New MILP Modeling: Improved Conditional Cube Attacks to Keccak-based Constructions

Ling Song[1,2,3], Jian Guo[2], and Danping Shi[1,3]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences, China
[2] Nanyang Technological University, Singapore
[3] Data Assurance and Communication Research Center,
Chinese Academy of Sciences, China

{songling.alpha,ntu.guo}@gmail.com, shidanping@iie.ac.cn

Abstract. In this paper, we provide a new MILP modeling to find better/optimal choices of conditional cubes. These choices generally find new or improved attacks against the keyed constructions based on Keccak permutations, including Keccak-MAC, KMAC, Kravatte, Keyak, and Ketje, in terms of attack complexities or the number of attacked rounds. Specifically, we find new key recovery attacks against KMAC128 and KMAC256, which are NIST standard way of constructing MAC from SHA-3, reduced to 7 and 9 rounds respectively. For Kravatte, up to 10 out of 14 rounds can be attacked similarly. The best attack against Lake Keyak with 128-bit keys is improved from 6 to 8 rounds in the nonce respected setting and 9 rounds of Lake Keyak can be attacked if the key size is 256. Attack complexity improvements are found generally on other constructions. To verify the correctness of our attacks, reduced-variants of the attacks against KMAC are implemented and tested on a PC practically.

Keywords: Keccak, SHA-3, KMAC, Kravatte, Keyak, Ketje, conditional cube attack, MILP

1 Introduction

The Keccak hash function family [4] is a proposal designed by Bertoni *et al.* and submitted to the SHA-3 competition [17] in 2008. It was selected as the final winner of the competition in 2012 , and subsequently standardized as SHA-3 [21] in 2015 by the National Institute of Standards and Technology of the U.S. (NIST). It supports four digest sizes from $\{224, 256, 384, 512\}$ to achieve different security levels. The standard SHA-3 and the original Keccak design differ only in the way how messages are padded, hence they share almost all security analysis.

Since the Keccak hash function was made public in 2008, it has attracted intensive cryptanalysis from the research community in many different settings. Against the three major properties of hash functions: collision, preimage and second-preimage resistance, the best practical collision/preimage attacks are up

to 6 and 4 out of the total 24 rounds, respectively. By observing the low algebraic degree of the Sbox in KECCAK, Guo *et al.* [13] proposed the linear structures for up to 3 rounds of KECCAK, where the Sbox can be re-expressed as linear transformations when the input is restricted to specific affine subspaces. In [19], Song *et al.* [19] found the first practical collision against 5-round KECCAK-224, where they used 3-round "connectors" based on the pioneer work by Qiao *et al.* [18] and Dinur *et al.* [9]. There is also a line of research on analyzing the security of keyed constructions based on KECCAK-p— the KECCAK permutations with variable width and rounds.

Message authentication codes are naturally among the first keyed constructions based on KECCAK-p, *e.g.*, KECCAK-MAC [3] and `KMAC` [22]. In [10], Dinur *et al.* proposed the first cube attack against KECCAK-MAC for up to 7-round key recovery and 8-round forgery attacks. The attack complexities were subsequently improved by Huang *et al.* using conditional cube attacks [14]. The authenticated encryption schemes KEYAK [6] and KETJE [5] are also based on KECCAK-p and its variants. Similar to the attacks against KECCAK-MAC, the conditional cube attack was applied to KEYAK for up to 8 out of 12 rounds [14], and to KETJE [12,15] for up to 7 out of 13 rounds.

Following a similar design strategy used for KECCAK-MAC, `KMAC` [22] is the standard way of constructing MAC from `SHA-3` by NIST. The major design difference is that, the master key is processed as an independent data block before processing the message in `KMAC`, while it was processed together with some message bits as the first data block in KECCAK-MAC. Hence, at the point of injecting the first message block, the internal state for `KMAC` is totally unknown, while the most bits of that for KECCAK-MAC are known. Similar observations were discovered and made use of in the so-called "Full-State Keyed Sponge (FKS)" [16] to improve the efficiency of keyed sponge constructions. It is interesting to note, despite the great similarity of KECCAK-MAC and `KMAC`, there is no existing cryptanalysis results against `KMAC` to the best of our knowledge.

More recently, a new permutation-based construction for building a psuedorandom function named Farfalle was proposed by Bertoni *et al.* [1]. An instance of this construction, named KRAVATTE, was proposed together based on KECCAK-p. Similar to `KMAC`, a key dependent variable was added to the internal state where the message block is injected, hence unknown to attackers. In what follows, we call these constructions with *fully unknown internal states*.

Our contributions. Based on the previous works [12,14,15] on conditional cube attacks against KECCAK-based keyed constructions, we provide a new MILP modeling. While the length of cube tester (the zero-sum property) is determined entirely by the algebraic degrees of the underlying permutations, the conditional cube attack could only be improved by finding cube variables with lesser conditions and keep the cube size large enough in the meanwhile. Our new MILP modeling is able to capture the characteristics of 2 KECCAK rounds, as well as the linear structures used in the first round. This new modeling is generic and im-

Table 1: Summary of our attacks on KMAC, KRAVATTE, and KECCAK-MAC with related works.

Target	Key Size	Capacity	n_r Rounds	Complexity	Reference
KMAC128	128	256	7	2^{76}	Section 5
KMAC256	256	512	9	2^{147}	
KRAVATTE	128	-	8	2^{65}	Section 6.1
	256	-	9	2^{129}	
	320	-	10	2^{257}	
KECCAK-MAC	128	256/512	7	2^{72}	[14]
		768	7	2^{75}	[15]
		1024	6	$2^{58.3}$	
		1024	6	2^{41}	Section 4

poses no unnecessary conditions, hence could be able to find optimal conditional cube variables whenever possible. This comes with a few key observations:

1. Instead of the initial state, the internal state value just before the first Sbox layer are used as (conditional) variables by setting the variables in the column parity kernel. This simple change removes all the unnecessary constraints brought up by the linear layer of the first KECCAK round, and enlarges the space covered by our search program.
2. We are able to model 2 KECCAK rounds together, i.e., Sbox layer of the first round, linear layer followed by Sbox layer again of the second round. To do this, we exhaustively list the propagations of variables through the Sbox so to keep the output of the Sbox linear. To deal with the linear layer of the second round, we divide the θ operation into two cases depending on whether there is spreading of variables and model them each individually. With all these together, we are able to convert all the necessary constraints in the search of better conditional cubes into the MILP language.

We apply this new MILP modeling to KECCAK-based keyed constructions including KECCAK-MAC, KMAC, KRAVATTE, KEYAK, and KETJE, and find generally new or better results for each of the constructions. Specifically

– For KMAC, due to the fact that it processes the key as an independent block compared with KECCAK-MAC, it should provide better security and hence becomes harder for attacker. With the same security level of 128 bits, we find attacks against KMAC128 reduced to 7 rounds, the same number of rounds found for KECCAK-MAC in previous works. For KMAC256 aiming for 256 bits security, we find attacks up to 9 rounds combining a technique to invert the last round.
– 8, 9 and 10 rounds could be attacked for KRAVATTE with 128, 256 and 320 bit keys, as summarized in Table 1.
– General complexity improvements are also found on the attacks against KEYAK and KETJE. Notably, we improve the attack against Lake KEYAK

Table 2: Summary of our attacks on KEYAK, KETJE and comparison with related works

Target	Key Size	n_r Rounds	Complexity	Nonce respected	Reference
Lake KEYAK	128	6	2^{37}	Yes	[10]
	128	8	$2^{71.01}$	Yes	Section 6.2
	128	8	2^{74}	No	[14]
	256	9	$2^{137.05}$	Yes	Section 6.2
River KEYAK	128	8	2^{77}	Yes	
KETJE Major	128	7	2^{83}	Yes	[15]
	128	7	$2^{71.24}$	Yes	Section 6.2
KETJE Minor	128	7	2^{81}	Yes	[15]
	128	7	$2^{73.03}$	Yes	Section 6.2
KETJE SR v1[‡]	128	7	2^{115}	Yes	[12]
	128	7	2^{92}	Yes	Section 6.2

[‡] For KETJE SR, better attacks are found only for v1, while for KETJE Major and KETJE Minor, better attacks are found for both v1 and v2 but only results for the latest v2 are listed due to space limit.

with 128-bit keys from 6 to 8 rounds in the nonce respected setting and 9 rounds of Lake KEYAK can be attacked if the key size is 256. Details are summarized in Table 2.

Organization. The rest of the paper is organized as follows. Section 2 gives a detailed description of KECCAK-p based constructions, including KECCAK, `KMAC`, KRAVATTE, KEYAK and KETJE, followed by an introduction in Section 3 to the related works. Our new MILP method is presented in Section 4, and applied to the key recovery attack of `KMAC` in Section 5. The application of our new model to KRAVATTE, KEYAK and KETJE is introduced in 6. Finally, Section 7 concludes the paper. Some technical details of the attacks are postponed to Appendix.

2 Description of `KMAC`, KRAVATTE, KEYAK and KETJE

2.1 KECCAK-p

The KECCAK-p permutations are specified with two parameters: the width of the permutation in bits $\boldsymbol{b}$ and the number of rounds $\boldsymbol{n_r}$. The KECCAK-p permutation with $\boldsymbol{n_r}$ rounds and width $\boldsymbol{b}$ is denoted by KECCAK-$p[\boldsymbol{b}, \boldsymbol{n_r}]$, where $\boldsymbol{n_r}$ is any positive integer and $\boldsymbol{b}$ can be any value of the form $25{\cdot}2^l$ for $l = 0, \ldots, 6$. The $\boldsymbol{b}$-bit state a for the KECCAK-$p[\boldsymbol{b}, \boldsymbol{n_r}]$ permutation is seen as a three-dimensional array of bits, namely $a[5][5][w]$ with $w = 2^l$. The expression $a[x][y][z]$ with $0 \leq x, y < 5$, $0 \leq z < w$, denotes the bit with (x, y, z) coordinate. The coordinates are always considered within modulo 5 for x and y and modulo w for z. The one-dimensional

portion $a[*][y][z]$ is called a *row*, $a[x][*][z]$ a *column* and $a[x][y][*]$ a *lane*. A lane of the state is also denoted by $a[x][y]$ by omitting the z index. At lane level, the state $a[x][y]$ becomes a 5×5 array as shown in Figure 1 with x for the column index and y for the row index.

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

Figure 1: Lane coordinates. Each square stands for a lane in the state.

The KECCAK-$p[\boldsymbol{b}, \boldsymbol{n_r}]$ permutation iterates an identical round function (up to a difference of round-dependent constant addition) $\boldsymbol{n_r}$ times, each of which consists of five bijective mappings $\mathtt{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$, with details as follows.

$$
\begin{aligned}
\theta:&\ A[x][y][z] \leftarrow A[x][y][z] + \Sigma_{y=0}^{4} A[x-1][y][z] + \Sigma_{y=0}^{4} A[x+1][y][z-1],\\
\rho:&\ A[x][y][z] \leftarrow A[x][y][(z+T(x,y))], \text{where } T(x,y)s \text{ are pre-defined rotation constants,}\\
\pi:&\ A[y][2x+3y][z] \leftarrow A[x][y][z],\\
\chi:&\ A[x][y][z] \leftarrow A[x][y][z] + ((A[x+1][y][z]+1) \cdot A[x+2][y][z]),\\
\iota:&\ A[0][0] \leftarrow A[0][0] + RC_{i_r}, \text{where } RC_{i_r} \text{ is the round constant for the } i_r\text{-th round.}
\end{aligned}
$$

Here, '+' denotes XOR and '$\cdot$' denotes logic AND. Expressions in the x and y coordinates should, as mentioned, be taken in modulo 5 and expressions in the z coordinate modulo w.

The KECCAK-f family of permutations is a specification of the KECCAK-p family to the case of $\boldsymbol{n_r} = 12 + 2l$, that is KECCAK-$f[\boldsymbol{b}]$ = KECCAK-$p[\boldsymbol{b}, 12 + 2l]$. The permutation underlying `SHA-3` and `KMAC` is of width 1600 bits and 24 rounds, *i.e.*, KECCAK-$f[1600]$ = KECCAK-$p[1600, 24]$.

2.2 The sponge construction and `KMAC`

The sponge construction is a framework for constructing hash functions from permutations, as depicted in Fig. 2. The construction consists of three components: an underlying $\boldsymbol{b}$-bit permutation f, a parameter $\boldsymbol{r}$ called *rate* and a padding rule. The *capacity* is defined as $\boldsymbol{c} := \boldsymbol{b} - \boldsymbol{r}$. A hash function following this construction takes in a message M as input and outputs a digest of $\boldsymbol{d}$ bits. Given the message M, it is first padded and split into $\boldsymbol{r}$-bit blocks. The $\boldsymbol{b}$-bit state is initialized to be all zeros. The sponge construction then proceeds in two phases. In the absorbing phase, each message block is XORed into the first $\boldsymbol{r}$

bits of the state, followed by application of the permutation f. This process is repeated until all message blocks are processed. Then, the sponge construction switches to the squeezing phase, where each iteration returns the first $\boldsymbol{r}$ bits of the state as output and then applies the permutation f to the current state. This repeats until all $\boldsymbol{d}$ bits digest are obtained.

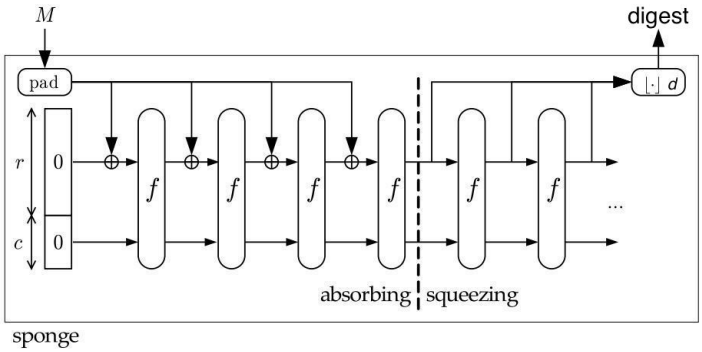


Figure 2: Sponge Construction [2].

