et al.*'s attack by 1 round, and get the very first 7-round key-recovery attack. For Keccak-MAC-512, we find more than 31 *ordinary cube variables* and improve Huang *et al.*'s attack by 1 round, and get the first 6-round key-recovery attack. These are the longest attacks that the cryptanalysis of Keccak with big capacity (768 or 1024) could reach.

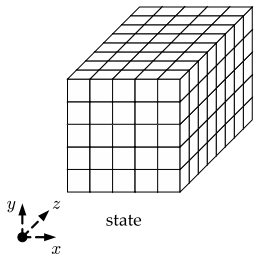
2. For Ketje Major, we conclude that when the nonce is no less than 11 lanes, a 7-round conditional cube attack could work. In addition, we get the borderline length of the nonce for the 6-round key-recovery attack is 8 lanes.
3. For Ketje Minor, we use a new *conditional cube variable* and find 124 *ordinary cube variables*. Then a new 7-round key-recovery attack is proposed, which improved the previous best result by a factor of 2^{15}.

Table 1. Summary of Key Recovery Attacks on Keccak Keyed Modes

Variant	Capacity	Attacked Rounds	Time	Source
Keccak-MAC	768	6	2^{40}	[20]
		7	2^{75}	Section 5.1
	1024	5	2^{24}	[20]
		6	$2^{58.3}$	Section 5.2
Variant	**Nonce**	**Attacked Rounds**	**Time**	**Source**
Ketje Major	–	6	2^{64}	[17]
	–	7	2^{96}	[17]
	≥ 512	6	2^{41}	Section 6.1
	≥ 704	7	2^{83}	Section 6.1
Ketje Minor	–	6	2^{64}	[17]
	–	7	2^{96}	[17]
	–	6	2^{49}	Section 6.2
	–	7	2^{81}	Section 6.2

1.2 Organization of the Paper

Section 2 gives some notations, and brief description on Keccak-permutations, Keccak-MAC, Ketje. Some related works are introduced in Section 3. Section 4 describes the MILP search model for conditional cube attack. Round-reduced key-recovery attacks on Keccak-MAC-384/512 are introduced in Section 5. Section 6 gives the applications to Ketje. Section 7 concludes this paper.



0, 0	1, 0	2, 0	3, 0	4, 0
0, 1	1, 1	2, 1	3, 1	4, 1
0, 2	1, 2	2, 2	3, 2	4, 2
0, 3	1, 3	2, 3	3, 3	4, 3
0, 4	1, 4	2, 4	3, 4	4, 4

Fig. 1. (a) The Keccak State [3], (b) State A In 2-dimension

2 Preliminaries

2.1 Notations

S_i	the intermediate state after i-round of Keccak-p, for example $S_{0.5}$ means the intermediate state before χ in 1st round of Keccak-p,
A	used in tables: for Keccak-MAC, the initial state, for Ketje, the state after π^{-1} of Keccak-p^*
$A[i][j]$	the 32/64-bit word indexed by $[i, j, *]$ of state A, $0 \leqslant i \leqslant 4$, $0 \leqslant j \leqslant 4$
$A[i][j][k]$	the bit indexed by $[i, j, k]$ of state A
v_i	the ith cube variable
K	128-bit key, for Keccak-MAC, $K = k_0 \|\| k_1$, both k_0 and k_1 are 64-bit, for Ketje Major, $K = k_0 \|\| k_1 \|\| k_2$, k_0 is 56-bit, k_1 is 64-bit, k_2 is 8-bit, for Ketje Minor, $K = k_0 \|\| k_1 \|\| k_2 \|\| k_3 \|\| k_4$, k_0 is 24-bit, k_1,k_2 and k_3 are 32-bit, k_4 is 8-bit
$k_i[j]$	the jth bit of k_i
capacity	in Keccak-MAC, it is the all zero padding bits; in Ketje, it is the padding of nonce

2.2 The Keccak-p permutations

The Keccak-p permutations are derived from the Keccak-f permutations [3] and have a tunable number of rounds. A Keccak-p permutation is defined by its width $b = 25 \times 2^l$, with $b \in \{25, 50, 100, 200, 400, 800, 1600\}$, and its number of rounds n_r, denoted as Keccak-$p[b]$. The round function $\mathtt{R}$ consists of five operations:

$$\mathtt{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

Keccak-$p[b]$ works on a state A of size b, which can be represented as 5×5 $\frac{b}{25}$-bit lanes, as depicted in Figure 1, $A[i][j]$ with i for the index of column and j for the index of row. In what follows, indexes of i and j are in set $\{0, 1, 2, 3, 4\}$ and they are working in modulo 5 without other specification.

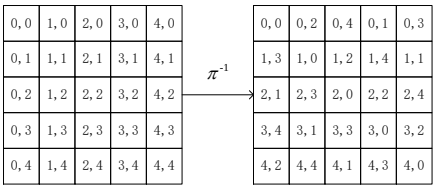


Fig. 2. π^{-1}

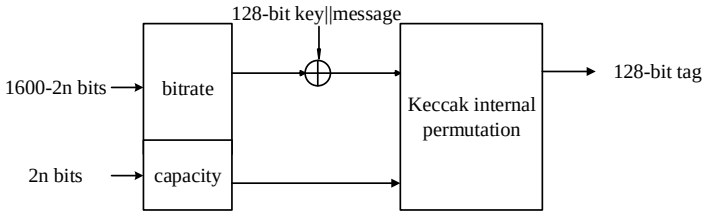


Fig. 3. Construction of Keccak-MAC-n

$\theta : A[x][y] = A[x][y] \oplus \sum_{j=0}^{4} (A[x-1][j] \oplus (A[x+1][j] \lll 1))$.
$\rho : A[x][y] = A[x][y] \lll r[x,y]$.
$\pi : A[y][2x+3y] = A[x][y]$.
$\chi : A[x][y] = A[x][y] \oplus ((\neg A[x+1][y]) \wedge A[x+2][y]$.
$\iota : A[0][0] = A[0][0] \oplus RC$.

In Ketje v2, *the twisted permutations*, Keccak-$p^*[b]$=$\pi \circ$ Keccak-$p[b] \circ \pi^{-1}$, are introduced to effectively re-order the bits in the state. π^{-1} is the inverse of π, shown in Figure 2.

$$\pi^{-1} : A[x+3y][x] = A[x][y].$$

2.3 Keccak-MAC

A MAC form of Keccak can be obtained by adding key as the prefix of message/nonce. As depicted in Figure 3, the input of Keccak-MAC-n is concatenation of key and message and n is half of the capacity length.

2.4 Ketje

Ketje [4] is a submission by Keccak team. It is a sponge-like construction. In Ketje v1, two instances are proposed, Ketje Sr and Jr with 400-bit and 200-bit state sizes, respectively. In the latest Ketje v2, another two instances Ketje

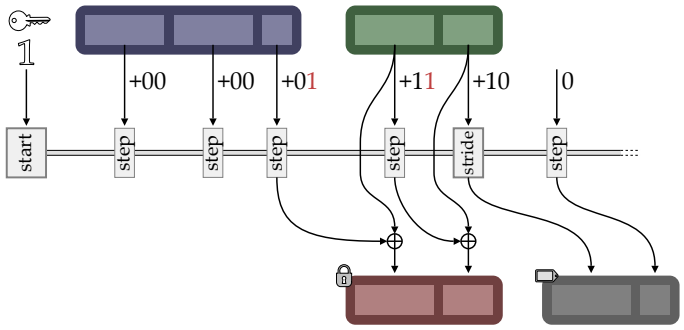


Fig. 4. Wrapping a Header and a Body with MonkeyWrap [4]

Minor and Major are added to the family, with 800-bit and 1600-bit state sizes, respectively. Ketje Sr is the primary recommendation. In the following, we give a brief overview about the Ketje v2. For a complete description, we refer to the design document [4].

The structure of Ketje is an authenticated encryption mode MonkeyWrap, shown Figure 4, which is based on MonkeyDuplex [6]. It consists of four parts as follows:

1. **The initialization phase**: The initialization takes the secret key K, the public nonce N and some paddings as the initial state. Then $n_{start} = 12$ rounds Keccak-p^* is applied.
2. **Processing associated data**: ρ-bit blocks associated data are padded to $(\rho + 4)$-bit and absorbed by xoring them to the state, then $n_{step} = 1$ round Keccak-p^* is applied. If associated data is empty, this procedure is still needed to be applied which means an empty block is padded to $(\rho + 4)$-bit and then processed similarly.
3. **Processing the plaintext**: Plaintext is processed in ρ-bit blocks in a similar manner, with ciphertext blocks extracted from the state right after adding the plaintext.
4. **Finalization**: The finalization with $n_{stride} = 6$ rounds Keccak-p^* and a series of $n_{step} = 1$ round Keccak-p^*s are performed to get the required length of tag T.

In Ketje v2, four concrete instances are proposed, shown in Table 2. $n_{start} = 12$,$n_{step} = 1$ and $n_{stride} = 6$. For Ketje Minor and Major, the recommended key length is 128-bit, so the maximal length of nonce is (800-128-18=)654 and (1600-128-18=)1454 bits. This paper discusses the shortest length of nonce that a conditional cube attack could be applied.

Table 2. Four Instances in Ketje v2

Name	f	ρ	Main use case
Ketje Jr	Keccak-p^*[200]	16	lightweight
Ketje Sr	Keccak-p^*[400]	32	lightweight
Ketje Minor	Keccak-p^*[800]	128	lightweight
Ketje Major	Keccak-p^*[1600]	256	high performance

3 Related Work

3.1 Cube Attack

At EUROCRYPT 2009, Dinur and Shamir introduced the cube attack [13], in which the output bit of a symmetric cryptographic scheme can be regarded as a polynomial $f(k_0, ..., k_{n-1}, v_0, ..., v_{m-1})$ over $GF(2)$, $k_0, ..., k_{n-1}$ are the secret variables (the key bits), $v_0, ..., v_{m-1}$ are the public variables (e.g. IV or nonce bits).

Theorem 1. *([13])*

$$f(k_0, ..., k_{n-1}, v_0, ..., v_{m-1}) = t \cdot P + Q(k_0, ..., k_{n-1}, v_0, ..., v_{m-1}) \tag{1}$$

t is called maxterm and is a product of certain public variables, for example $(v_0, ..., v_{s-1})$, $1 \leq s \leq m$, denoted as cube C_t. None of the monomials in Q is divisible by t. P is called superpoly, which does not contain any variables of C_t. Then the sum of f over all values of the cube C_t (cube sum) is

$$\sum_{v'=(v_0,...,v_{s-1})\in C_t} f(k_0, ..., k_{n-1}, v', v_s, ..., v_{m-1}) = P \tag{2}$$

where C_t contains all binary vectors of the length s, $v_s, ..., v_{m-1}$ are fixed to constant.