The KECCAK hash function follows the sponge construction and takes KECCAK-$f[\boldsymbol{b}]$ as the underlying permutation. In 2015, KECCAK was formally standardized by NIST as `SHA-3` [21], based on which more functions, including cSHAKE128, cSHAKE256 and `KMAC`, are derived in the NIST Special Publication 800-185 [22].

`KMAC` (KECCAK Message Authentication Code) is a keyed hash function with a variable-length output, and can be used as a pseudorandom function. It has two variants: `KMAC128` and `KMAC256`, based on KECCAK[c=256](M, L) and KECCAK[c=512](M, L), whose capacities are set to be 256 and 512 bits, respectively. The input of `KMAC` consists of the key K, the main message M, the output length L, the name string N = "KMAC" and the optional customization bit string S of any length (including 0). Given these inputs, `KMAC` first processes a block encoded from the public values N and S. Then it accepts a block of the padded key, and absorbs message blocks from the third call of permutation f onwards. Figure 3 demonstrates the procedure of `KMAC` processing one message block. Different from KECCAK, `KMAC` supports variable length output, e.g., `KMAC128` supports any output of length no less than 256 bits and at least 512 bits for `KMAC256`.

2.3 The Farfalle construction and KRAVATTE

Farfalle [1] is a permutation-based construction for building pseudorandom functions, as shown in Figure 4. It takes a key of variable-length and a message sequence as input, and outputs a bit stream of desired length. Farfalle has three parts: a key derivation, a compression layer and an expansion layer, which makes use of four permutations p_b, p_c, p_d, and p_e, and three rolling functions $roll_c, roll_e$, and $roll_f$. First, the key derivation generates $\boldsymbol{b}$-bit masks from the key using

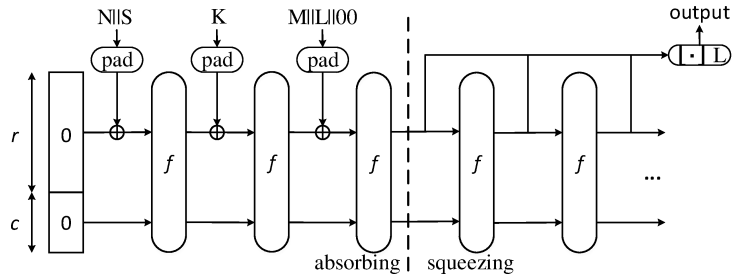
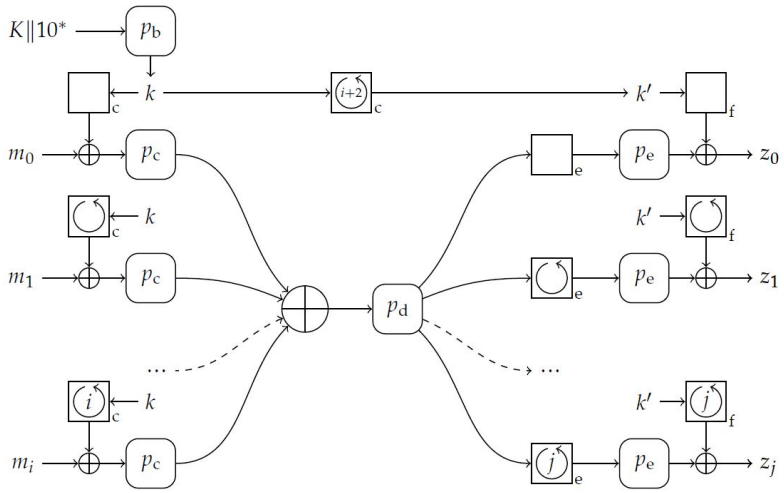


Figure 3: `KMAC` processing one message block



▢ stands for permutations and ▢ symbolizes rolling functions.

Figure 4: The Farfalle construction [1].

p_b, $roll_c$, and $roll_f$. These masks derived from the key are used for pre/post-whitening. Then, the compression layer computes a $\boldsymbol{b}$-bit accumulator from the message sequence by the parallel application of p_c. Finally, the expansion layer computes rolling states from the accumulator using $roll_e$, and passes the rolling states to p_e to generate the output. Due to the inherent parallelism of the Farfalle construction, Farfalle instances can be very efficient.

Proposed in [1], KRAVATTE is a Farfalle instance based on KECCAK-$p[\boldsymbol{b}, \boldsymbol{n_r}]$. Specifically,

$$p_b = p_c = \text{KECCAK-}p[1600, 6],$$
$$p_d = p_e = \text{KECCAK-}p[1600, 4].$$

$roll_f$ is the identity, and $roll_e = roll_c$ is a linear transformation to the five lanes $a[*][4]$ of the KECCAK-p state:

$$\begin{aligned} a[x][4] &\leftarrow a[x+1][4], \quad x \in 0,1,2,3 \\ a[4][4][z] &\leftarrow a[0][4][z-7] + a[1][4][z], \quad z > 60 \\ a[4][4][z] &\leftarrow a[0][4][z-7] + a[1][4][z] + a[1][4][z+3], \quad z \leq 60. \end{aligned}$$

Besides, $roll^i$ means repeatedly applying the rolling function i times. The suggested key size of KRAVATTE is less than or equal to 320 bits.

2.4 KEYAK and KETJE

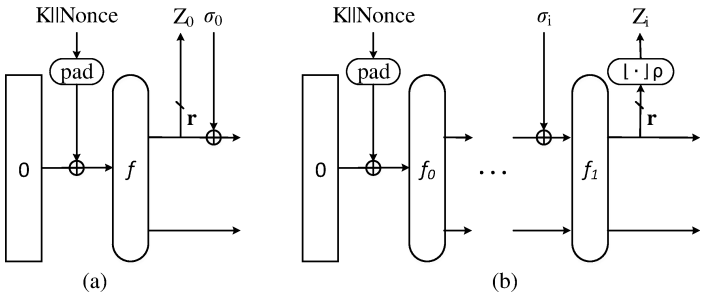


Figure 5: (a) KEYAK and (b) KETJE

KEYAK and KETJE [5, 6] are two KECCAK-p based authenticated encryption schemes, both of which are CAESAR candidates. Figure 5 (a) shows the scheme of KEYAK processing one message block. KEYAK has five instances. In this paper, we focus on River KEYAK and Lake KEYAK which are based on KECCAK-$p[800, 12]$ and KECCAK-$p[1600, 12]$ respectively. The capacity for both versions is 256. It is noted that any attack on Lake KEYAK is also applicable to the rest three instances.

Figure 5 (b) displays the scheme of KETJE processing message blocks. It employs a twisted version of KECCAK-p, denoted by KECCAK-$p^\star$, where KECCAK-$p^\star$ $=\pi\circ$KECCAK-p $\circ\pi^{-1}$. Specifically, the underlying permutations f_0 =KECCAK-$p[\boldsymbol{b}, 12]$ and f_1 =KECCAK-$p[\boldsymbol{b}, 1]$. KETJE has four instances which are:

Name	$\boldsymbol{b}$	ρ
KETJE JR	200	16
KETJE SR	400	32
KETJE Minor	800	128
KETJE Major	1600	256

In the old version of KETJE, KECCAK-p, instead of KECCAK-$p^\star$, is used.

2.5 Notations

In this paper, $\boldsymbol{r}$ and $\boldsymbol{c}$ in bold denote the rate and capacity for the sponge construction. $\boldsymbol{b}$ in bold stands for the width in bits of the permutation. The first three mappings θ, π, ρ of the round function of Keccak-p are linear, and we denote their composition by $\lambda \triangleq \pi \circ \rho \circ \theta$. The nonlinear layer χ applying to each row is called an Sbox. Unless otherwise stated, only one-block padded messages are considered in our attacks for `KMAC`. The message block, whether it is a $\boldsymbol{r}$-bit one for `KMAC` or a $\boldsymbol{b}$-bit one for Kravatte, is denoted by $a[x][y][z]$, $0 \leq x, y < 5$, $0 \leq z < 64$, and let $b = \lambda(a)$, $c = \chi(b)$.

3 Related Works

3.1 Cube attacks

The cube attack, a variant of higher order differential attacks, was introduced by Dinur and Shamir [11] in 2009. It considers the output bit of a cipher as an unknown Boolean polynomial $f(k_0, ..., k_{n-1}, v_0, ..., v_{m-1})$ where $k_0, ..., k_{n-1}$ are secret input variables and $v_0, ..., v_{m-1}$ are public input variables. Given a monomial t_I, the multiplication of all variables from a set I, any Boolean polynomial f can be written as the sum of terms which are supersets of t_I and terms that are not divisible by t_I:

$$f(k_0, ..., k_{n-1}, v_0, ..., v_{m-1}) = t_I \cdot p_{S_I} + q(k_0, ..., k_{n-1}, v_0, ..., v_{m-1}),$$

where p_{S_I} is called the superpoly of I in f. The basic idea of cube attacks and cube testers is that the sum of the outputs over the cube C_I which contains all possible bit vectors for variables in I is exactly p_{S_I}, while this is a random function for a random polynomial. By carefully selecting I, cube attacks aim to find a low-degree polynomial p_{S_I} in secret bits, and cube testers aim to distinguish p_{S_I} from a random function.

In [10], Dinur *et al.* applied cube attacks and cube testers to the keyed variants of Keccak, including Keccak-MAC, Keyak and a Keccak stream cipher.

3.2 Conditional cube attacks

In [14], Huang *et al.* developed conditional cube testers for keyed Keccak sponge function, where the propagation of certain cube variables are controlled in the first few rounds if some conditions are satisfied. There are two major advantages of conditional cube testers over ordinary cube testers. One is to potentially reduce the algebraic degree of the permutation under the conditions, and hence the required cube dimension to carry out the attack can be reduced accordingly. The other advantage of conditional cubes is that, the conditions, which control how the conditional cube variables propagate in the first few rounds, are related to the initial state values, which may contain the key information. By observing the cube sum of the final output, one may recover the key.

To proceed further, we provide the definition of conditional cube variables and a theorem from [14] below.

Definition 1 ([14]). *Cube variables that have propagation controlled in the first round and are not multiplied with each other in the second round of* Keccak *are* ***called conditional cube variables****. Cube variables that are not multiplied with each other in the first round and are not multiplied with any conditional cube variable in the second round are called* ***ordinary cube variables****.*

Theorem 1 ([14]). *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* $p = 2^n + 1$*), then the term* $v_0 v_1 ... v_{p-1} u_0 ... u_{q-1}$ *will not appear in the output polynomials of* $(n+2)$*-round* Keccak *sponge function.*

Using conditional cube testers, better key recovery attacks were obtained for Keccak-MAC and Keyak in [14]. Later, the attacks on Keccak-MAC were further improved with better conditional cubes found by an MILP medel in [15].

In previous works [14,15], the number of conditional cube variables is chosen to be 1, i.e., $p = 1$. Then, over a conditional cube with dimension 2^n, the cube sum is zero for $(n+1)$-round Keccak sponge function if the conditions are satisfied. Conditional cubes with more conditional cube variables may exist, but they are not so helpful. The reasons are as follows. Even though a greater p reduces the required dimension, it also reduces the largest dimension available due to an increasing number of conditions. On the other hand, more conditions do not improve the complexity of the key recovery attack. Therefore, we also set $p = 1$ in our attacks to be presented in Section 5 and 6.

3.3 Linear structures

In the cube attacks of keyed variants of Keccak [10], Dinur *et al.* proposed a method for linearizing the first round of Keccak-f. Inspired by this method, Guo *et al.* [13] developed a technique named *linear structure* which allows linearization of Keccak-f for up to 3 rounds. Based on the linear structures, a series of new zero-sum distinguishers of Keccak-f were proposed, as well as several new preimage attacks against Keccak.

Let $a[x, y]$, $x = 0, 2$, $y = 0, 1, 2, 3$ be variables and $a[x, 4] = \bigoplus_{y=0}^{3} a[x, y] \oplus \alpha_x$ with any constant α_x so that variables in each column sum to a constant. The core idea is to reduce the diffusion effect of θ. With all columns sum to constants, the variables do not propagate through θ. Note θ is the only mapping from λ with diffusion property, so λ does not diffuse the variables under this setting. Figure 6 shows how the variables influence the internal state under the transformation of Keccak-f round function $\mathtt{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$. All bits of the lanes with orange slashes have algebraic degree 1, those lanes in orange have algebraic degree at most 1 (meaning it is either a variable of degree 1 or a constant), and the other lanes are all constants where gray, light gray and white bits stand for values 1, 0,

and arbitrary constants, respectively. Note the algebraic degrees remain through the linear operations θ, ρ, π, and ι. The only non-linear operation is the χ which increases the algebraic degree through the AND operation of two neighboring bits. As shown in the figure, all variables before χ are not adjacent to each other, which makes sure that the algebraic degree of the state bits remains at most 1 after one round function `R`.

Moreover, bit 1 (0) on the left (right) of the variable helps to restrict the diffusion, while an unknown constant diffuses the variable in an uncertain way, as denoted by orange lanes where the bits may be variables or constants. This structure has degrees of freedom 512. Also, it can be regarded as a cube of dimension 512. Inspired by this linear structure, the diffusion effect of variables through χ are carefully studied in the next section, and a new MILP model is provided for searching conditional cube attacks for Keccak-p based constructions, especially finding conditional cubes with minimal bit conditions for constructions with fully unknown internal state.

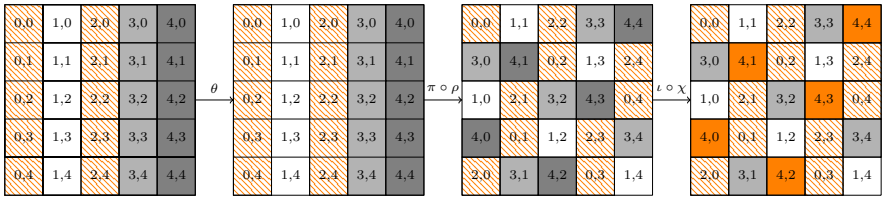


Figure 6: 1-round linear structure of Keccak-p with the degrees of freedom up to 512, with orange bits of degree at most 1, and gray, light gray and white bits being values 1, 0, and arbitrary unknown constants, respectively.