The basic idea is to find enough t whose P is linear and not a constant. This enables the key recovery through solving a system of linear equations.

3.2 Huang *et al.*'s Conditional Cube Attack

Conditional cube attack [20] was proposed by Huang *et al.* to attack Keccak keyed mode, including Keccak-MAC and Keyak. We quote some definitions and a theorem here.

Definition 1. *([20]) Cube variables that have propagation controlled in the first round and are not multiplied with each other after the second round of Keccak are called **conditional cube variables**. Cube variables that are not multiplied with each other after the first round and are not multiplied with any conditional cube variable after the second round are called **ordinary cube variables**.*

Theorem 2. *([20]) For $(n+2)$-round Keccak sponge function $(n>0)$, if there are $p(0 \leq p < 2^n+1)$ conditional cube variables $v_0, ..., v_{p-1}$, and $q = 2^{n+1}-2p+1$ ordinary cube variables, $u_0, ..., u_{q-1}$ (If $q = 0$, we set $p = 2^n+1$), the term $v_0v_1...v_{p-1}u_0...u_{q-1}$ will not appear in the output polynomials of $(n+2)$-round Keccak sponge function.*

Actually, we use the special case of the above theorem when $p = 1$. We describe it as a corollary for clearness.

Corollary 1. *For $(n+2)$-round Keccak sponge function $(n>0)$, if there is one conditional cube variable v_0, and $q = 2^{n+1}-1$ ordinary cube variables, $u_0, ..., u_{q-1}$, the term $v_0u_0...u_{q-1}$ will not appear in the output polynomials of $(n+2)$-round Keccak sponge function.*

4 Modeling Search Strategy

Define $A[x][y][z] = 1$ when it is an *ordinary cube variable* or *conditional cube variable*, else $A[x][y][z] = 0$.

4.1 Modeling CP-like-kernel

In the Keccak submission document [3], the original concept is illustrated as following: if all columns in a state have even parity, θ is the identity, which is illustrated. The *conditional cube variable* used in this is set in CP-kernel to reach a reduced diffusion. At ASIACRYPT 2016, Guo *et al.* [19] assign $A[1][y]$, $y = 0, 1, 2, 3$, to be variables and $A[1][4] = \bigoplus_{i=0}^{3} A[1][y]$ so that variables in each column sum to 0. Then θ is the identity. In fact, when the parity of a column remains constant, the variables in the column do not propagate through θ operation. We denoted this property as a *CP-like-kernel*. In order to reduce the diffusion of *ordinary cube variables*, we set them as CP-like-kernel.

In CP-like-kernel, if certain column contain *ordinary cube variables*, then the number of the variables must be no less than two. If $n(n = 2, 3, 4, 5)$ bits in a column contain cube variables, we set the $n-1$ bits to be independent *ordinary cube variables* and 1 bit variable to be the sum of the $n-1$ bits. So the constraints in modeling CP-like-kernel have the following two purposes:

1. Avoid the number of bits containing cube variable in each column from being one;
2. Record which column contains cube variables.

Given x, z, suppose $A[x][y][z], y = 0, 1, 2, 3, 4$ possibly contain *ordinary cube variables*. If there exists an *ordinary cube variable* in $A[x][y][z]$ for some y, then the dummy variable $d = 1$. Else $d = 0$. Then we get the following inequalities

$$\begin{cases} \sum_{y=0}^{4} A[x][y][z] \geq 2d \\ d \geq A[x][0][z] \\ d \geq A[x][1][z] \\ d \geq A[x][2][z] \\ d \geq A[x][3][z] \\ d \geq A[x][4][z] \end{cases} \tag{3}$$

For any x, z ($x = 0, 1, ..., 4, z = 0, 1, ..., 63$), denote the corresponding dummy variable as $d[x][z]$. $d[x][z]$ records whether the column $[x][z]$ contain cube variables as illustrated above. The $[x][z]$ column can provide $\sum_{y=0}^{4} A[x][y][z] - d[x][z]$ independent cube variables. The number of independent cube variables that the whole state can provide is to sum up the ones of all columns with $x = 0, 1...4, z = 0, 1...63$. Correspondingly, the objective function of the MILP model is set as

$$\sum_{x,y,z} A[x][y][z] - \sum_{x,z} d[x][z],$$

i.e. the number of cube variables in the whole state.

4.2 Modeling The First Round

We omit the θ operation in the first round, as it does not influence the distribution of cube variables according to the property of CP-like-kernel. With the help of SAGE [1], the Keccak round function can be operated in the form of algebraic symbols. So the internal the bits of state S_1 are describe as algebraic form functions about the bits of the initial state S_0. Using a easy search program, we know which two bits in state S_0 will be multiplied in S_1. Constraints are added according to the following two conditions:

1. (a) **Condition:** Any of the *ordinary cube variables* do not multiply with each other in the first round.
   (b) **Constraint:** If two bits $S_0[x_1][y_1][z_1]$ and $S_0[x_2][y_2][z_2]$ multiply, the constraint
   $$A[x_1][y_1][z_1] + A[x_2][y_2][z_2] \leq 1$$
   will be added to avoid their simultaneous selection as *ordinary cube variables*.
2. (a) **Condition:** The *conditional cube variable* does not multiply with any of the *ordinary cube variables* in the first round.

(b) **Constraint:** If one bit $S_0[x][y][z]$ multiplies with the *conditional cube variable*, the constraint

$$A[x][y][z] = 0$$

will be added to avoid it from being selected as *ordinary cube variables*.

4.3 Modeling The Second Round

We list Property 1 for the conditions added to control the diffusion of the *conditional cube variable* v_0.

Property 1. In χ operation, denote the input and output state as X and Y respectively, one bit $X[x][y][z]$ only multiplies with two bits $X[x-1][y][z]+1$ and $X[x+1][y][z]$.

(1) If only one bit $X[x][y][z]$ contains variable v_0, conditions $X[x-1][y][z]+1 = 0$ and $X[x+1][y][z] = 0$ can avoid v_0 from diffusing by χ.
(2) If only n bits $X[x_0][y_0][z_0], X[x_1][y_1][z_1]...X[x_{n-1}][y_{n-1}][z_{n-1}]$ contain variable v_0, $2n$ conditions

$$\begin{gathered} X[x_0-1][y_0][z_0]+1 = 0, X[x_0+1][y_0][z_0] = 0, \\ X[x_1-1][y_1][z_1]+1 = 0, X[x_1+1][y_1][z_1] = 0, \\ \cdots \\ X[x_{n-1}-1][y_{n-1}][z_{n-1}]+1 = 0, X[x_{n-1}+1][y_{n-1}][z_{n-1}] = 0 \end{gathered}$$

can avoid v_0 from diffusing by χ.

1. **Condition:** Under the above conditions added to the first round, the *conditional cube variable* does not multiply with any of the *ordinary cube variables* in the second round.
2. **Constraint:** If one bit $S_0[x][y][z]$ multiplies with the *conditional cube variable*, the constraint

$$A[x][y][z] = 0$$

will be added to avoid it from being selected as *ordinary cube variables*.

5 Applications to round-reduced Keccak-MAC

5.1 Attack on 7-round Keccak-MAC-384

For Keccak-MAC-384 with 1600-bit state, rate occupies 832 bits, and capacity 768 bits. As Figure 5 shows us, 128-bit key (k_0, k_1) locates at the first two yellow lanes, and *conditional cube variable* v_0 is set in CP-like-kernel as $S_0[2][0][0] = S_0[2][1][0] = v_0$ in blue, then the white bits represent nonce or message bits,

all of which can be selected as *ordinary cube variables*, while the grey ones are initialized with all zero. Note that the lanes, which are possible to be *ordinary cube variables*, obey CP-like-kernel. List these lanes in a set $\mathbb{V}$:

$\mathbb{V} = \{[0][1], [0][2], [1][1], [1][2], [2][0], [2][1], [2][2], [3][0], [3][1], [4][0], [4][1]\}$

Additionally, the subset of $\mathbb{V}$, $\mathbb{V}_i, i = 0, 1...4$ represents the set of lanes whose x-index equals 0,1...4 respectively.

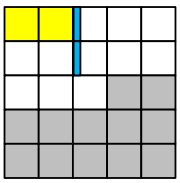


Fig. 5. The Initial State of Keccak-MAC-384

According to the modeling search strategy illustrated in Section 4, we search for the maximal number of independent *ordinary cube variables*. The objective function is

$$\sum_{x,y\in\mathbb{V},z\in\{0,1...63\}} A[x][y][z] \; - \sum_{x\in\{0,1...4\},z\in\{0,1...63\}} d[x][z],$$

To model the CP-like-kernel, constraints are in the following:

$$\begin{cases} \sum\limits_{x,y\in\mathbb{V}_x,z} A[x][y][z] \geq 2d[x][z] \\ d[x][z] \geq A[x][y][z], y \in \mathbb{V}_x \end{cases} for\ x = 0, 1...4, z = 0, 1...63 \tag{4}$$

The input state is initialized with key k, *conditional cube variable* v_0, possible *ordinary cube variables* v_i (placed in bit position $[x_i][y_i][z_i]$) and zero padding. After ρ, π, χ operation in the first round, the state is in the algebraic symbolic form of the initial state bits. If any v_0v_i exists, and the bit corresponding to i is $[x_i][y_i][z_i]$, constraint $A[x_i][y_i][z_i] = 0$ is added. If any v_iv_j exists, the bit corresponding to i, j, constraint $A[x_i][y_i][z_i] + A[x_j][y_j][z_j] \leq 1$ is added. The above constraints are to avoid any multiplication in the first round among cube variables. Additionally, we add the four bit conditions around *conditional cube variable* v_0 before the first χ operation to reduce its diffusion. After θ, ρ, π, χ operation in the second round, similarly, if any v_0v_i exists, and the bit corresponding to i is $[x_i][y_i][z_i]$, constraint $A[x_i][y_i][z_i] = 0$ is added to avoid any *ordinary cube variables* from multiplying with v_0 in the second second. With