4 New MILP-Based Method for Finding Conditional Cubes

In this section, techniques are introduced for searching conditional cubes for Keccak-p based constructions, especially those with fully unknown internal state. Using these techniques, the internal state is recovered first and the key can be calculated from the internal state. Taking `KMAC` as an example, we first give a simple 1-round linear structure and list several observations upon which a new MILP model is introduced for finding (optimal) conditional cubes for Keccak-p based constructions (with fully unknown internal state).

4.1 1-Round linear structure

Suppose the internal state before processing messages is denoted by $k[x][y]$, $0 \leq x, y < 5$. For convenience, the r-bit message block is denoted as $a[x][y]$, $0 \leq x, y < 5$, where the last $\boldsymbol{c}$ bits are set to 0. Figure 7 provides a 1-round linear

structure of KMAC128 and shows the transformation of the internal state under the first round function R after absorbing the message block. Following the same notations as in Section 3.3, lanes with orange slashes denote variables, orange lanes have algebraic degree at most 1, and bits in white lanes are constants. Here, the first four lanes of the first and the third columns of $a[x][y]$ are set to be variables such that the sum $\bigoplus_{y=0}^{3} a[x, y]$ equals to certain constants for $x = 0, 2$. The capacity of KMAC128 consists of four lanes, so these lanes can not be chosen as variables. As can be seen from Figure 7, the output of the first round function is linear since there are no adjacent variables at the input of χ. This 1-round linear structure of KMAC128 in Figure 7 has a degree of freedom up to 384. A similar 1-round linear structure can also be constructed for KMAC256. These 1-round linear structures have large degrees of freedom, which are helpful for constructing conditional cubes upon them.

The major difference of this linear structure with those proposed in [13] is that, all the constants before χ of the first round are unknown due to unknown initial states. Hence, it is impossible to determine how the variables are propagated due to the logic AND, where ANDing with 1 allows propagation, and no propagation otherwise. This makes it hard to track the positions of all variables in the second round deterministically, hence increases the difficulty to find better (lesser conditions and larger cube dimensions) conditional cube variables fulfilling the condition that there is no multiplication (a.k.a. AND operation) with any other variables in the second round. This is the key difficulty raised from fully unknown states, and we are to solve it in our new MILP modeling in the next subsection.

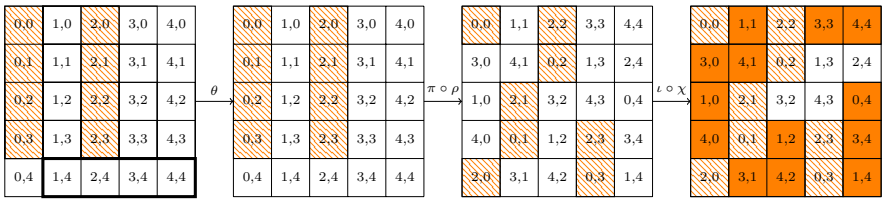


Figure 7: 1-round linear structure of KMAC128 with the degrees of freedom up to 384, with orange bits of degree at most 1, light gray and white bits being values 0, and arbitrary constants, respectively

From the 1-round linear structure, it is learnt that the algebraic degree of the internal state will remain as 1 without any condition. To construct a conditional cube, at least one variable should be selected such that it is not multiplied with any other variables in the second round, while there is no such restriction for the rest variables. Specifically, if an input bit of the χ in the second round contains the conditional variable, its two neighbouring bits should be constants. According to the property of Keccak-p (specifically the θ), each neighbouring bit is calculated from 11 output bits of the first round. These 11 bits may be

variables or constants, depending on the actual constant values involved in the χ of the first round. In the next subsection, this issue is settled by formalizing the diffusion effect of variables through χ.

Given a 2^n-dimensional conditional cube with one conditional cube variable and t bit conditions, it requires a time complexity of 2^{2^n+t} to recover t bits of the internal state for an $(n+1)$-round KECCAK-p based construction (with fully unknown internal state), hence the overall complexity to recover the internal state is around $\lceil \frac{|\boldsymbol{b}|}{t} \rceil \cdot 2^{2^n+t}$. Once the internal state is recovered, the key can be computed directly. It is inferred that the smaller t is, the lower the time complexity would be. So one aim of our new MILP model is to find conditional cubes with minimal bit conditions, meanwhile keeping the cube dimension large enough.

4.2 Modeling the non-linear layer χ

Mixed integer linear programming (MILP) is a general mathematical tool, which takes an objective function and a system of linear inequalities with respect to real numbers as input, and aims to search for an optimal solution which not only satisfies all the inequalities but also minimizes/maximizes the objective function.

The first observation before giving the MILP model is that, although one input bit to the first χ is calculated from 11 bits of the initial state, it is unnecessary for us to start from the initial state, as there is a bijective relation (the λ) between it and the state just before the χ. In the meanwhile, the 1-round linear structure could be started from the middle as well. Hence, instead of trying to derive everything from the very beginning, we start from the state just before χ. This simple yet crucial observation will reduce the complexity of the problem significantly, as will be seen later.

In order to describe our MILP model, more notations are needed here. Recall that the message block is denoted by a, and $b = \lambda(a)$, and k stands for the secret internal state. Let $k' = \lambda(k)$. Thus, $b \oplus k'$ is the input of the first χ and c indicates the output. The tuple (x, y, z) denotes the coordinates of one bit in the state. Additional notations A, B, C and V are used for modeling the search for cubes. Specifically, $A[x][y][z]$ ($B[x][y][z]$ or $C[x][y][z]$) is 1 if $a[x][y][z]$ (b[x][y][z] or c[x][y][z]) is a variable and 0 otherwise, while $V[x][y][z] = 1$ indicates a bit condition that $b[x][y][z]+k'[x][y][z]$ should be fixed. The number of bit conditions is denoted by t.

Note, we are to model two rounds of χ. Without losing any degree of freedom, we do it in two steps by modeling the first χ without imposing any additional condition, and the second χ using the output from our modeling of the first χ, *i.e.*, nested modeling. This may cost higher search complexity compared with previous works at first glance, we will see the effectiveness and power later. Due to the generality of our modeling, we could find optimal solutions whenever possible.

Modeling the first χ. Although χ is the only non-linear operation of KECCAK-p, modeling it into inequalities is non-trivial. Let us look at the computation

of one bit through χ. According to the algebraic expression of χ, $c[x][y][z] = b[x][y][z] + (1 + b[x+1][y][z]) \cdot b[x+2][y][z]$. For a conditional cube, the output bits of the first round should be linear, which can be guaranteed by the constraint that variables do not appear in adjacent input bits, namely $A[x][y][z] + A[x+1][y][z] \leq 1$. However, the value of input constants influence the diffusion of variables through χ and further influence the second round, as shown in Figure 6. However, as we find out, the diffusion patterns of variables through χ fall in a smaller than expected set as listed in Table 3, which makes the modeling of all cases possible without imposing any additional conditions.

Table 3: Diffusion of variables through χ. Symbol '*' denotes arbitrary value.

$B[x][y][z]$	$B[x+1][y][z]$	$B[x+2][y][z]$	$V[x+1][y][z]$	$V[x+2][y][z]$	$C[x][y][z]$
0	0	0	*	*	0
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	0	1	0
1	0	0	*	*	1
1	0	1	0	0	1
1	0	1	1	0	1

Table 4: Inequalities modeling the non-linear operation χ in the first round

$$
\begin{aligned}
B[x][y][z] - B[x+1][y][z] - B[x+2][y][z] - V[x+1][y][z] - V[x+2][y][z] - C[x][y][z] &\geq -2 \\
-B[x][y][z] - B[x+1][y][z] + V[x+2][y][z] + C[x][y][z] &\geq 0 \\
-B[x+2][y][z] - V[x+2][y][z] &\geq -1 \\
B[x][y][z] + B[x+1][y][z] + B[x+2][y][z] - C[x][y][z] &\geq 0 \\
-B[x][y][z] + C[x][y][z] &\geq 0 \\
-B[x+1][y][z] - B[x+2][y][z] + V[x+1][y][z] + V[x+2][y][z] + C[x][y][z] &\geq 0 \\
-B[x][y][z] - B[x+1][y][z] &\geq -1
\end{aligned}
$$

Now all patterns of the diffusion effect of χ are included in Table 3, and forms a finite set of discrete points in $\mathbb{R}^6$. To generate inequalities describing this set, as suggested by Sun *et al.* in [20], we first generate its convex hull. The convex hull of a set Q of discrete points in $\mathbb{R}^n$ is the smallest convex set that contains Q and can be described as a set of inequalities. The convex hull of a set in $\mathbb{R}^n$ can be generated by the *inequality_generator()* function in SageMath system. Usually, the number of inequalities returned by *inequality_generator()* is very large. However, a reduced set of inequalities can be selected using a greedy

algorithm from [20]. The reduced set of inequalities describing the diffusion effect of χ is given in Table 4.

Modeling the second χ. The conditional cube requires that conditional cube variables do not multiply with any other variables in the second round, which means their neighbouring bits S_i before the second χ should be constants. According to the round function R, each neighbouring bit S_i is calculated from 11 bits of $c[x][y][z]$. There are two cases depending on whether there is any variable among the 11 bits:

Case 1 For these 11 bits, none of them are variables, i.e., $C[x][y][z] = 0$;
Case 2 There are variables among the 11 bits and the XOR of these 11 bits form a linear equation which consumes 1 bit degree of freedom.

We introduce one more dummy variable e_i for S_i to indicate which case happens, where $e_i = 0$ for Case 1 and $e_i = 1$ for Case 2. Case 1 is simple, while for Case 2 one needs to pay attention to "uncertain propagations" or orange lanes in Figure 7 since no exact information can be derived from a linear equation containing variables with uncertain coefficients. So once Case 2 happens, additional conditions should be imposed to avoid uncertain propagation.

Similarly, all possible patterns of e_i and its related bits can be enumerated, see Table 5 for details and the set of inequalities are provided in Table 6. Specifically, if $c[x][y][z]$ is required in calculating S_i, the inequalities in Table 6 are added to the MILP model.

Table 5: Influence of conditional cube variables of the second χ. Symbol '*' denotes arbitrary value.

e_i	$B[x][y][z]$	$B[x+1][y][z]$	$B[x+2][y][z]$	$V[x+1][y][z]$	$V[x+2][y][z]$
0	*	*	*	*	*
1	0	0	0	*	*
1	1	0	0	*	*
1	1	0	1	1	0
1	0	0	1	1	0
1	0	1	0	0	1

4.3 Modeling the search for conditional cubes.

After introducing techniques for modeling χ, the following constraints are generated for searching conditional cubes.

Constraints for θ in the first round: Following the 1-round linear structure, the variables in each column of the message block, $a[x][y][z]$, $0 \leq y < 5$

Table 6: Inequalities modeling the non-linear operation χ in the second round

$$
\begin{aligned}
-e_i - B[x+1][y][z] - -B[x+2][y][z] &\geq -2 \\
-e_i + B[x][y][z] - B[x+1][y][z] + V[x+2][y][z] &\geq -1 \\
-e_i - B[x+2][y][z] + V[x+1][y][z] &\geq -1 \\
-e_i - B[x+1][y][z] - V[x+1][y][z] &\geq -2 \\
-e_i - B[x+2][y][z] - V[x+2][y][z] &\geq -2 \\
-e_i - B[x][y][z] - B[x+1][y][z] &\geq -2
\end{aligned}
$$

sum to constants such that θ acts like identity w.r.t. the variables. That is, $A[x][y][z] = B[x][y][z]$ in our search program, $0 \leq x, y < 5, 0 \leq z < 64$. As similarly done in [15], a dummy variable $D[x][z]$ is introduced for each column, and inequalities for each column are as follows.

$$
\begin{aligned}
D[x][z] \geq A[x][0][z], D[x][z] \geq A[x][1][z], \\
D[x][z] \geq A[x][2][z], D[x][z] \geq A[x][3][z], D[x][z] \geq A[x][4][z], \\
A[x][0][z] + A[x][1][z] + A[x][2][z] + A[x][3][z] + A[x][4][z] \geq 2 \cdot D[x][z].
\end{aligned} \tag{1}
$$

Constrains for χ of the first round:

1. If $B[x][y][z]$ indicates a *conditional* cube variable, then the neighbouring bits should be fixed constants such that it do not diffuse to other positions. It requires

$$
\begin{aligned}
B[x-1][y][z] = 0, B[x+1][y][z] = 0, \\
V[x-1][y][z] = 1, V[x+1][y][z] = 1.
\end{aligned} \tag{2}
$$

   Additionally, $B[x-2][y][z] = 0$ should be satisfied to avoid the conditional variable from diffusing to $C[x-2][y][z]$.
2. If $B[x][y][z]$ does not indicate a conditional cube variable, then the inequalities in Table 4 should hold. Note that the inequalities in Table 4 also exclude the cases where variables appear in two adjacent positions.

Constraints for χ of the second round: In the second round, only constraints for conditional cube variables are needed. Given the positions of conditional cube variables before the χ of the second round, a set of $c[x][y][z]$ can be determined for calculating each neighbouring bit S_i through the linear layer λ. Suppose T is the set of $c[x][y][z]$ for calculating all neighbouring bits S_i. Then impose inequalities in Table 5 to $(e_i, B[x][y][z], B[x+1][y][z], B[x+2][y][z], V[x+1][y][z], V[x+2][y][z])$ once $c[x][y][z] \in T$:

$$
\text{For } \; c[x][y][z] \in T, \text{apply inequalities in Table 5.} \tag{3}
$$

Constraint for the dimension: If a 2^n-dimensional conditional cube is desired, then set

$$\sum A[x][y][z] - \sum D[x][z] - \sum e_i = 2^n, \tag{4}$$

where $\sum D[x][z] + \sum e_i$ is the number of consumed degrees of freedom.