Table 3. Parameters set for attack on 7-round Keccak-MAC-384

Ordinary Cube Variables
$A[2][0][1]=v_1$,$A[2][1][1]=v_2$,$A[2][2][1]=v_1+v_2$,$A[3][0][3]=A[3][1][3]=v_3$,
$A[2][0][5]=A[2][2][5]=v_4$,$A[1][1][7]=A[1][2][7]=v_5$,$A[2][0][8]=A[2][1][8]=v_6$,
$A[3][0][9]=A[3][1][9]=v_7$,$A[4][0][10]=A[4][1][10]=v_8$,$A[2][1][11]=A[2][2][11]=v_9$,
$A[2][0][12]=A[2][1][12]=v_{10}$,$A[4][0][12]=A[4][1][12]=v_{11}$,$A[3][0][13]=A[3][1][13]=v_{12}$,
$A[2][1][14]=A[2][2][14]=v_{13}$,$A[4][0][14]=A[4][1][14]=v_{14}$,$A[0][1][15]=A[0][2][15]=v_{15}$,
$A[1][1][15]=A[1][2][15]=v_{16}$,$A[2][1][15]=A[2][2][15]=v_{17}$,$A[2][1][18]=A[2][2][18]=v_{18}$,
$A[2][1][19]=A[2][2][19]=v_{19}$,$A[2][0][20]=v_{20}$,$A[2][1][20]=v_{21}$,$A[2][2][20]=v_{20}+v_{21}$,
$A[3][0][20]=A[3][1][20]=v_{22}$,$A[2][0][21]=A[2][2][21]=v_{23}$,$A[0][1][22]=A[0][2][22]=v_{24}$,
$A[3][0][23]=A[3][1][23]=v_{25}$,$A[2][1][24]=A[2][2][24]=v_{26}$,$A[2][0][27]=A[2][2][27]=v_{27}$,
$A[0][1][28]=A[0][2][28]=v_{28}$,$A[1][1][30]=A[1][2][30]=v_{29}$,$A[3][0][30]=A[3][1][30]=v_{30}$,
$A[0][1][32]=A[0][2][32]=v_{31}$,$A[0][1][34]=A[0][2][34]=v_{32}$,$A[1][1][34]=A[1][2][34]=v_{33}$,
$A[3][0][35]=A[3][1][35]=v_{34}$,$A[0][1][37]=A[0][2][37]=v_{35}$,$A[0][1][38]=A[0][2][38]=v_{36}$,
$A[1][1][38]=A[1][2][38]=v_{37}$,$A[1][1][39]=A[1][2][39]=v_{38}$,$A[3][0][39]=A[3][1][39]=v_{39}$,
$A[1][1][40]=A[1][2][40]=v_{40}$,$A[3][0][40]=A[3][1][40]=v_{41}$,$A[2][0][41]=A[2][1][41]=v_{42}$,
$A[2][0][43]=A[2][1][43]=v_{43}$,$A[2][0][45]=A[2][1][45]=v_{44}$,$A[0][1][46]=A[0][2][46]=v_{45}$,
$A[3][0][46]=A[3][1][46]=v_{46}$,$A[0][1][47]=A[0][2][47]=v_{47}$,$A[0][1][49]=A[0][2][49]=v_{48}$,
$A[1][1][50]=A[1][2][50]=v_{49}$,$A[2][0][50]=A[2][1][50]=v_{50}$,$A[1][1][52]=A[1][2][52]=v_{51}$,
$A[2][1][52]=A[2][2][52]=v_{52}$,$A[2][0][53]=A[2][1][53]=v_{53}$,$A[2][1][56]=A[2][2][56]=v_{54}$,
$A[3][0][56]=A[3][1][56]=v_{55}$,$A[0][1][58]=A[0][2][58]=v_{56}$,$A[2][1][58]=A[2][2][58]=v_{57}$,
$A[0][1][59]=A[0][2][59]=v_{58}$,$A[0][1][60]=A[0][2][60]=v_{59}$,$A[2][0][61]=v_{60}$,$A[2][1][61]=v_{61}$,
$A[2][2][61]=v_{60}+v_{61}$,$A[2][0][62]=v_{62}$,$A[2][1][62]=v_{63}$,
$A[2][2][62]=v_{62}+v_{63}$,$A[4][0][63]=A[4][1][63]=v_{64}$,
Conditional Cube Variable
$A[2][0][0]=A[2][1][0]=v_0$
Bit Conditions
$A[4][0][44]=A[4][1][44]+A[2][2][45]$,
$A[2][0][4]=k_0[5]+k_1[5]+A[0][1][5]+A[2][1][4]+A[0][2][5]+A[2][2][4]+1$,
$A[2][0][59]=k_0[60]+A[2][1][59]+A[2][2][59]+1$,
$A[4][0][6]=A[2][0][7]+A[2][1][7]+A[4][1][6]+A[2][2][7]+A[3][1][7]$,
$A[2][2][23]=A[2][0][23]+A[4][0][22]+A[2][1][23]+A[4][0][22]$,
$A[2][2][46]=A[2][0][46]+A[4][0][45]+A[2][1][46]+A[4][1][45]$,
$A[2][2][36]=A[2][0][36]+A[4][0][35]+A[2][1][36]+A[4][1][35]$,
$A[2][2][63]=A[2][0][63]+A[4][0][62]+A[2][1][63]+A[4][1][62]$,
$A[2][2][42]=A[2][0][42]+A[4][0][41]+A[2][1][42]+A[4][4][41]$,
$A[0][2][35]=k_0[35]+A[3][0][36]+A[0][1][35]+A[3][1][36]$,
$A[2][2][53]=k_0[54]+A[0][1][54]+A[0][2][54]$,$A[4][1][13]=A[2][0][14]+A[4][0][13]$,
$A[1][2][29]=k_1[29]+A[3][0][28]+A[1][1][29]+A[3][1][28]$,
$A[2][2][6]=k_0[7]+A[2][0][6]+A[0][1][7]+A[2][1][6]+A[0][2][7]$,
$A[1][2][45]=k_1[45]+A[3][0][44]+A[1][1][45]+A[3][1][44]$,$A[4][1][19]=A[4][0][19]$,
$A[2][2][31]=A[2][0][31]+A[4][0][30]+A[2][1][31]+A[4][1][30]$,
$A[2][2][17]=k_0[18]+A[2][0][17]+A[0][1][18]+A[2][1][17]+A[0][2][18]$
Guessed Key Bits
$k_0[5]+k_1[5]$, $k_0[60]$,$k_0[35]$,$k_0[54]$, $k_1[29]$,$k_0[7]$,$k_1[45]$,$k_0[18]$

the help of Gurobi [2], the objective function is optimized under all the above constraints, i.e. with all cube variables obeying CP-like-kernel, the maximum of cube variables is 65. Actually, 64 cube variables are enough to perform the 7-round attack on Keccak-MAC-384. Both the cube variables and conditions are listed in Table 3.

In 7-round attack on Keccak-MAC-384, $2^6 = 64$ cube variables are denoted by $v_0, v_1...v_{63}$. Based on Corollary 1, v_0 is the conditional cube variable fixed in the beginning and $v_1, v_2...v_{63}$ are ordinary cube variables found by MILP search strategy. We summarize the requirements as following:

(1) $v_0, v_1...v_{63}$ do not multiply with each other in the first round;
(2) Under some conditions on key and nonce, v_0 does not multiply with any of $v_1, v_2...v_{63}$ in the second round.

While all the nonce bits are constant, all the bit conditions are satisfied if and only if all the key bits are guessed correctly. Thus, zero sums over the 128-bit tag with cube variables set as Table 3 mean a correct key guess.

We analyze the time and data complexity of the attack: with the parameters set in Table 3, the 8 guessed key bits $k_0[5] + k_1[5]$, $k_0[60]$, $k_0[35]$, $k_0[54]$, $k_1[29]$, $k_0[7]$, $k_1[45]$, $k_0[18]$ can be recovered. The time complexity of one recovery is $2^8 * 2^{64}$. According to the property of permutation, it is totally symmetric in z-axis. Thus we can obtain corresponding parameters set with any rotation of i-bit ($0 \leq i < 64$) in z-axis. Therefore, the guessed key bits rotated i-bit i.e. $k_0[i+5] + k_1[i+5]$, $k_0[i+60]$, $k_0[i+35]$, $k_0[i+54]$, $k_1[i+29]$, $k_0[i+7]$, $k_1[i+45]$, $k_0[i+18]$ can be recovered. Through simple count, for $0 \leq i < 8$, 70 independent key bits out of 128 key bits can be recovered, 8 iterations consumes $8 \times 2^8 \times 2^{64}$ and the remaining 58 key bits are left to exhaustive search consuming 2^{58}. Combine the two parts, the procedure consumes $8 \times 2^8 \times 2^{64} + 2^{58} = 2^{75}$ computations of 7-round of Keccak-MAC-384, correspondingly 2^{75} $(message, tag)$ pairs are needed. After the procedure above, all the 128 bits in k_0, k_1 can be recovered. Therefore, both time and data complexity of the attack are 2^{75}.

5.2 Attack on 6-round Keccak-MAC-512

For Keccak-MAC-512 with 1600-bit state, rate occupies 576 bits, and capacity 1024 bits. As Figure 6 shows us, 128-bit key (k_0, k_1) locates at the first two yellow lanes, and *conditional cube variable* v_0 is set in CP-like-kernel as $S_0[2][0][0] = S_0[2][1][0] = v_0$ in blue, then the white bits represent nonce bits, but only white ones highlighted by red thick lines can be selected as *ordinary cube variables*, while the grey ones are initialized with all zero. Note that the set of lanes possible to be *ordinary cube variables* is denoted as $\mathbb{V}$:

$$\mathbb{V} = \{[2][0], [2][1], [3][0], [3][1]\}$$

Additionally, the subset of $\mathbb{V}$, $\mathbb{V}_i, i = 2, 3$ represents the set of lanes whose x-index equals 2,3 respectively. According to the modeling search strategy illustrated in

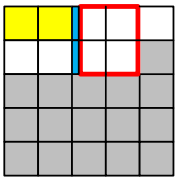


Fig. 6. The Initial State of Keccak-MAC-512

Section 4, we search the controllable bits for the most *ordinary cube variables*. The objective function is

$$\sum_{x,y\in\mathbb{V},z\in\{0,1...63\}} A[x][y][z] - \sum_{x\in\{0,1...4\},z\in\{0,1...63\}} d[x][z],$$

To model the CP-like-kernel, constraints are in the following according to Equations 3:

$$\begin{cases} \sum\limits_{x,y\in\mathbb{V}_x,z} A[x][y][z] \geq 2d[x][z] \\ d[x][z] \geq A[x][y][z], y \in \mathbb{V}_x \end{cases} for\ x = 2, 3, z = 0, 1...63 \tag{5}$$

The method of adding constraints to avoid multiplication is just the same as Keccak-MAC-384. With the help of Gurobi [2], the objective function is optimized under all the above constraints. The maximum of cube variables obeying CP-like-kernel is 26 (including a conditional cube variables). As the number of cube variables is not enough to perform the 6-round attack on Keccak-MAC-512, and many nonce bits are not utilized, we continue the search for appropriate *ordinary cube variables* among the single bits in lanes [0,1],[1,1],[4,0].

Modeling the single bits

A single bit here means it is the only bit in its column that contains cube variable, exactly, it is set as a new *ordinary cube variable*. As the optimization according to CP-like-kernel above, most cube variables have been settled. Additionally, the state is so large as 1600-bit. Although a single bit diffuse to 11 bits after the first θ operation, it may not multiply with all the other cube variables in the first round, and not multiply with *conditional cube variable* v_0 in the second round. The objective function is the sum of all possible bits to be *ordinary cube variables*. Then, constraints are added to avoid the above two kinds of multiplication in the same way.