Objective: The objective is to minimize bit conditions. That is

$$\text{Minimize}: \quad \sum V[x][y][z]. \tag{5}$$

Besides, there may exist additional constraint. For example, the last c bits and some padded bits cannot be variables. When all constraints are generated, an MILP solver is invoked to find a solution that minimizes the objective.

Extracting the bit conditions from the solution. In the model, $V[x][y][z] = 1$ indicates a bit condition. However, whether the input bit $k'[x][y][z] + b[x][y][z]$ should be 0 or 1 is not explicitly displayed. To this, we use Algorithm 1 to determine the constants. In fact, inequalities in Table 4 confine the solution of $(B[x][y][z], B[x+1][y][z], B[x+2][y][z], V[x+1][y][z], V[x+2][y][z], C[x][y][z])$ to the patterns in Table 3. There are three patterns where $V[x+1][y][z]$ or $V[x+2][y][z]$ is 1, but the third one is a shifted version of the second one. Therefore, we just need to determine whether it follows the first one or not, as in Algorithm 1. If it follows the first pattern, the constant is 1, otherwise, the constant is 0.

Algorithm 1: Extracting the bit conditions from the solution.

Input: A solution where all bits of B, V and C are assigned.
Output: Bit conditions
con $= \emptyset$;
for *All* $V[x][y][z]$ **do**
  if $V[x][y][z] = 1$ **then**
    $i \leftarrow (B[x-1][y][z], B[x][y][z], B[x+1][y][z], 1, V[x+1][y][z], C[x-1][y][z])$;
    if $i = (0, 0, 1, 1, 0, 0)$ **then**
      con$\leftarrow b[x][y][z] + k[x][y][z] + 1$;
    else
      con$\leftarrow b[x][y][z] + k[x][y][z]$;
    end
  end
end
return con;

4.4 Comparison with the existing MILP model

Very recently, Li *et al.* proposed an MILP model for searching conditional cubes [15] which sets every $b[x][y][z]$ to a constant if it relates to the conditional variable.

In our model, we incorporate the full diffusion effect of χ and hence consider a broader class of conditional cubes. In particular, $b[x][y][z]$ can be a variable even if it relates to the conditional variable. As a result, more conditional cubes can be found with a greater range of dimension. As demonstrated in Table 7, better conditional cubes are found using our model. In particular, given the dimension, our model returns conditional cubes with much less bit conditions. For example, the 32-dimensional conditional cube for Keccak-MAC-512 in [15] requires 24 bit conditions involving the key, while using our model, the number of bit conditions can be only 4 ($n = 5$ and $t = 4$), which reduces the time complexity of attacking 6-round Keccak-MAC-512 from $2^{58.3}$ [15] to $\lceil \frac{|k|}{t} \rceil \cdot 2^{2^n+t} = \lceil \frac{128}{4} \rceil \cdot 2^{2^5+4} = 2^{41}$. The largest cube of Keccak-MAC-512 found by our method has dimension 54, which is provided in Table 10.

Table 7: Comparison with the previous MILP model on Keccak-MAC with the conditional cube placed at $(2, 0, 0)$ and $(2, 1, 0)$. The number of bit conditions only takes those involving key bits into account.

Variant	Dimension	#Conditions	Reference
Keccak-MAC-384	65	8	[15]
	97	8	This
	65	**2**	
Keccak-MAC-512	32	24	[15]
	47	24	This
	32	**4**	
	54	42	

5 Application to KMAC

In this section, techniques described in Section 4 are used to find conditional cubes for KMAC, based on which key recovery attacks can be mounted on 7-round KMAC128 and 9-round KMAC256 respectively.

5.1 Cube attack on KMAC128

For KMAC128, the capacity is 256, which covers only four lanes. By setting two bits in $a[x][y][z]$, $0 \leq y < 4$ as the conditional cube variables[4], our MILP model could find large conditional cubes with 4 bit conditions which are least possible conditions. To make the attack clear, a toy cube of KMAC is introduced first, as

[4] There is an exception that no conditional cube can be found for conditional variables chosen from lanes $(1, 0), (1, 1)$. The reason is that Constraint 3 involves the conditional cube variables, leading the model infeasible.

shown in Table 8. This cube has dimension 16, and $a[0][0][0], a[0][1][0]$ are chosen to be the conditional cube variable. The 4 bit conditions can be derived directly from the positions of the conditional cube variable since only the conditional cube variable contributes to bit conditions in this case. Otherwise, Algorithm 1 is used to deduce bit conditions. Note that, $b = \lambda(a)$ and the relation between $a[x][y][z]$ and $b[x][y][z]$ is not expressed explicitly in the bit conditions. The rest 15 ordinary cube variables can be extracted from $A[x][y][z]$, $0 \le x, y < 5, 0 \le z < 64$ which are represented as a 5×5 array of lanes and labeled as 'Positions of cube variables' in the table. In the remainder of the paper, the bit conditions are omitted if they come only from the conditional cube variable.

Table 8: A toy cube of `KMAC`. Positions of cube variables are derived from a 5×5 array of lanes in hexadecimal using little-endian format where '0' is replaced with '-'.

Positions of cube variables
`4--8--168D-2---1|----------------|--8-----------1|----------------|----------------`
`66-8--16C3-28-19|----------------|1-C-8----1-----1|----------------|----------------`
`26----1-4F4-8-18|----------------|1-4-8----1------|----------------|----------------`
`24-----2--4---1-|----------------|----------------|----------------|----------------`
`----------------|----------------|----------------|----------------|----------------`
The conditional cube variable: $a[0][0][0] = a[0][1][0] = v_0$
Ordinary cube variables
$a[0][1][4] = v_1$, $a[0][2][4] = v_2$, $a[0][3][4] = v_1 + v_2$, $a[0][1][15] = a[0][2][15] = v_3$, $a[0][0][17] = a[0][1][17] = v_4$, $a[0][2][22] = a[0][3][22] = v_5$, $a[0][1][24] = a[0][2][24] = v_6$, $a[0][1][30] = a[0][2][30] = v_7$, $a[0][1][57] = a[0][2][57] = v_8$, $a[0][1][58] = v_9$, $a[0][2][58] = v_{10}$, $a[0][3][58] = v_9 + v_{10}$, $a[0][1][61] = v_{11}$, $a[0][2][61] = v_{12}$, $a[0][3][61] = v_{11} + v_{12}$, $a[0][0][62] = a[0][1][62] = v_{13}$, $a[2][0][0] = a[2][1][0] = v_{14}$, $a[2][1][24] = a[2][2][24] = v_{15}$.
Conditions
$b[0][3][36] = k'[0][3][36] + 1$, $b[2][3][36] = k'[2][3][36]$, $b[4][0][0] = k'[4][0][0] + 1$, $b[1][0][0] = k'[1][0][0]$.

For `KMAC128`, 64-dimensional conditional cubes are enough for attacking 7 rounds of `KMAC128`. In the following, multiple 64-dimensional conditional cubes are used for the recovery of the internal state.

1. **Recover t bits of the internal state.** Given a 64-dimensional conditional cube with t bit conditions where $t = 4$ for `KMAC128`, the t bits of the secret internal state $k'[x][y][z]$ involving in the conditions are guessed and then the constant part of the messages is chosen such that the t bit conditions are satisfied. The right guess is detected by assigning all possible values to each cube variable and checking the sum of all outputs under the guess. If the cube sum is zero, then the corresponding guess is the right one with overwhelming probability and the t bits of the secret internal state are recovered. The time complexity for recovering the t bits of the internal state is $2^{64+t} = 2^{68}$.

2. **Recover t lanes of the internal state.** Due to the z-axis translation invariance of Keccak-f, a conditional cube is still a conditional cube after being rotated along the z-axis. A cube and all its rotations are z-axis equivalent. However, for `KMAC` the padding rule may break the z-axis equivalence. To avoid it from happening, the last lane of the r-bit message block is set to be inactive. Therefore, by rotating the cube bit by bit, t lanes of the internal state would be recovered in $2^6 \cdot 2^{68} = 2^{74}$ calls of 7-round `KMAC128`.
3. **Recover the whole internal state.** Ten z-axis equivalent conditional cubes are used to recover the full internal state. The details of these cubes are given in Table 11 and 12, and the order of the lanes recovered are displayed in Figure 8. The total time complexity of recovering the whole internal state is $2^6 \cdot 2^{64}(1 \cdot 2^4 + 3 \cdot 2^3 + 6 \cdot 2^2) = 2^{76}$.

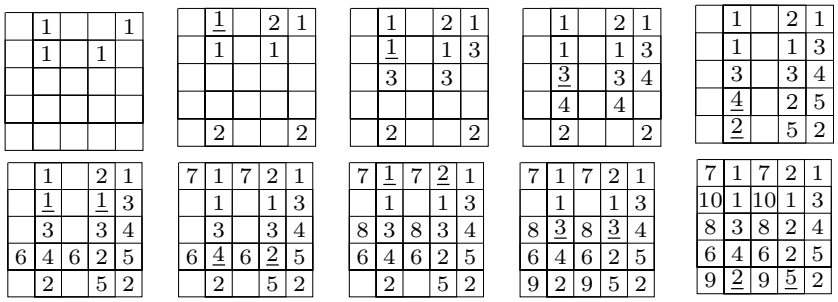


Figure 8: The lanes recovered using ten z-axis equivalent conditional cubes. The underline means bits of these lanes are involved in conditions but they are already known.

5.2 Cube attack on `KMAC256`

`KMAC256` has a capacity of 512 which is equivalent to 8 lanes. Including the last lane of the message block where certain bits are padded, there are 9 lanes which can not contain variables. Apart from this, the cube search for `KMAC256` remains as that for `KMAC128`. Our MILP model could find many 128-dimensional conditional cubes which can be used to attack 8 rounds of `KMAC256`. Since the output length of `KMAC256` can be more than 320 bits, the first 5 lanes of the output can be reversed through the χ of the last round. This immediately increases the attacked rounds by one, as this inversion covers the χ of the last round, while λ does not increase algebraic degree. As a result, 9 rounds of `KMAC256` can be attacked.

Choice of the conditional cube variable. By setting two bits in $a[x][y][z]$, $0 \leq y < 3$ as the conditional cube variables, the obtained cubes have more than 30 bit conditions. The increase of bit conditions is caused by the increase of

capacity. In order to reduce the number of bit conditions, we place the conditional cube variable in a 2-round column parity kernel (CP-kernel), and thus it does not diffuse even in the second round, leading to a small Constraint 3. As studied in [8], the minimal Hamming weight of a 2-round CP-kernel differential trail of Keccak-f[1600] is 6. Among all the 2-round CP-kernel differential trails, only those which have no difference in the last 9 lanes can be applied to the conditional cube search of `KMAC256`. Fortunately, there is one (only one) 2-round CP-kernel differential trail satisfying this requirement. The active bit positions of the 2-round CP-kernel differential trail are

$$[(0,0,0),(0,1,0),(1,0,63),(1,2,63),(2,1,30),(2,2,30)].$$

By setting the conditional cube variable to these six bit positions, our MILP model returns 128-dimensional cubes with 12 bit conditions, with which 11 lanes (one lane overlapped) of the internal state can be recovered. With these 11 lanes known, cubes with the conditional cube variable placed in a column of $a[x][y][z]$, $0 \le y < 3$ can then be exploited to recover the rest lanes.

To recover the whole internal state, three z-axis equivalent conditional cubes as shown in Table 13 are used and lanes recovered in each cube are displayed in Figure 9. As can be learned from the figure, the time complexity of the internal state recovery is $2^6 \cdot 2^{128}(2^{12} + 2^{11} + 2^3) \approx 2^{147}$ calls of 9-round `KMAC256`.

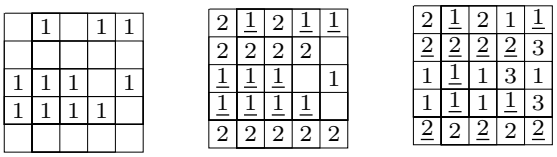


Figure 9: The lanes recovered using three z-axis equivalent conditional cubes. The underline means bits of these lanes are involved in conditions but they are already known.

5.3 Experimental verification

Since the attacks on both variants of `KMAC` are impractical with current computation power, the correctness of the attacks is verified on conditional cubes with small dimensions. We do no change to the attacks except reducing the number of rounds for the cube tester in the middle, so the attack complexity reduces to a practical level. We implement two conditional cube attacks[5]: one based the 16-dimensional toy cube in Table 8 for fast verification, and the other based on a 32-dimensional cube for attacking 7-round `KMAC256` (or 6-round `KMAC128`). The correctness of our attacks are confirmed by both experiments.

[5] The source codes will available online soon.

6 Other Applications

In this section, our new model are applied to other Keccak-p based constructions, including Kravatte, Keyak and Ketje.

6.1 Application to Kravatte

Cube attacks on Kravatte can be done by changing one of the input message blocks while keeping the rest the same. In Kravatte, $\boldsymbol{b}$-bit message blocks are XORed with a $\boldsymbol{b}$-bit mask before being processed by the underlying permutation. The $\boldsymbol{b}$-bit masks, derived from the key, are fully unknown. Considering the mask as the internal state, the attacks in Section 5 can be directly applied to Kravatte. That is, key recovery attacks can be achieved on 7/8-round Kravatte with 128/256-bit keys. It is interesting to note, although there is an additional layer of mask added as post-whitening key, it does not change the effectiveness of the attack since this post-whitening key can be viewed as an unknown constant added to the last round and it does not affect the algebraic degree of the cube tester.

Moreover, Kravatte imposes less constraints on cube variables since full-state message blocks are used. Even though the message blocks are XORed with the mask before applying θ of the first round, we could choose the values of the message block before χ, that is $b[x][y][z]$. In this way, the diffusion effect of θ in the first round is skipped. By setting two bits in $b[x][y][z]$, $0 \leq y < 5$ as the conditional cube variable, sufficiently many 256/512-dimensional cubes are obtained with four bit conditions. Examples of 256/512-dimensional conditional cubes is given in Table 9.