Table 4. Parameters set for attack on 6-round Keccak-MAC-512

Ordinary Cube Variables
$A[3][0][56]=A[3][1][56]=v_1$,$A[2][0][1]=A[2][1][1]=v_2$,$A[2][0][8]=A[2][1][8]=v_3$,
$A[2][0][12]=A[2][1][12]=v_4$,$A[2][0][23]=A[2][1][23]=v_5$,$A[2][0][41]=A[2][1][41]=v_6$,
$A[2][0][43]=A[2][1][43]=v_7$,$A[2][0][45]=A[2][1][45]=v_8$,$A[2][0][50]=A[2][1][50]=v_9$,
$A[2][0][53]=A[2][1][53]=v_{10}$,$A[2][0][62]=A[2][1][62]=v_{11}$,$A[3][0][3]=A[3][1][3]=v_{12}$,
$A[3][0][4]=A[3][1][4]=v_{13}$,$A[3][0][9]=A[3][1][9]=v_{14}$,$A[3][0][12]=A[3][1][12]=v_{15}$,
$A[3][0][13]=A[3][1][13]=v_{16}$,$A[3][0][14]=A[3][1][14]=v_{17}$,$A[3][0][20]=A[3][1][20]=v_{18}$,
$A[3][0][23]=A[3][1][23]=v_{19}$,$A[3][0][27]=A[3][1][27]=v_{20}$,$A[3][0][33]=A[3][1][33]=v_{21}$,
$A[3][0][35]=A[3][1][35]=v_{22}$,$A[3][0][39]=A[3][1][39]=v_{23}$,$A[3][0][40]=A[3][1][40]=v_{24}$,
$A[3][1][46]=A[3][0][46]=v_{25}$,$A[2][1][56]=v_{26}$,$A[4][0][12]=v_{27}$,$A[2][0][56]=v_{28}$,
$A[0][1][33]=v_{29}$,$A[0][1][57]=v_{30}$,$A[4][0][60]=v_{31}$
Conditional Cube Variable
$A[2][0][0]=A[2][1][0]=v_0$
Bit Conditions
$A[4][0][44] = 0$,$A[2][0][59] = k_0[60] + A[2][1][59] + A[0][1][60] + 1$,
$A[2][0][4] = k_0[5] + k_1[5] + A[0][1][5] + 1 + A[2][1][4]$,
$A[4][0][6] = A[2][0][7] + A[2][1][7] + A[3][1][7]$,
$A[2][0][46] = A[4][0][45] + A[2][1][46]$,$A[2][0][31] = A[4][0][30] + A[2][1][31]$,
$A[4][0][3] = k_0[5] + k_1[5] + A[0][1][5] + 1$,$A[0][1][19] = k_0[19]$,
$A[3][1][30] = k_0[29] + A[3][0][30] + A[0][1][29]$,$A[0][1][34] = k_0[34]$,
$A[3][1][22] = k_0[21] + A[3][0][22] + A[0][1][21]$,$A[1][1][28] = k_1[28]$,
$A[3][1][36] = k_0[35] + A[3][0][36] + A[0][1][35]$,$A[0][1][51] = 0$,
$A[3][1][49] = k_0[48] + A[3][0][49] + A[0][1][48]$,$A[2][0][18] = A[2][1][18] + 1$,
$A[3][1][41] = k_0[40] + A[3][0][41] + A[0][1][40]$,$A[4][0][8] = k_0[8] + k_1[7] + A[1][1][7]$,
$A[3][0][63] = k_0[62] + A[0][1][60] + A[3][1][63]$,$A[4][0][51] = k_1[50] + A[1][1][50]$,
$A[2][0][51] = k_0[52] + A[0][1][52] + A[2][1][51] + 1$,
$A[2][0][63] = A[4][0][62] + A[2][1][63] + A[3][1][63]$,
$A[2][1][58] = k_0[59] + A[2][0][58] + A[0][1][59]$,
$A[4][0][13] = k_0[13] + k_1[12] + k_1[34] + A[1][1][12] + A[1][1][34] + 1$,
$A[2][0][26] = A[3][0][26] + A[4][0][25] + A[2][1][26] + 1$,
$A[3][0][16] = k_1[17] + A[1][1][17] + A[3][1][16]$,
$A[1][1][24] = k_1[24] + A[4][0][25] + A[0][1][25] + 1$,
$A[2][0][42] = A[4][0][41] + A[2][1][42] + A[3][1][42] + 1$,
$A[4][0][40] = 1$, $A[2][0][17] = k_0[18] + A[0][1][18] + A[2][1][17]$,
$A[3][0][15] = k_1[16] + A[1][1][16] + A[2][1][16] + A[3][1][15] + 1$,
$A[3][0][6] = k_0[5] + A[0][1][5] + A[3][1][6] + 1$,
$A[2][0][33] = A[2][1][33]$, $A[0][1][13] = k_0[13] + 1$,
$A[3][0][59] = k_0[58] + A[0][1][58] + A[3][1][59] + 1$,
$A[0][1][32] = k_0[32] + A[4][0][30] + A[1][1][32]$
Guessed Key Bits
$k_0[60]$, $k_0[5] + k_1[5]$, $k_0[19]$, $k_0[29]$, $k_0[34]$, $k_0[21]$, $k_0[35]$, $k_0[48]$, $k_0[40]$,
$k_0[62]$, $k_0[52]$,$k_0[59]$, $k_0[13] + k_1[12] + k_1[34]$,$k_1[17]$,$k_1[28]$,$k_1[24]$,
$k_0[8] + k_1[7]$, $k_0[18]$, $k_1[16]$,$k_0[5]$, $k_1[50]$, $k_0[13]$, $k_0[58]$, $k_0[32]$

Another 6 single bits are found as 6 new *ordinary cube variables*. Totally, we find (6+26=)32 dimension cube and based on it a 6 round key-recovery attack on Keccak-MAC-512 is achieved. Both the cube variables and conditions are listed in Table 4.

In 6-round attack on Keccak-MAC-512, $2^5 = 32$ cube variables denoted by $v_0, v_1...v_{31}$. Based on Corollary 1, v_0 is the conditional cube variable and $v_1, v_2...v_{31}$ are ordinary cube variables. We summarize the requirements as following:

(1) $v_0, v_1...v_{31}$ do not multiply with each other in the first round;
(2) Under some conditions on key and nonce, v_0 does not multiply with any of $v_1, v_2...v_{31}$ in the second round.

All the bit conditions are satisfied if and only if all the key bits are guessed correctly. Thus, zero sums over the 128-bit tag with cube variables set as Table 4 suggest a correct key guess. Furthermore, the similar key recovery can be performed with any offset in z-axis.

We analyze the time and data complexity of the attack: 4 iterations in z-axis recover 72 key bits, and the remaining 56 key bits are recovered by exhaustive search, thus the procedure consumes $4 \times 2^{24} \times 2^{32} + 2^{56} = 2^{58.3}$ computations of 6-round initialization of Keccak-MAC-512, correspondingly $2^{58.3}$ $(message, tag)$ pairs are needed. After the procedure above, all the 128 bits in k_0, k_1 can be recovered. Therefore, both time and data complexity of the attack are $2^{58.3}$.

6 Attacks on round-reduced Initialization of Ketje

At 6 March 2017, the Keccak team announces the Ketje cryptanalysis prize to encourage the cryptanalysis.

6.1 Attacks on round-reduced Initialization of Ketje Major

Ketje Major operates on a 1600-bit state, the recommended key length is 128-bit, which is similar to Keccak-MAC. We focus on the instances with recommended 128-bit key. The number of nonce bits in Ketje Major is variable from 0 to 1454.
To explore the resistance against conditional cube attack of the different instances, we apply the MILP search strategy to search the possible cube variables in the instances with different lengths of nonce, and list the corresponding number of cube variables in Table 5. Similar to attacks on Keccak-MAC described in Section 5.1, 5.2, 32 cube variables are needed to perform 6-round attack, and 64 cube variables are needed to perform 7-round attack. Thus, Table 5 tells us that when the nonce is no less than 704 bits (11 lanes), cube variables are enough to

Table 5. The number of cube variables in CP-like-kernel in different nonces in Ketje Major

nonce: bits(lanes)	number of cube variables in CP-like-kernel
448(7)	21
512(8)	41
576(9)	50
640(10)	59
704(11)	75
832(13)	81

Table 6. Parameters set for attack on 6-round Ketje Major

Ordinary Cube Variables
$A[4][1][2] = A[4][4][2] = v_1$, $A[4][1][4] = A[4][4][4] = v_2$, $A[4][1][10] = A[4][4][10] = v_3$, $A[4][1][11] = A[4][4][11] = v_4$, $A[3][0][14] = A[3][3][14] = v_5$, $A[3][0][17] = A[3][3][17] = v_6$, $A[4][1][19] = A[4][4][19] = v_7$, $A[4][1][20] = A[4][4][20] = v_8$, $A[4][1][27] = A[4][4][27] = v_9$, $A[3][0][28] = A[3][3][28] = v_{10}$, $A[4][1][28] = A[4][4][28] = v_{11}$, $A[3][0][33] = A[3][3][33] = v_{12}$, $A[3][0][36] = A[3][3][36] = v_{13}$, $A[3][0][37] = A[3][3][37] = v_{14}$, $A[4][1][38] = A[4][4][38] = v_{15}$, $A[3][0][45] = A[3][3][45] = v_{16}$, $A[4][1][59] = A[4][4][59] = v_{17}$, $A[4][1][60] = A[4][4][60] = v_{18}$, $A[2][2][18] = A[2][4][18] = v_{19}$, $A[2][2][19] = A[2][4][19] = v_{20}$, $A[2][2][51] = A[2][4][51] = v_{21}$, $A[2][2][27] = A[2][4][27] = v_{22}$, $A[2][2][28] = A[2][4][28] = v_{23}$, $A[2][2][52] = A[2][4][52] = v_{24}$, $A[2][2][53] = A[2][4][53] = v_{25}$, $A[2][2][36] = A[2][4][36] = v_{26}$, $A[2][2][37] = A[2][4][37] = v_{27}$, $A[2][2][39] = A[2][4][39] = v_{28}$, $A[2][2][55] = A[2][4][55] = v_{29}$, $A[2][2][60] = A[2][4][60] = v_{30}$, $A[2][2][62] = A[2][4][62] = v_{31}$
Conditional Cube Variable
$A[3][0][0] = A[3][3][0] = v_0$
Bit Conditions
$A[3][3][41] = k_1[42] + A[1][0][42] + A[3][0][41] + A[2][2][42] + A[1][3][42] + 1$, $A[4][4][7] = A[3][0][7] + A[0][2][6] + A[3][3][7]$, $A[2][4][31] = k_1[31] + A[1][0][31] + A[3][0][30] + A[1][3][31] + A[3][3][30] + 1$, $A[3][3][8] = A[3][0][8] + A[4][1][8] + A[0][2][7]$, $A[4][4][49] = A[2][1][50] + A[4][1][49] + A[2][2][50] + A[3][3][50] + A[2][4][50]$, $A[2][4][11] = A[2][1][11] + A[3][3][11] + 1$, $A[2][4][61] = A[2][1][61] + A[2][2][61] + A[3][3][61]$, $A[0][2][38] = k_0[30] + k_1[38] + A[2][1][37] + 1$, $A[4][4][12] = A[2][1][13] + A[4][1][12] + A[3][3][13] + A[2][4][13]$
Guessed Key Bits
$k_1[42]$,$k_1[31]$,$k_0[30] + k_1[38]$

perform 7-round attack on Ketje Major and 6-round attack on Ketje Major can be performed if the nonce is no less than 512 bits (8 lanes).

As instances with more nonce bits can directly use the parameters of instances with less nonce bits, we list the details of 6-round and 7-round attacks on Ketje Major with 512-bit and 704-bit nonce.

Attack on 6-round Initialization of Ketje Major According to parameters set in Table 6, guess the 3 key bits listed, compute cube sums on variables $v_0, ..., v_{31}$, zero cube sums suggest a right key(i.e. 3 guessed key bits in Table 6). It consumes $2^3 \times 2^{32} = 2^{35}$ computations of 6-round initialization of KETJE MAJOR. According to the property of permutation, it is totally symmetric in z-axis. Thus we can obtain corresponding parameters set with any rotation of i-bit ($0 \leq i < 64$) in z-axis. Therefore, 128 key bits can be recovered by 64 iterations for $0 \leq i < 64$, so the time complexity is $64 \times 2^3 \times 2^{32} = 2^{41}$.

Attack on 7-round Initialization of Ketje Major We use $A[1][0][0]$=$A[1][3][0]$=v_0 as condition cube variable. According to parameters set in Table 7, guess the 16 key bits listed, compute cube sums on variables $v_0, ..., v_{63}$, zero cube sums suggest a right key(i.e. 16 guessed key bits in Table 7). It consumes $2^{16} \times 2^{64} = 2^{80}$ computations of 7-round initialization of KETJE MAJOR. Similar to the case above, 46 key bits can be recovered by 4 iterations for $0 \leq i < 4$, and the remaining 82 key bits can be recovered by exhaustive search. The time complexity is $4 \times 2^{16} \times 2^{64} + 2^{82} = 2^{83}$.

6.2 Attacks on round-reduced Initialization of Ketje Minor

The state of Ketje Minor is 800-bit, which is the half of the state size of Keccak-MAC (1600-bit). As the upper part of Figure 7 shows, in Huang *et al.*'s attack on Keccak-MAC, one *conditional cube variable* v_0 is chosen, placed in two black bits of S_0. After adding some conditions, the conditional cube variable v_0 is diffused to 22 bits shown in state $S_{1.5}$, we denote the diffusion pattern as 2-2-22. For the state of Ketje Minor is much smaller, the *conditional cube variable* in 2-2-22 pattern diffuses relatively much greater, there are only 26 *ordinary cube variables* in CP-like-kernel optimized with MILP search strategy, which is not enough for 7-round attack.

In order to solve the problem, we find a new *conditional cube variable*. As shown in the lower part of Figure 7, after adding some conditions, the diffusion pattern is 6-6-6 and only 6 bits in $S_{1.5}$ contains the conditional cube variable. At last, we find enough *ordinary cube variables* with the MILP tool to launch the key-recovery attacks on 5/6/7-round reduced Ketje Minor.

Table 7. Parameters set for attack on 7-round KETJE MAJOR

Ordinary Cube Variables
$A[3][2][0]=A[3][3][0]=v_1$,$A[1][0][1]=A[1][3][1]=v_2$,$A[4][1][4]=A[4][4][4]=v_3$,
$A[3][0][5]=v_4$,$A[3][2][5]=v_5$,$A[3][3][5]=v_4+v_5$,$A[1][0][7]=A[1][3][7]=v_6$,
$A[1][0][9]=A[1][3][9]=v_7$,$A[3][2][9]=A[3][3][9]=v_8$,$A[4][1][9]=A[4][4][9]=v_9$,
$A[3][0][10]=v_{10}$,$A[3][2][10]=v_{11}$,$A[3][3][10]=v_{10}+v_{11}$,$A[4][1][10]=A[4][4][10]=v_{12}$,
$A[3][2][11]=A[3][3][11]=v_{13}$,$A[4][1][11]=A[4][4][11]=v_{14}$,$A[1][0][12]=A[1][3][12]=v_{15}$,
$A[3][2][15]=A[3][3][15]=v_{16}$,$A[1][0][17]=A[1][3][17]=v_{17}$,$A[1][0][19]=A[1][3][19]=v_{18}$,
$A[4][1][20]=A[4][4][20]=v_{19}$,$A[4][1][26]=A[4][4][26]=v_{20}$,$A[3][0][27]=A[3][2][27]=v_{21}$,
$A[1][0][29]=A[1][3][29]=v_{22}$,$A[3][2][30]=A[3][3][30]=v_{23}$,$A[3][2][31]=A[3][3][31]=v_{24}$,
$A[1][0][32]=A[1][3][32]=v_{25}$,$A[1][0][33]=A[1][3][33]=v_{26}$,$A[4][1][33]=A[4][4][33]=v_{27}$,
$A[3][0][38]=A[3][2][38]=v_{28}$,$A[1][0][39]=A[1][3][39]=v_{29}$,$A[3][0][41]=A[3][3][41]=v_{30}$,
$A[3][0][42]=A[3][2][42]=v_{31}$,$A[1][0][43]=A[1][3][43]=v_{32}$,$A[3][0][43]=A[3][3][43]=v_{33}$,
$A[3][0][45]=A[3][2][45]=v_{34}$,$A[3][0][46]=v_{35}$,$A[3][2][46]=v_{36}$,$A[3][3][46]=v_{35}+v_{36}$,
$A[3][0][47]=A[3][2][47]=v_{37}$,$A[3][0][48]=A[3][2][48]=v_{38}$,$A[3][0][49]=v_{39}$,
$A[3][2][49]=v_{40}$,$A[3][3][49]=v_{39}+v_{40}$,$A[3][2][50]=A[3][3][50]=v_{41}$,
$A[3][2][51]=A[3][3][51]=v_{42}$,$A[3][2][52]=A[3][3][52]=v_{43}$,$A[4][1][52]=A[4][4][52]=v_{44}$,
$A[3][2][53]=A[3][3][53]=v_{45}$,$A[3][0][56]=v_{46}$,$A[3][2][56]=v_{47}$,$A[3][3][56]=v_{46}+v_{47}$,
$A[3][2][60]=A[3][3][60]=v_{48}$,$A[4][1][61]=A[4][4][61]=v_{49}$,$A[1][0][62]=A[1][3][62]=v_{50}$,
$A[3][2][63]=A[3][3][63]=v_{51}$,$A[2][2][20]=A[2][4][20]=v_{52}$,$A[2][1][26]=A[2][4][26]=v_{53}$,
$A[1][0][4]=A[1][3][4]=v_{54}$,$A[2][2][33]=A[2][4][33]=v_{55}$,$A[2][1][35]=v_{56}$,
$A[2][2][35]=v_{57}$,$A[2][4][35]=v_{56}+v_{57}$,$A[2][1][40]=A[2][2][40]=v_{58}$,
$A[2][1][44]=A[2][2][44]=v_{59}$,$A[2][2][45]=A[2][4][45]=v_{60}$,$A[2][2][54]=A[2][4][54]=v_{61}$,
$A[2][1][23]=A[2][2][23]=v_{62}$,$A[1][0][2]=A[1][3][2]=v_{63}$
Bit Conditions
$A[4][4][42]=k_1[41] + A[1][0][41] + A[4][1][42] + A[0][2][42] + A[1][3][41] + 1$,
$A[2][4][48]=k_0[38] + k_1[48] + A[1][0][48] + A[1][3][48] + A[0][2][46]$,
$A[4][4][47]=k_1[46] + A[1][0][46] + A[4][1][47] + A[1][3][46] + 1$,
$A[3][3][58]=k_1[59] + A[1][0][59] + A[3][0][58] + A[2][1][59] + A[3][2][58] + A[1][3][59]$,
$A[3][3][17]=k_0[8] + A[3][0][17] + A[0][2][16] + A[3][2][17]$,
$A[3][3][26]=k_0[17] + A[3][0][26] + A[0][2][25] + A[3][2][26]$,
$A[3][3][27]=k_0[18] + A[0][2][26]$, $A[3][3][47]=k_0[38] + A[0][2][46]$,
$A[3][3][7]=k_1[8] + A[1][0][8] + A[3][0][7] + A[3][2][7] + A[1][3][8]$,
$A[3][3][48]=k_0[39] + A[0][2][47]$,$A[4][4][44]=A[2][1][45] + A[4][1][44] + A[3][3][45]$,
$A[3][3][55]=k_0[46] + A[3][0][55] + A[0][2][54] + A[3][2][55]$,
$A[4][4][41]=A[2][0][42] + A[2][1][42] + A[4][1][41] + A[3][3][42] + A[2][4][42]$,
$A[4][4][46]=k_1[45] + A[1][0][45] + A[4][1][46] + A[0][2][46] + A[1][3][45] + 1$,
$A[2][4][52]=k_1[52] + A[1][0][52] + A[3][0][51] + A[1][3][52]$,
$A[0][2][43]=k_0[35] + k_1[43] + A[2][0][42] + A[2][1][42] + A[2][4][42] + 1$,
$A[1][3][61]=k_1[61] + A[1][0][61] + A[3][0][60] + A[2][1][61]$,
$A[0][2][44]=k_1[43] + A[2][1][45] + A[3][3][45] + 1$
Guessed Key Bits
$k_1[41]$,$k_0[38] + k_1[48]$,$k_1[46]$,$k_1[59]$,$k_0[8]$,$k_0[17]$,$k_0[18]$,$k_0[38]$,$k_1[8]$,$k_0[39]$,$k_0[46]$,
$k_1[45]$,$k_1[52]$,$k_0[35] + k_1[43]$,$k_1[61]$,$k_1[43]$