Inversion the last round. First, we consider the case that the output contains as lest one row of lanes of the state but not the whole state. If the last round can be reversed, then one more round can be attacked with the same cube, as the attacks of `KMAC256` in Section 5. To make an inversion of the last round possible, the post-whitening key is needed. However, to detect the right guess of the internal state bits involved in the conditions, an inversion of a few Sboxes are enough. Suppose the post whitening key related to u Sboxes are guessed. Namely, $5u$ bits are guessed. For each output of the cube, an additional inversion of the u Sboxes is processed under each value of the $5u$ post-whitening key bits and check the cube sum on the $5u$-bit inputs to the u Sboxes. If there are t bit conditions, then $5u > t$ should hold to make the cube sum distinguishable. Briefly, the t-bit internal state and the $5u$-bit post-whitening key are recovered as follows.

1. Recover the first t bits of the internal state.
   (a) Guess the t-bit internal state;
      i. Construct a conditional cube such that the t bit conditions are satisfied.
      ii. Get the outputs of this cube and keep u sets of 5-bit outputs related to the u Sboxes such that each element in the sets appears odd times in the outputs (discard those appear even times).

iii. For each of the u Sboxes, guess the corresponding 5-bit post-whitening key and check the cube sum of the inputs of the Sbox. Save the guess such that the cube sum is zero. On average, there is one candidate left for each Sbox.

(b) For each t-bit guess, there remains one $5u$-bit post-whitening key on average. Using additional 2^t more conditional cubes by changing the constant part, the unique value for the $t + 5u$ bits can be recovered.

2. Recover the rest $b - t$ bits of the internal state.
   - The $5u$-bit post-whitening key is known and now the u Sboxes of the last round can be reversed. which make it possible to recover the rest $b - t$ bits of the internal state without guessing additional bit of the post-whitening key.

In Step (iii), each set contains at most 32 elements and is processed independently, so the time complexity of this step is less than $u \cdot 2^5 \cdot 2^5 = u \cdot 2^{10}$ and can be neglected. In short, the recovery of the $(t+5u)$ bits in Step 1 and 2 take a time complexity of $(2^t + 2^t) \cdot 2^{2^n}$ and $2^{2^n+t} \cdot 2^6 \cdot 10$ respectively for a 2^{2^n}-dimensional conditional cube, where $2^{6+t} \cdot 10 < 2^{14}$ is the estimated factor of recovering the internal state using conditional cubes with $t = 4$ bit conditions, as the attack of `KMAC128` in Section 5.1.

Here, details of the attack are omitted for Kravatte. Since 64/128/256/512-dimensional conditional cubes can be found with 4 bit conditions, we set $t = 4$ and choose $u = 2$. Thus, the time complexity of the attack is about 2^{2^n+14}. As a result, 8/9/10-round Kravatte can be attacked with time complexities 2^{78}, 2^{142} and 2^{270} if the key size is greater than 78, 142 bits and 270 bits, respectively. As an extreme case, when $n = 9$, there is an attack against 11-round Kravatte of complexity 2^{526}. This is a valid attack when the scheme uses a key of length more than 526 bits, and claims a security level higher than 526 as well. However, the recommended key length is no more than 320 bits, so the 11-round attack is of theoretical use.

Second, if the whole state is taken as the output, a simpler attack of Kravatte can be obtained by recovering the post-whitening key. In this case, an ordinary cube with dimension $2^n + 1$ is used. Similar to Step (ii) and (iii), the post-whitening key can be recovered Sbox by Sbox. The time complexity will be dominated by the computation of the cube, i.e., 2^{2^n+1}. So the time complexities of attacking 8/9/10-round Kravatte could be 2^{65}, 2^{129} and 2^{257} if the key size is greater than 65, 129 bits and 257 bits, respectively.

6.2 Application to Keyak and Ketje

This section considers conditional cube attacks of Keyak and Ketje under the nonce respect setting, i.e., cube variables are chosen from the positions where the nonce is loaded.

Figure 10 shows the key pack of Keyak and Ketje respectively (for Ketje, it shows the key pack after π^{-1}), where blue positions stand for the key, light

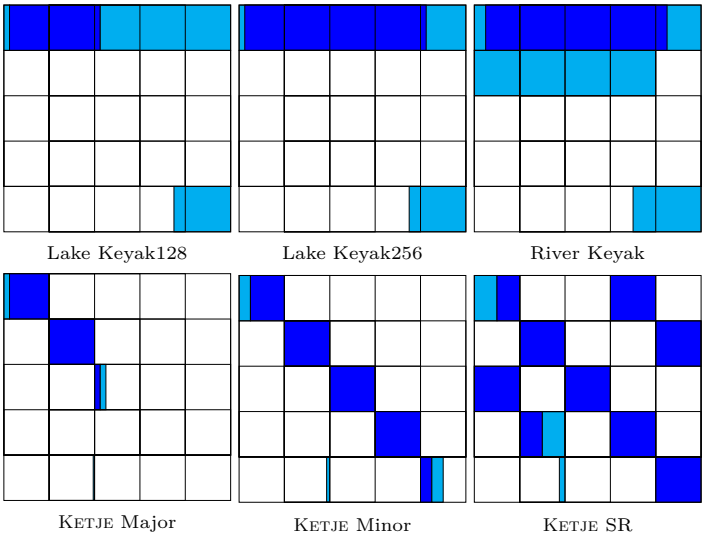


Figure 10: Key pack of Keyak and Ketje where blue means the key, light blue denotes padded or encoded bits and white lanes are the nonce.

blue positions denote padded or encoded bits and white positions are the nonce. This means that cube variables should be chosen from white lanes.

All instances of Keyak and Ketje considered in this paper use 128-bit keys, except Lake Keyak, where 256-bit keys are supported by replacing Keccak-p $[1600, 12]$ with Keccak-$p[1600, 14]$. Our main results are as follows and summarized in Table 2.

Lake Keyak128 Using a 64-dimensional cube with 2 bit conditions involving the key (see Table 14), the key recovery attack of 8-round Lake Keyak128 costs a data and time complexities $2^2 \cdot 2^{64} \cdot 32 + 2^{64} = 2^{71.01}$ where the last χ can be partially reversed due to large output length.

Lake Keyak256 Using a 128-dimensional cube with 4 bit conditions involving the key (see Table 15), the key recovery attack of 9-round Lake Keyak256 costs a data and time complexities less than $2^4 \cdot 2^{128} + 2^3 \cdot 2^{128} \cdot 63 + 2^{128} = 2^{137.05}$.

River Keyak Using a 64-dimensional cube with 12 bit conditions involving the key (see Table 16, these 12 bit conditions involve 11 bits key information), the key recovery attack of 8-round River Keyak costs a data and time complexities $2^{11} \cdot 2^{64} + 2^{10} \cdot 2^{64} \cdot 6 + 2^{128-71} = 2^{77}$.

Ketje **Major** Using a 64-dimensional cube with 3 bit conditions involving the key (see Table 17), the key recovery attack of 7-round Ketje Major costs a data and time complexities $2^3 \cdot 2^{64} \cdot 3 + 2^2 \cdot 2^{64} \cdot 2 + 2^1 \cdot 2^{64} \cdot (64-5) + 2^{64} = 2^{71.24}$.

Ketje **Minor** Using a 64-dimensional cube with 4 bit conditions involving the key (see Table 18), the key recovery attack of 7-round Ketje Minor costs a data and time complexities less than $2^4 \cdot 2^{64} + 2^3 \cdot 2^{64} \cdot 63 + 2^{64} = 2^{73.03}$.

For KETJE SR and KETJE JR, our model could not find attacks which are better than the existing ones in [12]. However, for KETJE SR with KECCAK-p as the underlying permutation, namely, KETJE SR v1, better attacks on 7-round KETJE SR are found using a 64-dimensional cube with 27 bit conditions (see Table 19, all bit conditions involve the key) and the time and data complexities are $2^{27} \cdot 2^{64} \cdot 2 + 2^{128-54} = 2^{92}$. Therefore, KETJE instances using KECCAK-$p^\star$ are stronger than those instances using KECCAK-p under our attacks.

7 Conclusions

In the paper, we proposed a new MILP model for searching conditional cubes for KECCAK-p based keyed constructions. Particularly, we incorporated the diffusion effect of variables through the non-linear layer and took a broader class of conditional cubes into account. With the new model, conditional cubes with desired dimensions and least bit conditions were found for two KECCAK-p based constructions with unknown internal state, `KMAC` and KRAVATTE. As a result, key recovery attacks of 7-round `KMAC128`, 9-round `KMAC256` and up to 10-round KRAVATTE can be mounted respectively. To the best of our knowledge, these are the first cryptanalysis results against `KMAC` and KRAVATTE. The application of our model to KEYAK and KETJE gives rise to new attacks or better attacks with reduced complexities.

Open discussion. The conditional cube attack on KRAVATTE is also applicable to full-state keyed sponge [16] or duplex [7], both of which take in $\boldsymbol{b}$-bit instead of $\boldsymbol{r}$-bit message blocks. In addition, each time the duplex absorbs a message block, it leaks $\boldsymbol{r}$ bits of the internal state as output, which decreases the time complexity slightly. For existing KECCAK-p based full-state keyed constructions, conditional cubes of relatively large dimensions can be found. Regarding this, conditional cubes that fully linearize the first two rounds might exist and can be used to improve attacks on KECCAK-p based full-state keyed constructions with short key length, e.g., 128 bits. However, due to the difficulties in controlling all cube variables in the second round, finding such conditional cubes remains an open problem.

References

1. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Farfalle: Parallel Permutation-Based Cryptography. Cryptology ePrint Archive, Report 2016/1188 (2016), https://eprint.iacr.org/2016/1188
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic Sponge functions. Submission to NIST (Round 3) (2011), http://sponge.noekeon.org/CSF-0.1.pdf
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography - 18th International Workshop,

SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers. LNCS, vol. 7118, pp. 320–337. Springer (2011), https://doi.org/10.1007/978-3-642-28496-0_19
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak Reference. http://keccak.noekeon.org (January 2011), version 3.0
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR submission: Ketje v2. Candidate of CAESAR Competition (September 2016)
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR submission: Keyak v2. Candidate of CAESAR Competition (September 2016)
7. Daemen, J., Mennink, B., Van Assche, G.: Full-State Keyed Duplex With Built-In Multi-User Support. to appear in ASIACRYPT 2017, available at https://eprint.iacr.org/2017/498 (2017)
8. Daemen, J., Van Assche, G.: Differential propagation analysis of keccak. In: Canteaut, A. (ed.) Fast Software Encryption: 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers. pp. 422–441. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
9. Dinur, I., Dunkelman, O., Shamir, A.: Improved practical attacks on round-reduced keccak. Journal of Cryptology 27(2), 183–209 (2014)
10. 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: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. LNCS, vol. 9056, pp. 733–761. Springer (2015)
11. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. LNCS, vol. 5479, pp. 278–299. Springer (2009), https://doi.org/10.1007/978-3-642-01001-9_16
12. 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), https://doi.org/10.13154/tosc.v2017.i1.259-280
13. Guo, J., Liu, M., Song, L.: Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. LNCS, vol. 10031, pp. 249–274 (2016)
14. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional Cube Attack on Reduced-Round Keccak Sponge Function. In: Coron, J., Nielsen, J.B. (eds.) 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. LNCS, vol. 10211, pp. 259–288 (2017), https://doi.org/10.1007/978-3-319-56614-6_9
15. Li, Z., Bi, W., Dong, X., Wang, X.: Improved Conditional Cube Attacks on Keccak Keyed Modes with MILP Method. to appear in ASIACRYPT 2017, available at https://eprint.iacr.org/2017/804 (2017)
16. Mennink, B., Reyhanitabar, R., Vizár, D.: Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption. In: Iwata, T., Cheon, J.H. (eds.) 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. LNCS, vol. 9453, pp. 465–489. Springer (2015), https://doi.org/10.1007/978-3-662-48800-3_19

17. NIST: SHA-3 COMPETITION. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html (2007-2012)
18. Qiao, K., Song, L., Liu, M., Guo, J.: New Collision Attacks on Round-Reduced Keccak. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017, Paris, France, April 30 - May 4, 2017, Proceedings, Part III. LNCS, vol. 10212, pp. 216–243 (2017)
19. Song, L., Liao, G., Guo, J.: Non-full Sbox Linearization: Applications to Collision Attacks on Round-Reduced Keccak. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II. LNCS, vol. 10402, pp. 428–451. Springer (2017), https://doi.org/10.1007/978-3-319-63715-0_15
20. 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: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. LNCS, vol. 8873, pp. 158–178. Springer (2014), https://doi.org/10.1007/978-3-662-45611-8_9
21. The U.S. National Institute of Standards and Technology: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions . Federal Information Processing Standard, FIPS 202 (5th August 2015), http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
22. The U.S. National Institute of Standards and Technology: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash. NIST Special Publication 800-185 (21st December 2016), http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf

A Appendix: Conditional Cubes of KMAC, Kravatte, Keccak-MAC, Keyak and Ketje

Table 9: Two conditional cubes of KRAVATTE with dimension 256 and 512 respectively

256	**Positions of cube variables**
	-81-81-1-8---841 E2A-1--8-2-A---- 18-C-58-----22-- -25-8--12----82- --8254-2-51141-2
	D8-154--2-24---1 ---8-322C288--9- 8--58-5-2-2-724- 4--8-3---------1 2--2--7--41221-8
	-------8--2-3--- 582-8-5183-8-461 --1414--1----A-- 31-----18-2----- 8-92-E4-141--151
	-4-----12-8----- -1811-1--A-82-4F -C24-1-2----42-- -1--18412-82--48 1-832-8-8C3-81-1
	82484-28--416-24 --8--1124-2----1 32-55-21--C2C89- -948--42--3-2-24 1--1--8---828-81
	Position of the conditional cube variable
	(0,0,0), (0,1,0)

512	**Positions of cube variables**
	38A885217146-889 865-385682896-42 69AC418-384-1634 -45-A679C184A8C8 83-31-86-C115112
	F1--24-2----88-1 --51-3B4E3-A24F2 9824D84318E54A-- 6248-21-8---8-41 -49359ED4C77752A
	9449-1-C8-2-2-14 4AA-EC833B4B9CEA 211D12444---2215 56A-4113A9254428 2112BEC-545289C3
	-5--4-C93-42-8-2 7--1-612CAA9A479 88F4912-----1A84 25-142-76-438478 DAFA382-8DB471-5
	C44-9-2911E-4-F4 1B11255-EA-294-B E48448A1--C122-4 -37B111891365-7- 18--4642-8-1A4-B
	Position of the conditional cube variable
	(0,0,0), (0,1,0)

Table 10: 54-dimensional conditional cube for KECCAK-MAC 512. There are 58 bit conditions (42 of them involve the key) which can be exacted from B, V and C using Algorithm 1. The second round consumes 5 degrees of freedom.