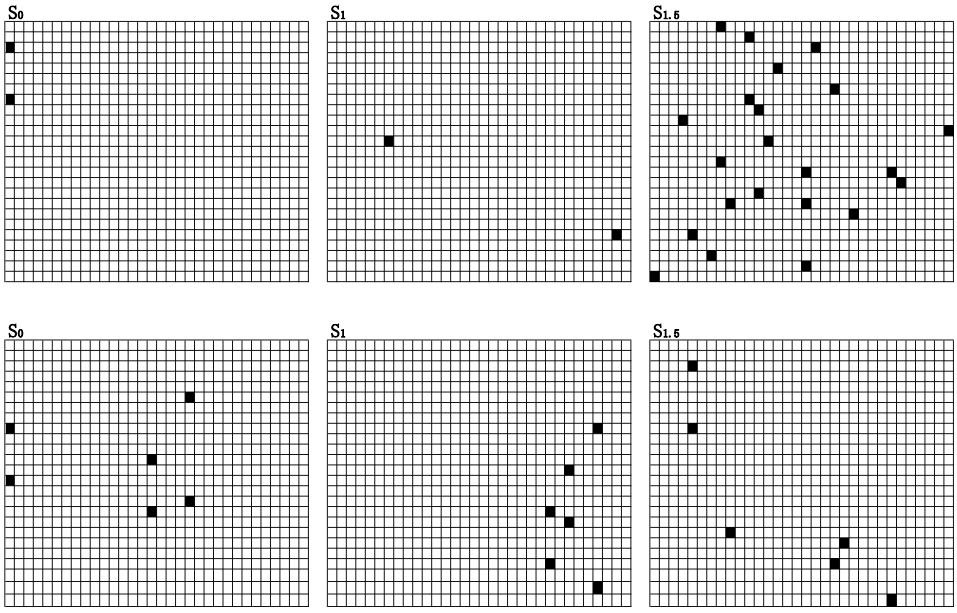


Fig. 7. Diffusions of the Conditional Cube Variable in 2-2-22 and 6-6-6 Pattern in Ketje Minor

In details, from S_0 to S_1, $\theta, \rho, \pi, \chi, \iota$ are operated in sequence. θ operation holds the distribution of v_0 according to CP-like-kernel. Operations ρ and π only permute the bit positions, while ι only adds a constant. Thus, we only need to control the diffusion of χ operation. We denote the state before χ operation in the first round as $S_{0.5}$. According to Property 1-(2), 12-bit conditions based on key and nonce are introduced to keep the 6 bits containing v_0 from diffusion. Then the diffusion of v_0 maintains the 6-6-6 pattern.

Attack on 5-round Initialization of Ketje Minor In 5-round attack, we choose $2^4 = 16$ cube variables denoted by $v_0, v_1...v_{15}$. Based on Corollary 1, v_0 is the conditional cube variable and $v_1, v_2...v_{15}$ are ordinary cube variables. We summarize the requirements as following:

(1) $v_0, v_1...v_{15}$ do not multiply with each other in the first round;
(2) Under some conditions on key and nonce, v_0 does not multiply with any of $v_1, v_2...v_{15}$ in the second round.

Under (1), any of cube variables $v_0, v_1...v_{15}$ only exists as a one-degree term in the output of 1-round Ketje Minor, i.e. the degree of any bit in S_1 is no more than one. The degree of one round function is 2. When we say the degree of some

Table 8. Parameters Set for Attack on the 5-round Initialization of Ketje Minor

Conditional Cube Variable
$A[0][1][19]=A[0][3][19]=A[1][2][15]=A[1][3][15]=A[3][1][0]=A[3][2][0]=v_0$
Ordinary Cube Variables
$A[2][0][2]=A[2][1][2]=v_1$,$A[2][0][4]=A[2][1][4]=v_2$,$A[2][0][7]=A[2][1][7]=v_3$, $A[2][0][11]=A[2][1][11]=v_4$,$A[2][0][12]=A[2][1][12]=v_5$,$A[2][0][20]=A[2][1][20]=v_6$, $A[2][0][23]=A[2][1][23]=v_7$,$A[2][0][29]=A[2][1][29]=v_8$,$A[2][0][30]=A[2][1][30]=v_9$, $A[3][0][3]=A[3][1][3]=v_{10}$,$A[3][0][6]=A[3][1][6]=v_{11}$,$A[3][0][12]=A[3][1][12]=v_{12}$, $A[3][0][13]=A[3][1][13]=v_{13}$,$A[3][0][17]=A[3][1][17]=v_{14}$,$A[3][0][21]=A[3][1][21]=v_{15}$.
Bit Conditions
$A[1][0][24]=k_1[24] + A[4][0][25] + A[4][1][25] + A[0][2][25] + A[1][2][24]$ $+ A[4][2][25] + A[1][3][24] + A[4][3][25] + A[1][4][24] + A[4][4][25] + 1$ $A[1][0][31]=k_1[31] + k_3[30] + A[3][0][30] + A[3][1][30] + A[1][2][31]$ $+ A[3][2][30] + A[1][3][31] + A[2][4][31] + A[3][4][30] + 1$ $A[1][0][19]=k_1[19] + k_3[18] + A[3][0][18] + A[2][1][19] + A[3][1][18]$ $+ A[1][2][19] + A[3][2][18] + A[1][3][19] + A[1][4][19] + A[3][4][18] + 1$ $A[0][1][16]=k_0[8] + k_3[17] + A[0][2][16] + A[3][2][17] + A[0][3][16]$ $+ A[4][3][17] + A[0][4][16] + A[3][4][17]$ $A[0][1][13]=k_0[5] + k_2[12] + A[0][2][13] + A[1][2][13] + A[0][3][13]$ $+ A[2][3][12] + A[0][4][13] + A[2][4][12]$ $A[1][0][20]=k_1[20] + A[4][0][21] + A[0][1][21] + A[4][1][21] + A[1][2][20]$ $+ A[4][2][21] + A[1][3][20] + A[4][3][21] + A[1][4][20] + A[4][4][21] + 1$ $A[1][0][10]=k_1[10] + k_3[9] + A[3][0][9] + A[3][1][9] + A[1][2][10] + A[3][2][9]$ $+ A[1][3][10] + A[2][3][10] + A[1][4][10] + A[3][4][9]$ $A[0][1][27]=k_0[19] + k_3[28] + A[3][0][28] + A[4][0][28] + A[3][1][28]$ $+ A[0][2][27] + A[3][2][28] + A[0][3][27] + A[0][4][27] + A[3][4][28] + 1$ $A[0][1][15]=k_0[7] + k_3[16] + A[3][0][16] + A[3][1][16] + A[0][2][15]$ $+ A[3][2][16] + A[4][2][16] + A[0][3][15] + A[0][4][15] + A[3][4][16]$ $A[0][1][20]=k_0[12] + k_3[21] + A[0][2][20] + A[3][2][21] + A[4][2][21]$ $+ A[0][3][20] + A[0][4][20] + A[3][4][21] + 1$ $A[0][1][26]=k_0[18] + k_2[25] + A[2][0][25] + A[2][1][25] + A[0][2][26]$ $+ A[0][3][26] + A[2][3][25] + A[0][4][26] + A[1][4][26] + A[2][4][25]$ $A[1][0][25]=k_1[25] + k_3[24] + A[2][0][25] + A[3][0][24] + A[3][1][24]$ $+ A[1][2][25] + A[3][2][24] + A[1][3][25] + A[1][4][25] + A[3][4][24] + 1$
Guessed Key Bits
$k_1[24]$,$k_1[31] + k_3[30]$,$k_1[19] + k_3[18]$,$k_0[8] + k_3[17]$,$k_0[5] + k_2[12]$, $k_1[20]$,$k_1[10] + k_3[9]$,$k_0[19] + k_3[28]$,$k_0[7] + k_3[16]$,$k_0[12] + k_3[21]$, $k_0[18] + k_2[25]$,$k_1[25] + k_3[24]$

state, we mean the highest degree among the cube variables in all terms of the state. If conditions in (2) are met, according to Corollary 1, the term $v_0v_1...v_{15}$ will not appear in S_5, so the degree over cube variables $v_0, v_1...v_{15}$ is at most 15. Otherwise, the degree of S_5 is 16.

Thus, under given conditions on key and nonce, the cube sums of all bits in S_5 over $v_0, v_1...v_{15}$ are zero, otherwise the cube sums are random if those conditions are not met. Actually, $\rho = 128$ bits of S_5 are known in Ketje Minor. If the cube sum on each of the 128 bits is zero, we can determine that the corresponding conditions are satisfied.

As Table 8 shows, the 12 bit conditions are related to key and nonce bits. We guess the 12 key bits with all the possible values. While all the nonce bits are constant, all the bit conditions are satisfied if and only if all the key bits are guessed correctly. Thus, zero sums over the 128 known bits of S_5 ($S_5[0][0], S_5[1][1], S_5[2][2], S_5[3][3]$[5]) with cube variables set as Table 8 mean a correct key guess. We give an example here for intuition, in which key is generated randomly and all the controllable nonce bits are set as zero.

128-bit key ($K = k_0||k_1||k_2||k_3||k_4$):
`10100000110101100111010011011100011100100001110111011100101101101111111001001110100101100010101010001011110100011110010110000010100`
The correct value for the guessed key bits in Table 8 is `110111101010`.
guessed value: `000000000000`,
cube sums: `0xf0217c64, 0x8a61f7e1, 0x67f01330, 0xa9b1c06`
...
guessed value: `110111101010`,
cube sums: `0x0, 0x0, 0x0, 0x0`
...
guessed value: `000011010110`,
cube sums: `0xf4c1bc4, 0xea79d2a4, 0xc2880990, 0x8ae4140d`
...
guessed value: `111111111111`,
cube sums: `0x7b115312, 0xa9156874, 0x9cabc23, 0x6ecd5ef9`

Furthermore, we can perform the similar key recovery with any offset $0, 1...31$ in z-axis. We analyze the time and data complexity of the attack: the procedure consumes $32 \times 2^{12} \times 2^{16} = 2^{33}$ computations of 5-round initialization of Ketje Minor, correspondingly 2^{33} $(nonce, plaintext, ciphertext, tag)$ pairs are needed. After the procedure above, all the 120 bits in k_0, k_1, k_2, k_3 can be recovered, and the remaining 8 bits of k_4 can be determined by brute search. Therefore, time complexity of the attack is 2^{33} computations of 5-round initialization of Ketje Minor, and data complexity is 2^{33} $(nonce, plaintext, ciphertext, tag)$ pairs.

[5] These four 32-bit words are the first four words after π which are the output bits of Keccak-p^*.

Attack on 6-round Initialization of Ketje Minor In 6-round attack, similar to the 5-round attack, we choose $2^5 = 32$ cube variables denoted by $v_0, v_1...v_{31}$. Based on Corollary 1, v_0 is the conditional cube variable and $v_1, v_2...v_{31}$ are ordinary cube variables. We summarize the requirements as following:

(1) $v_0, v_1...v_{31}$ do not multiply with each other in the first round;
(2) Under some conditions on key and nonce, v_0 does not multiply with any of $v_1, v_2...v_{31}$ in the second round.

The recovery attack can be performed similarly to 5-round attack. While all the nonce bits are constant, all the bit conditions are satisfied if and only if all the key bits are guessed correctly. Thus, zero sums over the 128 known bits of S_6 ($S_6[0][0], S_6[1][1], S_6[2][2], S_6[3][3]$) with conditional cube variable set as Table 8 and ordinary cube variables set as Table 9 mean a correct key guess. We give an example here for intuition, in which key is generated randomly and all the controllable nonce bits are set as zero.

128-bit key ($K = k_0||k_1||k_2||k_3||k_4$):
`10010110000010011000101001010110101011101101100111001001110110100011111111010110110100111011111001010001011011100011100111011011011`
The correct value for the guessed key bits in Table 8 is `100001001100`.
guessed value: `000000000000`,
cube sums: `0x555b48a6, 0xcce8cd70, 0x9e41800d, 0x66b12d4f`
...
guessed value: `100001001100`,
cube sums: `0x0, 0x0, 0x0, 0x0`
...
guessed value: `010101101100`,
cube sums: `0xc61fa207, 0x24f02427, 0x3fed45e0, 0x36a8326d`
...
guessed value: `111111111111`,
cube sums: `0x834061d2, 0x14200817, 0xd56d2379, 0xc93e01f8`

We analyze the time and data complexity of the attack: the procedure consumes $32 \times 2^{12} \times 2^{32} = 2^{49}$ computations of 6-round initialization of Ketje Minor, correspondingly 2^{49} $(nonce, plaintext, ciphertext, tag)$ pairs are needed. After the procedure above, all the 120 bits in k_0, k_1, k_2, k_3 can be recovered, and the remaining 8 bits in k_4 can be determined by brute search. Therefore, both time and data complexity of the attack are 2^{49}.

Attack on 7-round Initialization of Ketje Minor In 7-round attack, similar to the 5/6-round attack, we choose $2^6 = 64$ cube variables denoted by $v_0, v_1...v_{63}$. Based on Corollary 1, v_0 is the conditional cube variable and $v_1, v_2...v_{63}$ are ordinary cube variables. We summarize the requirements as following:

Table 9. Ordinary Cube Variables and Bit Conditions for Attack on the 6-round Initialization of Ketje Minor

Ordinary Cube Variables
$A[2][0][2]=A[2][1][2]=v_1$,$A[2][0][4]=A[2][1][4]=v_2$,$A[2][0][7]=A[2][1][7]=v_3$, $A[2][0][11]=A[2][1][11]=v_4$,$A[2][0][12]=A[2][1][12]=v_5$,$A[2][0][20]=A[2][1][20]=v_6$, $A[2][0][23]=A[2][1][23]=v_7$,$A[2][0][29]=A[2][1][29]=v_8$,$A[2][0][30]=A[2][1][30]=v_9$, $A[3][0][3]=A[3][1][3]=v_{10}$,$A[3][0][6]=A[3][1][6]=v_{11}$,$A[3][0][12]=A[3][1][12]=v_{12}$, $A[3][0][13]=A[3][1][13]=v_{13}$,$A[3][0][17]=A[3][1][17]=v_{14}$,$A[3][0][21]=A[3][1][21]=v_{15}$, $A[3][0][22]=A[3][1][22]=v_{16}$,$A[3][0][26]=A[3][1][26]=v_{17}$,$A[3][0][31]=A[3][1][31]=v_{18}$, $A[4][0][0]=A[4][1][0]=v_{19}$,$A[4][0][5]=A[4][1][5]=v_{20}$,$A[4][0][8]=A[4][1][8]=v_{21}$, $A[4][0][15]=A[4][1][15]=v_{22}$,$A[4][0][18]=A[4][1][18]=v_{23}$,$A[4][0][22]=A[4][1][22]=v_{24}$, $A[4][0][24]=A[4][1][24]=v_{25}$,$A[0][2][1]=A[0][3][1]=v_{26}$,$A[0][2][5]=A[0][3][5]=v_{27}$, $A[0][2][10]=A[0][3][10]=v_{28}$,$A[0][2][15]=A[0][3][15]=v_{29}$,$A[0][2][31]=A[0][3][31]=v_{30}$, $A[1][2][4]=A[1][3][4]=v_{31}$.
Bit Conditions
$A[1][0][24]=k_1[24] + A[4][0][25] + A[4][1][25] + A[0][2][25] + A[1][2][24] + A[4][2][25] + A[1][3][24] + A[4][3][25] + A[1][4][24] + A[4][4][25] + 1$
$A[1][0][31]=k_1[31] + k_3[30] + A[3][0][30] + A[3][1][30] + A[1][2][31] + A[3][2][30] + A[1][3][31] + A[2][4][31] + A[3][4][30] + 1$
$A[1][0][19]=k_1[19] + k_3[18] + A[3][0][18] + A[2][1][19] + A[3][1][18] + A[1][2][19] + A[3][2][18] + A[1][3][19] + A[1][4][19] + A[3][4][18] + 1$
$A[0][1][16]=k_0[8] + k_3[17] + A[0][2][16] + A[3][2][17] + A[0][3][16] + A[4][3][17] + A[0][4][16] + A[3][4][17]$
$A[0][1][13]=k_0[5] + k_2[12] + A[0][2][13] + A[1][2][13] + A[0][3][13] + A[2][3][12] + A[0][4][13] + A[2][4][12]$
$A[1][0][20]=k_1[20] + A[4][0][21] + A[0][1][21] + A[4][1][21] + A[1][2][20] + A[4][2][21] + A[1][3][20] + A[4][3][21] + A[1][4][20] + A[4][4][21] + 1$
$A[1][0][10]=k_1[10] + k_3[9] + A[3][0][9] + A[3][1][9] + A[1][2][10] + A[3][2][9] + A[1][3][10] + A[2][3][10] + A[1][4][10] + A[3][4][9]$
$A[0][1][27]=k_0[19] + k_3[28] + A[3][0][28] + A[4][0][28] + A[3][1][28] + A[0][2][27] + A[3][2][28] + A[0][3][27] + A[0][4][27] + A[3][4][28] + 1$
$A[0][1][15]=k_0[7] + k_3[16] + A[3][0][16] + A[3][1][16] + A[3][2][16] + A[4][2][16] + A[0][4][15] + A[3][4][16]$
$A[0][1][20]=k_0[12] + k_3[21] + A[0][2][20] + A[3][2][21] + A[4][2][21] + A[0][3][20] + A[0][4][20] + A[3][4][21] + 1$
$A[0][1][26]=k_0[18] + k_2[25] + A[2][0][25] + A[2][1][25] + A[0][2][26] + A[0][3][26] + A[2][3][25] + A[0][4][26] + A[1][4][26] + A[2][4][25]$
$A[1][0][25]=k_1[25] + k_3[24] + A[2][0][25] + A[3][0][24] + A[3][1][24] + A[1][2][25] + A[3][2][24] + A[1][3][25] + A[1][4][25] + A[3][4][24] + 1$

(1) $v_0, v_1 ... v_{63}$ do not multiply with each other in the first round;

(2) Under some conditions on key and nonce, v_0 does not multiply with any of $v_1, v_2 ... v_{63}$ in the second round.

While all the nonce bits are constant, all the bit conditions are satisfied if and only if all the key bits are guessed correctly. Thus, zero sums over the 128 known bits of S_7 ($S_7[0][0], S_7[1][1], S_7[2][2], S_7[3][3]$) with conditional cube variable set as Table 8 and ordinary cube variables set as Table 10 mean a correct key guess.

We analyze the time and data complexity of the attack: the procedure consumes $32 \times 2^{12} \times 2^{64} = 2^{81}$ computations of 7-round initialization of Ketje Minor, correspondingly 2^{81} $(nonce, plaintext, ciphertext, tag)$ pairs are needed. After the procedure above, all the 120 bits in k_0, k_1, k_2, k_3 can be recovered, and the remaining 8 bits in k_4 can be determined by brute search. Therefore, both time and data complexity of the attack are 2^{81}.

Table 10. Ordinary Cube Variables for Attack on the 7-round Initialization of Ketje Minor

Ordinary Cube Variables
$A[0][1][0]=v_1$,$A[0][2][0]=v_2$,$A[0][3][0]=v_3$,$A[0][4][0]=v_1 + v_2 + v_3$,$A[0][2][1]=v_4$,
$A[0][3][1]=v_5$,$A[0][4][1]=v_4 + v_5$,$A[0][1][2]=A[0][3][2]=v_6$,$A[0][2][3]=A[0][4][3]=v_7$,
$A[0][1][4]=v_8$,$A[0][3][4]=v_9$,$A[0][4][4]=v_8 + v_9$,$A[0][1][5]=v_{10}$,$A[0][2][5]=v_{11}$,
$A[0][3][5]=v_{10} + v_{11}$,$A[0][1][6]=v_{12}$,$A[0][2][6]=v_{13}$,$A[0][4][6]=v_{12} + v_{13}$,
$A[0][1][8]=A[0][3][8]=v_{14}$,$A[0][1][9]=v_{15}$,$A[0][2][9]=v_{16}$,$A[0][3][9]=v_{17}$,
$A[0][4][9]=v_{15} + v_{16} + v_{17}$,$A[0][2][10]=v_{18}$,$A[0][3][10]=v_{19}$,$A[0][4][10]=v_{18} + v_{19}$,
$A[0][1][13]=v_{20}$,$A[0][3][13]=v_{21}$,$A[0][4][13]=v_{20} + v_{21}$,$A[0][1][14]=v_{22}$,$A[0][2][14]=v_{23}$,
$A[0][4][14]=v_{22} + v_{23}$,$A[0][1][15]=v_{24}$,$A[0][2][15]=v_{25}$,$A[0][3][15]=v_{24} + v_{25}$,
$A[0][1][16]=v_{26}$,$A[0][2][16]=v_{27}$,$A[0][4][16]=v_{26} + v_{27}$,$A[0][2][17]=A[0][4][17]=v_{28}$,
$A[0][2][19]=A[0][4][19]=v_{29}$,$A[0][3][21]=A[0][4][21]=v_{30}$,$A[0][1][22]=v_{31}$,
$A[0][2][22]=v_{32}$,$A[0][3][22]=v_{33}$,$A[0][4][22]=v_{31} + v_{32} + v_{33}$,$A[0][2][23]=A[0][4][23]=v_{34}$,
$A[0][1][24]=v_{35}$,$A[0][3][24]=v_{36}$,$A[0][4][24]=v_{35} + v_{36}$,$A[0][1][25]=v_{37}$,$A[0][3][25]=v_{38}$,
$A[0][4][25]=v_{37} + v_{38}$,$A[0][1][27]=A[0][4][27]=v_{39}$,$A[0][1][30]=A[0][2][30]=v_{40}$,
$A[0][1][31]=v_{41}$,$A[0][2][31]=v_{42}$,$A[0][3][31]=v_{43}$,$A[0][4][31]=v_{41} + v_{42} + v_{43}$,
$A[2][0][1]=A[2][4][1]=v_{44}$,$A[2][0][2]=v_{45}$,$A[2][1][2]=v_{46}$,$A[2][3][2]=v_{47}$,
$A[2][4][2]=v_{45} + v_{46} + v_{47}$,$A[2][1][3]=v_{48}$,$A[2][3][3]=v_{49}$,$A[2][4][3]=v_{48} + v_{49}$,
$A[2][0][4]=v_{50}$,$A[2][1][4]=v_{51}$,$A[2][3][4]=v_{50} + v_{51}$,$A[2][0][5]=A[2][4][5]=v_{52}$,
$A[2][0][7]=v_{53}$,$A[2][1][7]=v_{54}$,$A[2][3][7]=v_{53} + v_{54}$,$A[2][0][9]=A[2][3][9]=v_{55}$,
$A[2][0][10]=A[2][4][10]=v_{56}$,$A[2][0][11]=v_{57}$,$A[2][1][11]=v_{58}$,$A[2][4][11]=v_{57} + v_{58}$,
$A[2][0][12]=v_{59}$,$A[2][1][12]=v_{60}$,$A[2][4][12]=v_{59} + v_{60}$,$A[2][1][13]=v_{61}$,$A[2][3][13]=v_{62}$,
$A[2][4][13]=v_{61} + v_{62}$,$A[2][1][14]=A[2][4][14]=v_{63}$.