B				
----------	----------	----------	----------	----------
642937249274-65D	----------	----------	----------	----------
----------	6C1F8D14DE3646D8	----------	----------	----------
----------	----------	----------	----------	----------
D86C1F8D14DE3646	2493A-32EB2149B9	----------	----------	----------
V				
----------	----------	----------	----------	----------
----------	2-2-35--1-------	----------	----------	4---2--------2-1
2--4-51-18---24-	----------	-----1---8----C-	----------	----------
----------	----------	----------	----------	----------
2-8-2-12-8-1----	4-6415--1-8-3---	----2--229-1-9A-	----------	4-4----4---2-2-2
C				
----------	----------	----------	----------	----------
642937249274-65D	----------	----------	242917249274-45C	44-9-2248274-65D
6C1F8C14D6364618	6C1F8D14DE3646D8	----------	----------	4C1B88-4C6364498
----------	----------	----------	----------	----------
FCFF9FBDD6FE765F	2493A-32EB2149B9	----------	982C1F8914DC3444	9C1B8AADE77E4FFF

Cube variables

$a[2][0][0] = a[2][1][0] = v0, a[2][0][1] = a[2][1][1] = v1, a[2][0][3] = a[2][1][3] = v2,$
$a[2][0][4] = a[2][1][4] = v3, a[2][0][8] = a[2][1][8] = v4, a[2][0][11] = a[2][1][11] = v5,$
$a[2][0][12] = a[2][1][12] = v6, a[2][0][14] = a[2][1][14] = v7, a[2][0][15] = a[2][1][15] = v8,$
$a[2][0][19] = a[2][1][19] = v9, a[2][0][20] = a[2][1][20] = v10, a[2][0][21] = a[2][1][21] = v11,$
$a[2][0][22] = a[2][1][22] = v12, a[2][0][24] = a[2][1][24] = v13, a[2][0][25] = a[2][1][25] = v14,$
$a[2][0][28] = a[2][1][28] = v15, a[2][0][30] = a[2][1][30] = v16, a[2][0][34] = a[2][1][34] = v17,$
$a[2][0][36] = a[2][1][36] = v18, a[2][0][37] = a[2][1][37] = v19, a[2][0][41] = a[2][1][41] = v20,$
$a[2][0][42] = a[2][1][42] = v21, a[2][0][43] = a[2][1][43] = v22, a[2][0][44] = a[2][1][44] = v23,$
$a[2][0][45] = a[2][1][45] = v24, a[2][0][46] = a[2][1][46] = v25, a[2][0][52] = a[2][1][52] = v26,$
$a[2][0][53] = a[2][1][53] = v27, a[2][0][55] = a[2][1][55] = v28, a[2][0][56] = a[2][1][56] = v29,$
$a[2][0][61] = a[2][1][61] = v30, a[2][0][62] = a[2][1][62] = v31, a[3][0][0] = a[3][1][0] = v32,$
$a[3][0][3] = a[3][1][3] = v33, a[3][0][6] = a[3][1][6] = v34, a[3][0][9] = a[3][1][9] = v35,$
$a[3][0][12] = a[3][1][12] = v36, a[3][0][13] = a[3][1][13] = v37, a[3][0][14] = a[3][1][14] = v38,$
$a[3][0][16] = a[3][1][16] = v39, a[3][0][17] = a[3][1][17] = v40, a[3][0][20] = a[3][1][20] = v41,$
$a[3][0][23] = a[3][1][23] = v42, a[3][0][25] = a[3][1][25] = v43, a[3][0][30] = a[3][1][30] = v44,$
$a[3][0][33] = a[3][1][33] = v45, a[3][0][34] = a[3][1][34] = v46, a[3][0][36] = a[3][1][36] = v47,$
$a[3][0][38] = a[3][1][38] = v48, a[3][0][39] = a[3][1][39] = v49, a[3][0][40] = a[3][1][40] = v50,$
$a[3][0][42] = a[3][1][42] = v511, a[3][0][45] = a[3][1][45] = v52, a[3][0][46] = a[3][1][46] = v53,$
$a[3][0][54] = a[3][1][54] = v54, a[3][0][56] = a[3][1][56] = v55, a[3][0][57] = a[3][1][57] = v56,$
$a[3][0][58] = a[3][1][58] = v57, a[3][0][61] = a[3][1][61] = v58,$
$v3 + v48 = 0, v28 + v43 = 0, v19 + v39 = 0,$
$v25 + v28 + v58 = 0, v13 + v57 = 0$

Table 11: Cube variables for attacking 7-round KMAC128 (1)

1	**Positions of cube variables**
	3-4--2B-1CC-1--1 ---------- C-8-84-147-4C18- ---------- ----------
	3442-2881-4----8 ---------- 3-8-8--24D1-8482 ---------- ----------
	A4-5--312CD-1--9 ---------- B---84-3-9-4-5-- ---------- ----------
	A447-2992-1-1--- ---------- 6-8------B1-C5-2 ---------- ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(0,0,0), (0,2,0)

2	Positions of cube variables
	---------- ---------- -844288158-94125 ---------- C4428A1--2-1443A
	---------- ---------- 2C-42--111-8-12C ---------- 84848-1---82542-
	---------- ---------- 2454--814944-1-1 ---------- 44C6-A1--24-1-1A
	---------- ---------- --5--88-18454--8 ---------- C--4-2---2C3-428
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(2,0,0), (2,2,0)

3	**Positions of cube variables**
	6F-5B-8C-28-6--8 ---------- ---------- -844---7D1-8-4-1 ----------
	61-438---2E838-A ---------- ---------- -1-1--24848----8 ----------
	-F-128-C--4-79-A ---------- ---------- -945--37458444-9 ----------
	6E-18-8---28-1-- ---------- ---------- -8----3-5--C4--8 ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(3,0,0), (3,2,0)

4	**Positions of cube variables**
	---------- 4--8-8177-3--A-1 ---------- 11-34C82-1-422EC ----------
	---------- ---8---671-2---- ---------- -8-24D82--1424C- ----------
	---------- 4-88-81A6132-A-1 ---------- 98-3488---1--AEC ----------
	---------- --88-8-93--2---- ---------- 89---9---1--2E-C ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(1,0,0), (1,2,0)

5	**Positions of cube variables**
	---------- 8E168-61-1--1-A- ---------- ---------- -8--2-31--2243-1
	---------- 8F-68--1---2B-3- ---------- ---------- --1--1B-4-9666-2
	---------- -8168-61-5--2-31 ---------- ---------- 8C1--1814-B667-3
	---------- -71------4-28-B1 ---------- ---------- 8C1-212-4-------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(4,0,0), (4,2,0)

Table 12: Cube variables for attacking 7-round KMAC128 (2)

6	**Positions of cube variables**
	3--2---44D-22--A ---------- 8488-4---8-28-28 ---------- ----------
	24-2---444-22--B ---------- 3-868-24-9368223 ---------- ----------
	24-----44D9-2--B ---------- 94-C842--964---- ---------- ----------
	34-2-----59-2--2 ---------- A48E-4-4-85-82-B ---------- ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(0,1,0), (0,2,0)
7	**Positions of cube variables**
	---------- C4-8---61-2411-8 ---------- -2--54-88-136242 ----------
	---------- --29---611-21--B ---------- --284C--C-5A6242 ----------
	---------- C421-1-C-92419-3 ---------- 82--98--C-58-248 ----------
	---------- 8-28-1-818-6-9-2 ---------- 8-2894-8--4122-A ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(1,1,0), (1,2,0)
8	**Positions of cube variables**
	---------- ---------- -24-2-C41A---1-- ---------- C4-48B--1-21E42-
	---------- ---------- 26C82--492---1-1 ---------- 84-48B-1--62C42-
	---------- ---------- A598-18--8-4-2-1 ---------- 4-8418--1-61A-2-
	---------- ---------- A1D8-1448A-4-2-- ---------- 848492-1--4344--
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(2,1,0), (2,2,0)
9	**Positions of cube variables**
	41132-1-4A48-928 ---------- ---------- 248-8---911A-3-- ----------
	43-12-8--44928-A ---------- ---------- 24---8--9-1----9 ----------
	--12249-46C92--2 ---------- ---------- 2C8---1-4912-71B ----------
	-2-1248-4889-92- ---------- ---------- -88-881-D9-A-412 ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(3,1,0), (3,2,0)
10	**Positions of cube variables**
	---------- 36118--18-811-8- ---------- ---------- -44-2-19--C-15-2
	---------- 1A-18--1F--6B-F- ---------- ---------- -97---3-4-1-29-1
	---------- 2C1-8---2-852-1- ---------- ---------- -D2-2-394-C43C-1
	---------- -6-18---F--38-F- ---------- ---------- -C7---2-4-14---2
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(4,1,0), (4,2,0)

Table 13: Cube variables for attacking 9-round KMAC256

1	**Positions of cube variables**
	5D6-1149-E843113 C--4-2--1--2-182 -52-828----8C8-2 ----A-E2-4--2182 ---4--3--3---84-
	D56-91-944F-511B 4--4-2-----2-182 --21C2--4-2-88-2 4--2A2A2----2-9- ------3--2--1-4-
	5D-8B1494F7-D138 C--4-2--1--2-182 -5-1C-8-4-28C8-2 4--2A262-4--2112 ---4--3--1--184-
	DD-8A14-45B4B122 ---------- ---------- ---------- ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(0,0,0), (0,1,0), (1,0,63), (1,2,63), (2,1,30), (2,2,30)

2	**Positions of cube variables**
	5-58-8B1D5-87-93 342C-41822-51849 21618-843-8486-1 4-13-86486-4984E 8-38-249-3-1---2
	42888-3-D---4-1- 1----21-2-----6- 3-4---841-8-86-4 4--2286-1--4984- 8-3-8--1-2-1---2
	16-C8-91-7-8B49A 342C-6-8-2-51869 11218-843-84-6-5 --132-6496----4E ---88248-1------
	46DC-8-1D2-8941B ---------- ---------- ---------- ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(4,1,0), (4,2,0)
	Conditions
	$b[0][0][18] = k'[0][0][18] + 1,\ b[1][4][29] = k'[1][4][29] + 1,\ b[3][4][39] = k'[3][4][39],\ b[1][0][26] = k'[1][0][26],$
	$b[0][1][20] = k'[0][1][20] + 1,\ b[1][4][39] = k'[1][4][39] + 1,\ b[2][0][46] = k'[2][0][46],\ b[1][1][26] = k'[1][1][26],$
	$b[0][2][20] = k'[0][2][20] + 1,\ b[1][4][47] = k'[1][4][47] + 1,\ b[2][1][20] = k'[2][1][20],\ b[1][1][52] = k'[1][1][52],$
	$b[0][3][56] = k'[0][3][56] + 1,\ b[2][4][13] = k'[2][4][13] + 1,\ b[2][2][43] = k'[2][2][43],\ b[1][2][14] = k'[1][2][14],$
	$b[0][4][26] = k'[0][4][26] + 1,\ b[2][4][32] = k'[2][4][32] + 1,\ b[2][3][45] = k'[2][3][45],\ b[3][1][0] = k'[3][1][0],$
	$b[1][1][14] = k'[1][1][14] + 1,\ b[2][4][58] = k'[2][4][58] + 1,\ b[2][3][46] = k'[2][3][46],\ b[1][1][0] = k'[1][1][0],$
	$b[1][1][29] = k'[1][1][29] + 1,\ b[4][0][8] = k'[4][0][8] + 1,\ b[3][0][13] = k'[3][0][13],\ b[1][1][1] = k'[1][1][1],$
	$b[1][2][36] = k'[1][2][36] + 1,\ b[3][3][17] = k'[3][3][17],\ b[3][1][13] = k'[3][1][13],\ b[1][0][1] = k'[1][0][1],$
	$b[1][2][55] = k'[1][2][55] + 1,\ b[3][3][44] = k'[3][3][44],\ b[4][4][42] = k'[4][4][42],$
	$b[1][3][30] = k'[1][3][30] + 1,\ b[3][3][58] = k'[3][3][58],\ b[4][4][54] = k'[4][4][54].$

3	**Positions of cube variables**
	ADCA402-84-26B1- 5-E5A841----2--- 1-3----C-4-888C- -2222-18EEA-1-C- --5---115--21--1
	E-18-4--8-126412 1-E-98-9----2--- --3-8--4A--4-2C4 42-36-1921A-1-C2 -91---1-5--23--1
	2CDA441-841-6C92 4-45B-48-------- 1-2-8--CA4-C8AC4 4-234-19EF8---C2 -94----14---2---
	E98A-43-84126392 ---------- ---------- ---------- ----------
	---------- ---------- ---------- ---------- ----------
	Position of the conditional cube variable
	(4,0,0), (4,1,0)
	Conditions
	$b[0][1][20] = k'[0][1][20] + 1,\ b[2][4][28] = k'[2][4][28] + 1,\ b[2][1][20] = k'[2][1][20],\ b[1][1][0] = k'[1][1][0],$
	$b[0][4][20] = k'[0][4][20] + 1,\ b[4][0][31] = k'[4][0][31] + 1,\ b[2][4][36] = k'[2][4][36],\ b[1][4][5] = k'[1][4][5],$
	$b[0][4][26] = k'[0][4][26] + 1,\ b[4][1][21] = k'[4][1][21] + 1,\ b[3][1][17] = k'[3][1][17],\ b[2][3][3] = k'[2][3][3],$
	$b[0][4][62] = k'[0][4][62] + 1,\ b[4][3][27] = k'[4][3][27] + 1,\ b[3][2][31] = k'[3][2][31],\ b[2][4][3] = k'[2][4][3],$
	$b[1][1][16] = k'[1][1][16] + 1,\ b[4][0][4] = k'[4][0][4] + 1,\ b[3][2][51] = k'[3][2][51],\ b[3][1][0] = k'[3][1][0],$
	$b[1][1][29] = k'[1][1][29] + 1,\ b[4][4][4] = k'[4][4][4] + 1,\ b[3][4][31] = k'[3][4][31],\ b[3][1][4] = k'[3][1][4],$
	$b[1][2][60] = k'[1][2][60] + 1,\ b[1][0][11] = k'[1][0][11],\ b[4][4][40] = k'[4][4][40],\ b[3][2][9] = k'[3][2][9],$
	$b[1][3][60] = k'[1][3][60] + 1,\ b[1][0][26] = k'[1][0][26],\ b[4][4][42] = k'[4][4][42],\ b[3][3][0] = k'[3][3][0],$
	$b[1][4][29] = k'[1][4][29] + 1,\ b[1][3][27] = k'[1][3][27].$

Table 14: One 64-dimensional cube of Lake KEYAK

Cube variables
$a[0][1][36] = v_1, a[0][3][36] = v_2, a[0][4][36] = v_1 + v_2, a[0][2][52] = a[0][3][52] = v_3,$ $a[0][1][53] = a[0][2][53] = v_4, a[0][1][62] = v_5, a[0][2][62] = v_6, a[0][4][62] = v_5 + v_6,$ $a[0][1][63] = v_7, a[0][3][63] = v_8, a[0][4][63] = v_7 + v_8, a[1][1][19] = v_9,$ $a[1][2][19] = v_{10}, a[1][3][19] = v_{11}, a[1][4][19] = v_9 + v_{10} + v_{11}, a[1][1][31] = v_{12},$ $a[1][2][31] = v_{13}, a[1][4][31] = v_{12} + v_{13}, a[1][2][32] = v_{14}, a[1][3][32] = v_{15},$ $a[1][4][32] = v_{14} + v_{15}, a[1][1][34] = v_{16}, a[1][2][34] = v_{17}, a[1][4][34] = v_{16} + v_{17},$ $a[1][2][37] = a[1][3][37] = v_{18}, a[1][1][39] = a[1][4][39] = v_{19}, a[1][2][41] = a[1][4][41] = v_{20},$ $a[1][3][51] = a[1][4][51] = v_{21}, a[1][1][60] = a[1][3][60] = v_{22}, \mathbf{a[2][2][0] = a[2][3][0] = v_0},$ $a[2][2][4] = a[2][4][4] = v_{23}, a[2][1][10] = a[2][3][10] = v_{24}, a[2][1][13] = v_{25},$ $a[2][2][13] = v_{26}, a[2][4][13] = v_{25} + v_{26}, a[2][1][19] = v_{27}, a[2][3][19] = v_{28},$ $a[2][4][19] = v_{27} + v_{28}, a[2][1][28] = v_{29}, a[2][3][28] = v_{30}, a[2][4][28] = v_{29} + v_{30},$ $a[2][3][31] = a[2][4][31] = v_{31}, a[2][1][37] = v_{32}, a[2][2][37] = v_{33}, a[2][3][37] = v_{34},$ $a[2][4][37] = v_{32} + v_{33} + v_{34}, a[2][2][39] = v_{35}, a[2][3][39] = v_{36},$ $a[2][4][39] = v_{35} + v_{36}, a[2][1][45] = a[2][4][45] = v_{37}, a[2][2][55] = v_{38}, a[2][3][55] = v_{39},$ $a[2][4][55] = v_{38} + v_{39}, a[2][1][57] = a[2][4][57] = v_{40}, a[2][1][60] = a[2][4][60] = v_{41},$ $a[3][1][11] = v_{42}, a[3][3][11] = v_{43}, a[3][4][11] = v_{42} + v_{43}, a[3][1][20] = v_{44},$ $a[3][3][20] = v_{45}, a[3][4][20] = v_{44} + v_{45}, a[3][2][29] = a[3][4][29] = v_{46},$ $a[3][2][31] = a[3][3][31] = v_{47}, a[3][2][45] = a[3][3][45] = v_{48}, a[4][1][5] = v_{49},$ $a[4][2][5] = v_{50}, a[4][3][5] = v_{49} + v_{50}, a[4][1][14] = a[4][2][14] = v_{51},$ $a[4][2][16] = a[4][3][16] = v_{52}, a[4][1][21] = v_{53}, a[4][2][21] = v_{54}, a[4][3][21] = v_{53} + v_{54},$ $a[4][1][22] = a[4][3][22] = v_{55}, a[4][1][24] = a[4][3][24] = v_{56}, a[4][1][32] = a[4][3][32] = v_{57},$ $a[4][1][34] = v_{58}, a[4][2][34] = v_{59}, a[4][3][34] = v_{58} + v_{59}, a[4][1][43] = a[4][2][43] = v_{60},$ $a[4][1][58] = v_{61}, a[4][2][58] = v_{62}, a[4][3][58] = v_{61} + v_{62}, a[4][2][59] = a[4][3][59] = v_{63}.$
Bit conditions
$a[2][0][62] + a[1][1][63] + a[2][1][62] + a[0][2][63] + a[2][2][62] + a[2][3][62] + a[2][4][62]$ $+ k_{55} + 1 = 0,$ $a[0][0][5] + a[0][1][5] + a[2][1][4] + a[0][2][5] + a[1][2][5] + a[0][3][5] + a[2][3][4]$ $+ a[0][4][5] + k_{124} + 1 = 0,$ $a[2][0][22] + a[4][0][21] + a[2][1][22] + a[2][2][22] + a[2][3][22] + a[3][3][22] + a[2][4][22]$ $+ a[4][4][21] = 0,$ $a[2][0][23] + a[4][0][22] + a[2][1][23] + a[2][2][23] + a[4][2][22] + a[2][3][23] + a[2][4][23]$ $+ a[3][4][23] + a[4][4][22] = 0.$

Table 15: One 64-dimensional cube of Lake KEYAK-256

Positions of cube variables
`---------------- ---------------- ---------------- ---------------- ----------------`
`2811C-225---411- --28-1--3-18--4- 92-411-4-2-3B-18 -9---5-3F-5746-2 --D--8---9--28-7`
`2-3---3--8----5- 6-A-4---2-1----3 1---31-19-C-2-18 881--18-414448-- --8--8---9-----3`
`-881C-124---4174 4-88-1--2--6--13 -----1-59--2--1- 811--48-114-8E12 --D------1--28-5`
`-8B-C-1-58----74 6-8-41--3-1E--5- 92-42--412C19-18 -----483F-178212 ----------------`
Position of the conditional cube variable
(1,2,0),(1,3,0)
Bit conditions
$a[4][0][38] + a[0][1][38] + a[1][1][37] + a[4][1][38] + a[1][2][37] + a[4][2][38] + a[1][3][37] + a[4][3][38] + a[1][4][37] + a[4][4][38] + k_{93} + 1 = 0,$ $a[4][0][42] + a[1][1][41] + a[4][1][42] + a[0][2][42] + a[1][2][41] + a[4][2][42] + a[1][3][41] + a[4][3][42] + a[1][4][41] + a[4][4][42] + k_{97} + 1 = 0,$ $a[1][1][59] + a[3][1][58] + a[1][2][59] + a[3][2][58] + a[1][3][59] + a[2][3][59] + a[3][3][58] + a[1][4][59] + a[3][4][58] + k_{115} + k_{242} = 0,$ $a[1][1][48] + a[3][1][47] + a[1][2][48] + a[3][2][47] + a[1][3][48] + a[3][3][47] + a[1][4][48] + a[2][4][48] + a[3][4][47] + k_{104} + k_{231} = 0.$

Table 16: One 64-dimensional cube of River KEYAK

Cube variables
$a[0][2][5] = a[0][4][5] = v_1, a[0][3][7] = a[0][4][7] = v_2, a[0][3][8] = a[0][4][8] = v_3,$ $a[0][2][15] = a[0][3][15] = v_4, a[0][3][16] = a[0][4][16] = v_5, a[0][2][23] = a[0][4][23] = v_6,$ $a[0][3][28] = a[0][4][28] = v_7, \mathbf{a[0][2][29]} = \mathbf{a[0][4][29]} = \mathbf{v_0}, a[1][2][0] = a[1][4][0] = v_8,$ $a[1][3][1] = a[1][4][1] = v_9, a[1][2][9] = v_{10}, a[1][3][9] = v_{11}, a[1][4][9] = v_{10} + v_{11},$ $a[1][2][11] = a[1][3][11] = v_{12}, a[1][3][12] = a[1][4][12] = v_{13}, a[1][2][15] = a[1][3][15] = v_{14},$ $a[1][2][16] = a[1][3][16] = v_{15}, a[1][2][18] = a[1][4][18] = v_{16}, a[1][2][19] = a[1][4][19] = v_{17},$ $a[1][2][20] = a[1][4][20] = v_{18}, a[1][3][21] = a[1][4][21] = v_{19}, a[1][2][25] = v_{20}, a[1][3][25] = v_{21},$ $a[1][4][25] = v_{20} + v_{21}, a[2][2][7] = a[2][3][7] = v_{22}, a[2][2][8] = v_{23}, a[2][3][8] = v_{24},$ $a[2][4][8] = v_{23} + v_{24}, a[2][2][17] = a[2][3][17] = v_{25}, \mathbf{a[2][3][18]} = \mathbf{a[2][4][18]} = \mathbf{v_0},$ $a[2][3][19] = a[2][4][19] = v_{26}, a[2][2][24] = a[2][4][24] = v_{27}, a[2][2][29] = a[2][4][29] = v_{28},$ $a[2][2][30] = a[2][4][30] = v_{29}, a[2][2][31] = a[2][3][31] = v_{30}, a[3][2][0] = v_{31}, a[3][3][0] = v_{32},$ $a[3][4][0] = v_{31} + v_{32}, a[3][2][1] = v_{33}, a[3][3][1] = v_{34}, a[3][4][1] = v_{33} + v_{34},$ $a[3][2][2] = v_{35}, a[3][3][2] = v_{36}, a[3][4][2] = v_{35} + v_{36}, a[3][2][3] = v_{37}, a[3][3][3] = v_{38},$ $a[3][4][3] = v_{37} + v_{38}, a[3][2][4] = v_{39}, a[3][3][4] = v_{40}, a[3][4][4] = v_{39} + v_{40},$ $a[3][2][5] = v_{41}, a[3][3][5] = v_{42}, a[3][4][5] = v_{41} + v_{42}, a[3][2][9] = a[3][3][9] = v_{43},$ $a[3][3][11] = a[3][4][11] = v_{44}, a[3][2][13] = v_{45}, a[3][3][13] = v_{46}, a[3][4][13] = v_{45} + v_{46},$ $a[3][2][17] = a[3][3][17] = v_{47}, a[3][2][25] = a[3][3][25] = v_{48}, a[3][2][28] = a[3][3][28] = v_{49},$ $a[4][1][1] = a[4][2][1] = v_{50}, a[4][1][2] = a[4][2][2] = v_{51}, a[4][1][3] = a[4][2][3] = v_{52},$ $a[4][1][7] = a[4][2][7] = v_{53}, a[4][1][8] = a[4][3][8] = v_{54}, a[4][1][14] = v_{55}, a[4][2][14] = v_{56},$ $a[4][3][14] = v_{55} + v_{56}, a[4][1][15] = a[4][2][15] = v_{57}, a[4][1][16] = v_{58}, a[4][2][16] = v_{59},$ $a[4][3][16] = v_{58} + v_{59}, a[4][1][22] = a[4][2][22] = v_{60}, \mathbf{a[4][2][25]} = \mathbf{a[4][3][25]} = \mathbf{v_0},$ $a[4][1][27] = v_{61}, a[4][2][27] = v_{62}, a[4][3][27] = v_{61} + v_{62}, a[4][1][31] = a[4][2][31] = v_{63}.$
Bit conditions
$a[4][0][18] + a[2][1][19] + a[4][1][18] + a[2][2][19] + a[4][2][18] + a[4][3][18] + a[4][4][18] + k_{75}$ $+ k_{107} = 0, a[0][1][14] + a[2][1][13] + a[0][2][14] + a[2][2][13] + a[0][3][14] + a[2][3][13]$ $+ a[0][4][14] + a[2][4][13] + k_6 + k_{38} + k_{69} = 0, a[0][1][11] + a[3][1][12] + a[4][1][12]$ $+ a[0][2][11] + a[3][2][12] + a[0][3][11] + a[3][3][12] + a[0][4][11] + a[3][4][12] + k_3 + k_{100}$ $+ 1 = 0, a[4][0][8] + a[2][1][9] + a[3][1][9] + a[2][2][9] + a[4][2][8] + a[2][3][9] + a[2][4][9]$ $+ a[4][4][8] + k_{65} + 1 = 0, a[2][1][8] + a[3][2][8] + a[4][3][7] + a[4][4][7] + k_{64} + k_{127}$ $+ 1 = 0, a[0][1][23] + a[2][1][22] + a[1][2][23] + a[2][2][22] + a[0][3][23] + a[2][3][22] + a[2][4][22]$ $+ k_{15} + k_{78} + 1 = 0, a[0][1][19] + a[2][1][18] + a[0][2][19] + a[2][2][18] + a[0][3][19] + a[1][3][19]$ $+ a[0][4][19] + k_{11} + k_{74} = 0, a[0][0][2] + a[0][1][2] + a[2][1][1] + a[0][2][2] + a[2][2][1]$ $+ a[0][3][2] + a[1][3][2] + a[2][3][1] + a[0][4][2] + a[2][4][1] + k_{57} + 1 = 0, a[0][0][6] + a[0][1][6]$ $+ a[3][1][7] + a[0][2][6] + a[3][2][7] + a[0][3][6] + a[3][3][7]$ $+ a[4][3][7] + a[0][4][6] + a[3][4][7] + k_{95} + 1 = 0, a[4][0][23] + a[1][1][22] + a[4][1][23]$ $+ a[1][2][22] + a[4][2][23] + a[0][3][23] + a[1][3][22] + a[4][3][23] + a[1][4][22] + a[4][4][23]$ $+ k_{46} = 0, a[4][0][15] + a[1][1][14] + a[1][2][14] + a[1][3][14] + a[4][3][15] + a[0][4][15]$ $+ a[1][4][14] + a[4][4][15] + k_{38} = 0, a[4][0][8] + a[2][1][9] + a[2][2][9] + a[4][2][8]$ $+ a[2][3][9] + a[2][4][9] + a[3][4][9] + a[4][4][8] + k_{65} = 0.$

Table 17: One 64-dimensional cube of KETJE Major.