7 Conclusion

In this paper, we comprehensively study the conditional cube attack against Keccak keyed modes. In order to find enough ordinary cube variables in low degrees of freedom of Keccak keyed modes, we introduce a new MILP model. We show how to model the CP-like-kernel and model the way that the ordinary

Table 11. Bit Conditions for Attack on the 7-round Initialization of Ketje Minor

Bit Conditions
$A[1][0][24]=k_1[24] + A[4][0][25] + A[4][1][25] + A[0][2][25] + A[1][2][24]$ $+ A[4][2][25] + A[1][3][24] + A[4][3][25] + A[1][4][24] + A[4][4][25] + 1$
$A[1][0][31]=k_1[31] + k_3[30] + A[3][0][30] + A[3][1][30] + A[1][2][31]$ $+ A[3][2][30] + A[1][3][31] + A[2][4][31] + A[3][4][30] + 1$
$A[1][0][19]=k_1[19] + k_3[18] + A[3][0][18] + A[2][1][19] + A[3][1][18]$ $+ A[1][2][19] + A[3][2][18] + A[1][3][19] + A[1][4][19] + A[3][4][18] + 1$
$A[3][0][17]=k_0[8] + k_3[17] + A[3][1][17] + A[3][2][17] + A[0][3][16]$ $+ A[4][3][17] + A[3][4][17]$
$A[1][2][13]=k_0[5] + k_2[12] + A[0][2][13] + A[2][3][12]$
$A[1][0][20]=k_1[20] + A[4][0][21] + A[0][1][21] + A[4][1][21] + A[1][2][20]$ $+ A[4][2][21] + A[1][3][20] + A[4][3][21] + A[1][0][4][20] + A[4][4][21] + 1$
$A[1][0][10]=k_1[10] + k_3[9] + A[3][0][9] + A[3][1][9] + A[1][2][10] + A[3][2][9]$ $+ A[1][3][10] + A[2][3][10] + A[1][4][10] + A[3][4][9]$
$A[3][0][28]=k_0[19] + k_3[28] + A[4][0][28] + A[3][1][28] + A[0][2][27]$ $+ A[3][2][28] + A[0][3][27] + A[3][4][28] + 1$
$A[3][0][16]=k_0[7] + k_3[16] + A[3][1][16] + A[3][2][16] + A[4][2][16]$ $+ A[0][4][15] + A[3][4][16]$
$A[3][0][21]=k_0[12] + k_3[21] + A[0][1][20] + A[3][1][21] + A[0][2][20]$ $+ A[3][2][21] + A[4][2][21] + A[0][3][20] + A[0][4][20] + A[3][4][21] + 1$
$A[1][4][26]=k_0[18] + k_2[25] + A[2][0][25] + A[0][1][26] + A[2][1][25]$ $+ A[0][2][26] + A[0][3][26] + A[2][3][25] + A[0][4][26] + A[2][4][25]$
$A[1][0][25]=k_1[25] + k_3[24] + A[2][0][25] + A[3][0][24] + A[3][1][24]$ $+ A[1][2][25] + A[3][2][24] + A[1][3][25] + A[1][4][25] + A[3][4][24] + 1$

cube variables do not multiply together in the first round as well as do not multiply with the conditional cube variable in the second round. Then, a series of linear inequality system are brought out, which accurately restrict the way to add an ordinary cube variable. Then, by choosing the objective function of the maximal number of ordinary cube variables, we convert Huang *et al.*'s greedy algorithm into an MILP problem and maximal number of ordinary cube variables is determined. Based on this method, we extend the best previous attacks on round-reduced Keccak-MAC-384 and Keccak-MAC-512 by 1 round, and achieve the first 7-round and 6-round key-recovery attacks, respectively. In addition, we

give some results on Ketje Major and Minor and get the best results on these two ciphers.

Currently, the cryptanalysis progress of symmetric-key ciphers heavily depends on automated evaluation tools. For many reasons, the cryptanalysis of the new SHA-3 standard Keccak is very hard and limited, more evaluation tools on Keccak are urgently needed. The MILP method introduced in this paper enriches the Keccak tools, and helps academic communities study Keccak much easier.

Acknowledgments

We would like to thank the anonymous reviewers of Asiacrypt 2017 who helped us to improve this paper. This work is supported by China's 973 Program (No. 2013CB834205), the National Key Research and Development Program of China (No. 2017YFA0303903), the National Natural Science Foundation of China (No. 61672019,61402256), the Fundamental Research Funds of Shandong University (No. 2016JC029), National Cryptography Development Fund(No.MMJJ20170121), Zhejiang Province Key R&D Project (No. 2017C01062).

References

1. Http://www.sagemath.org/
2. Http://www.gurobi.com/
3. Berton, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak sponge function family, `http://keccak.noekeon.org/`
4. Berton, G., Daemen, J., Peeters, M., Assche, G.V., Keer, R.V.: CAESAR submission: Ketje v2 (2016), `http://competitions.cr.yp.to/round3/ketjev2.pdf`
5. Berton, G., Daemen, J., Peeters, M., Assche, G.V., Keer, R.V.: CAESAR submission: Keyak v2 (2016), `http://competitions.cr.yp.to/round3/keyakv22.pdf`
6. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the sponge: Single-pass authenticated encryption and other applications. In: Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers. pp. 320–337 (2011), `http://dx.doi.org/10.1007/978-3-642-28496-0_19`
7. Boura, C., Canteaut, A., De Canniere, C.: Higher-order differential properties of Keccak and luffa. In: FSE. vol. 6733, pp. 252–269. Springer (2011)
8. Cui, T., Jia, K., Fu, K., Chen, S., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations. IACR Cryptology ePrint Archive 2016, 689 (2016), `http://eprint.iacr.org/2016/689`
9. Daemen, J., Van Assche, G.: Differential propagation analysis of Keccak. In: FSE. vol. 7549, pp. 422–441. Springer (2012)
10. Dinur, I., Dunkelman, O., Shamir, A.: New attacks on Keccak-224 and Keccak-256. In: Fast Software Encryption. pp. 442–461. Springer (2012)

11. Dinur, I., Dunkelman, O., Shamir, A.: Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In: International Workshop on Fast Software Encryption. pp. 219–240. Springer (2013)
12. Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Cube attacks and cube-attack-like cryptanalysis on the round-reduced KECCAK sponge function. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. pp. 733–761 (2015), http://dx.doi.org/10.1007/978-3-662-46800-5_28
13. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. pp. 278–299 (2009), http://dx.doi.org/10.1007/978-3-642-01001-9_16
14. Dobraunig, C., Eichlseder, M., Mendel, F.: Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In: Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II. pp. 490–509 (2015), http://dx.doi.org/10.1007/978-3-662-48800-3_20
15. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Cryptanalysis of Ascon. In: Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings. pp. 371–387 (2015), http://dx.doi.org/10.1007/978-3-319-16715-2_20
16. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1. 2. Submission to the CAESAR Competition (2016)
17. Dong, X., Li, Z., Wang, X., Qin, L.: Cube-like attack on round-reduced initialization of KETJE SR. IACR Trans. Symmetric Cryptol. 2017(1), 259–280 (2017), http://tosc.iacr.org/index.php/ToSC/article/view/594
18. Duc, A., Guo, J., Peyrin, T., Wei, L.: Unaligned rebound attack: Application to KECCAK. In: Fast Software Encryption. pp. 402–421. Springer (2012)
19. Guo, J., Liu, M., Song, L.: Linear structures: Applications to cryptanalysis of round-reduced KECCAK. In: Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. pp. 249–274 (2016), http://dx.doi.org/10.1007/978-3-662-53887-6_9
20. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional cube attack on reduced-round KECCAK sponge function. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II. pp. 259–288 (2017), http://dx.doi.org/10.1007/978-3-319-56614-6_9
21. Jean, J., Nikolić, I.: Internal differential boomerangs: practical analysis of the round-reduced KECCAK-f permutation. In: International Workshop on Fast Software Encryption. pp. 537–556. Springer (2015)

22. Li, Z., Dong, X., Wang, X.: Conditional cube attack on round-reduced ascon. IACR Transactions on Symmetric Cryptology 2017(1), 175–202 (2017)
23. Mella, S., Daemen, J., Assche, G.V.: New techniques for trail bounds and application to differential trails in KECCAK. IACR Trans. Symmetric Cryptol. 2017(1), 329–357 (2017), http://tosc.iacr.org/index.php/ToSC/article/view/597
24. Morawiecki, P., Pieprzyk, J., Srebrny, M.: Rotational cryptanalysis of round-reduced KECCAK. In: International Workshop on Fast Software Encryption. pp. 241–262. Springer (2013)
25. Morawiecki, P., Pieprzyk, J., Srebrny, M.: Rotational cryptanalysis of round-reduced KECCAK. In: Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers. pp. 241–262 (2013), http://dx.doi.org/10.1007/978-3-662-43933-3_13
26. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: International Conference on Information Security and Cryptology. pp. 57–76. Springer (2011)
27. Qiao, K., Song, L., Liu, M., Guo, J.: New collision attacks on round-reduced KECCAK. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 216–243. Springer (2017)
28. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III. pp. 185–215 (2017), http://dx.doi.org/10.1007/978-3-319-56617-7_7
29. Song, L., Liao, G., Guo, J.: Non-full sbox linearization: Applications to collision attacks on round-reduced KECCAK. In: Annual International Cryptology Conference. pp. 428–451. Springer (2017)
30. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 158–178. Springer (2014)
31. Wu, S., Wang, M.: Security evaluation against differential cryptanalysis for block cipher structures. IACR Cryptology ePrint Archive 2011, 551 (2011)
32. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. pp. 648–678 (2016), http://dx.doi.org/10.1007/978-3-662-53887-6_24