Cube variables
$a[1][3][4] = a[1][4][4] = v_1, a[1][2][28] = a[1][4][28] = v_2, a[1][0][29] = a[1][4][29] = v_3,$ $a[1][0][30] = v_4, a[1][2][30] = v_5, a[1][3][30] = v_4 + v_5, a[1][2][39] = a[1][3][39] = v_6,$ $a[1][2][44] = a[1][3][44] = v_7, a[1][2][52] = a[1][3][52] = v_8, a[1][3][56] = a[1][4][56] = v_9,$ $a[1][0][57] = v_{10}, a[1][2][57] = v_{11}, a[1][3][57] = v_{12}, a[1][4][57] = v_{10} + v_{11} + v_{12},$ $\mathbf{a[2][0][0]} = \mathbf{a[2][1][0]} = \mathbf{v_0}, a[2][3][9] = a[2][4][9] = v_{13}, a[2][3][10] = a[2][4][10] = v_{14},$ $a[2][1][19] = a[2][2][19] = v_{15}, a[2][0][21] = v_{16}, a[2][2][21] = v_{17}, a[2][3][21] = v_{18},$ $a[2][4][21] = v_{16} + v_{17} + v_{18}, a[2][0][28] = a[2][1][28] = v_{19}, a[2][3][33] = a[2][4][33] = v_{20},$ $a[2][2][38] = a[2][4][38] = v_{21}, a[2][1][58] = a[2][2][58] = v_{22}, a[3][0][3] = v_{23}, a[3][1][3] = v_{24},$ $a[3][4][3] = v_{23} + v_{24}, a[3][0][4] = v_{25}, a[3][1][4] = v_{26}, a[3][3][4] = v_{27},$ $a[3][4][4] = v_{25} + v_{26} + v_{27}, a[3][1][12] = v_{28}, a[3][2][12] = v_{29}, a[3][3][12] = v_{28} + v_{29},$ $a[3][1][29] = v_{30}, a[3][2][29] = v_{31}, a[3][3][29] = v_{30} + v_{31}, a[3][0][39] = a[3][4][39] = v_{32},$ $a[3][2][48] = a[3][3][48] = v_{33}, a[3][0][56] = v_{34}, a[3][1][56] = v_{35}, a[3][2][56] = v_{34} + v_{35},$ $a[3][1][60] = a[3][4][60] = v_{36}, a[4][1][1] = v_{37}, a[4][2][1] = v_{38}, a[4][4][1] = v_{37} + v_{38},$ $a[4][0][5] = v_{39}, a[4][1][5] = v_{40}, a[4][4][5] = v_{39} + v_{40}, a[4][0][10] = v_{41}, a[4][1][10] = v_{42},$ $a[4][2][10] = v_{43}, a[4][3][10] = v_{44}, a[4][4][10] = v_{41} + v_{42} + v_{43} + v_{44}, a[4][1][17] = v_{45},$ $a[4][3][17] = v_{46}, a[4][4][17] = v_{45} + v_{46}, a[4][0][22] = v_{47}, a[4][2][22] = v_{48},$ $a[4][3][22] = v_{47} + v_{48}, a[4][2][23] = v_{49}, a[4][3][23] = v_{50}, a[4][4][23] = v_{49} + v_{50},$ $a[4][2][34] = v_{51}, a[4][3][34] = v_{52}, a[4][4][34] = v_{51} + v_{52}, a[4][1][42] = v_{53},$ $a[4][2][42] = v_{54}, a[4][3][42] = v_{53} + v_{54}, a[4][0][55] = a[4][2][55] = v_{55}, a[4][0][56] = v_{56},$ $a[4][2][56] = v_{57}, a[4][4][56] = v_{56} + v_{57}, a[4][1][61] = v_{58}, a[4][2][61] = v_{59},$ $a[4][3][61] = v_{60}, a[4][4][61] = v_{58} + v_{59} + v_{60}, a[4][0][62] = v_{61}, a[4][1][62] = v_{62},$ $a[4][2][62] = v_{63}, a[4][3][62] = v_{61} + v_{62} + v_{63}.$
Bit conditions
$a[0][0][5] + a[1][0][5] + a[2][0][4] + a[0][1][5] + a[2][1][4] + a[0][2][5] + a[0][3][5] + a[2][3][4]$ $+ a[0][4][5] + a[2][4][4] + k_{124} + 1,$ $a[2][0][7] + a[4][0][6] + a[2][1][7] + a[3][1][7] + a[4][1][6] + a[4][2][6] + a[2][3][7] + a[4][3][6]$ $+ a[2][4][7] + a[4][4][6] + k_{127},$ $a[2][0][45] + a[4][0][44] + a[2][1][45] + a[4][1][44] + a[2][2][45] + a[3][2][45] + a[4][2][44]$ $+ a[2][3][45] + a[4][3][44] + a[2][4][45] + a[4][4][44],$ $a[2][0][59] + a[0][1][60] + a[2][1][59] + a[0][2][60] + a[2][2][59] + a[0][3][60] + a[2][3][59]$ $+ a[0][4][60] + a[1][4][60] + a[2][4][59] + k_{52} + 1.$

Table 18: One 64-dimensional cube of KETJE Minor

Cube variables
$a[0][1][1] = v_1, a[0][2][1] = v_2, a[0][3][1] = v_3, a[0][4][1] = v_1 + v_2 + v_3,$ $a[0][1][7] = a[0][2][7] = v_4, a[0][1][11] = v_5, a[0][2][11] = v_6, a[0][4][11] = v_5 + v_6,$ $a[0][1][13] = v_7, a[0][2][13] = v_8, a[0][3][13] = v_7 + v_8, a[0][1][16] = a[0][3][16] = v_9,$ $a[0][2][17] = a[0][3][17] = v_{10}, a[0][1][20] = v_{11}, a[0][2][20] = v_{12},$ $a[0][3][20] = v_{11} + v_{12}, a[0][1][22] = v_{13}, a[0][2][22] = v_{14}, a[0][3][22] = v_{15},$ $a[0][4][22] = v_{13} + v_{14} + v_{15}, a[0][1][23] = v_{16}, a[0][2][23] = v_{17},$ $a[0][3][23] = v_{16} + v_{17}, a[0][2][26] = a[0][4][26] = v_{18}, a[0][1][30] = v_{19}, a[0][2][30] = v_{20},$ $a[0][4][30] = v_{19} + v_{20}, a[1][0][4] = v_{21}, a[1][2][4] = v_{22}, a[1][3][4] = v_{21} + v_{22},$ $a[1][2][15] = a[1][3][15] = v_{23}, a[1][0][19] = v_{24}, a[1][2][19] = v_{25}, a[1][3][19] = v_{24} + v_{25},$ $a[1][0][20] = a[1][2][20] = v_{26}, a[1][0][23] = a[1][2][23] = v_{27}, a[1][0][26] = a[1][2][26] = v_{28},$ $a[1][2][28] = a[1][3][28] = v_{29}, a[1][0][30] = a[1][3][30] = v_{30}, a[2][0][0] = a[2][4][0] = v_{31},$ $a[2][0][1] = a[2][3][1] = v_{32}, a[2][3][2] = a[2][4][2] = v_{33}, a[2][1][4] = a[2][3][4] = v_{34},$ $a[2][1][5] = a[2][3][5] = v_{35}, a[2][0][6] = a[2][4][6] = v_{36}, a[2][0][7] = a[2][3][7] = v_{37},$ $a[2][3][8] = a[2][4][8] = v_{38}, a[2][0][9] = a[2][1][9] = v_{39}, a[2][1][11] = v_{40}, a[2][3][11] = v_{41},$ $a[2][4][11] = v_{40} + v_{41}, a[2][0][17] = a[2][3][17] = v_{42}, a[2][1][19] = a[2][4][19] = v_{43},$ $a[2][0][20] = a[2][1][20] = v_{44}, a[2][0][22] = a[2][4][22] = v_{45}, a[2][0][23] = a[2][4][23] = v_{46},$ $a[2][0][26] = a[2][3][26] = v_{47}, a[2][0][27] = v_{48}, a[2][1][27] = v_{49}, a[2][3][27] = v_{48} + v_{49},$ $a[2][1][28] = v_{50}, a[2][3][28] = v_{51}, a[2][4][28] = v_{50} + v_{51}, \mathbf{a[3][1][0] = a[3][4][0] = v_0},$ $a[3][1][4] = v_{52}, a[3][2][4] = v_{53}, a[3][4][4] = v_{52} + v_{53}, a[3][2][5] = a[3][4][5] = v_{54},$ $a[3][0][11] = v_{55}, a[3][1][11] = v_{56}, a[3][2][11] = v_{57}, a[3][4][11] = v_{55} + v_{56} + v_{57},$ $a[3][2][13] = a[3][4][13] = v_{58}, a[3][0][19] = a[3][1][19] = v_{59}, a[3][2][23] = a[3][4][23] = v_{60},$ $a[4][0][3] = a[4][3][3] = v_{61}, a[4][0][4] = v_{62}, a[4][1][4] = v_{63}, a[4][2][4] = v_{62} + v_{63},$ $a[4][0][9] = v_{64}, a[4][2][9] = v_{65}, a[4][3][9] = v_{64} + v_{65}, a[4][1][10] = a[4][2][10] = v_{66},$ $a[4][0][12] = a[4][3][12] = v_{67}, a[4][0][13] = a[4][2][13] = v_{68}, a[4][1][14] = a[4][3][14] = v_{69},$ $a[4][1][17] = a[4][3][17] = v_{70}, a[4][0][23] = v_{71}, a[4][1][23] = v_{72}, a[4][2][23] = v_{73},$ $a[4][3][23] = v_{71} + v_{72} + v_{73}, a[4][0][26] = a[4][2][26] = v_{74}, a[4][1][29] = v_{75},$ $a[4][2][29] = v_{76}, a[4][3][29] = v_{75} + v_{76}, a[4][0][30] = v_{77}, a[4][1][30] = v_{78},$ $a[4][2][30] = v_{77} + v_{78}, v_{29} + v_{39} + v_{55} + v_{58} + v_{67} = 0, v_5 + v_6 + v_{26} + v_{31} + v_{73} = 0,$ $v_2 + v_{28} + v_{36} + v_{55} + v_{56} + v_{76} = 0, v_3 + v_{48} + v_{49} = 0, v_{10} + v_{16} + v_{41} + v_{52} = 0,$ $v_{18} = 0, v_6 + v_{22} = 0, v_{30} + v_{51} + v_{61} = 0, v_{16} + v_{17} + v_{24} + v_{25} + v_{30} + v_{32} + v_{42} + v_{62} = 0,$ $v_{21} + v_{22} + v_{33} + v_{64} + v_{65} + v_{78} = 0, v_{29} + v_{47} + v_{68} = 0, v_5 + v_{19} + v_{20} + v_{39} + v_{43} = 0,$ $v_{19} + v_{50} + v_{56} + v_{69} = 0, v_{11} + v_{38} + v_{63} = 0, v_7 + v_8 + v_{26} + v_{37} + v_{46} + v_{69} + v_{74} = 0.$
Bit conditions
$a[3][0][29] + a[4][0][29] + a[0][1][28] + a[3][1][29] + a[0][2][28] + a[3][2][29] + a[0][3][28]$ $+ a[0][4][28] + a[3][4][29] + k_{20} + k_{117},$ $a[1][0][25] + a[2][0][25] + a[3][0][24] + a[3][1][24] + a[1][2][25] + a[3][2][24] + a[1][3][25]$ $+ a[1][4][25] + a[3][4][24] + k_{49} + k_{112} + 1,$ $a[3][0][16] + a[0][1][15] + a[3][1][16] + a[0][2][15] + a[3][2][16] + a[4][2][16] + a[0][3][15]$ $+ a[0][4][15] + a[3][4][16] + k_7 + k_{104},$ $a[1][0][9] + a[3][0][8] + a[3][1][8] + a[1][2][9] + a[3][2][8] + a[1][3][9] + a[2][3][9] + a[1][4][9]$ $+ a[3][4][8] + k_{33} + k_{96} + 1.$

Table 19: One 64-dimensional cube of KETJE SR v1

Cube variables
$a[0][2][0] = a[0][4][0] = v_1, a[0][2][1] = a[0][4][1] = v_2, a[0][2][2] = a[0][4][2] = v_3,$ $a[0][2][3] = a[0][4][3] = v_4, \mathbf{a[0][2][6]} = \mathbf{a[0][4][6]} = \mathbf{v_0}, a[0][3][7] = a[0][4][7] = v_5,$ $a[0][2][8] = v_6, a[0][3][8] = v_7, a[0][4][8] = v_6 + v_7, a[0][2][10] = a[0][4][10] = v_8,$ $a[0][3][14] = a[0][4][14] = v_9, a[0][2][15] = a[0][4][15] = v_{10}, a[1][2][0] = a[1][4][0] = v_{11},$ $a[1][2][1] = v_{12}, a[1][3][1] = v_{13}, a[1][4][1] = v_{12} + v_{13}, a[1][2][2] = v_{14},$ $a[1][3][2] = v_{15}, a[1][4][2] = v_{14} + v_{15}, a[1][2][3] = v_{16}, a[1][3][3] = v_{17},$ $a[1][4][3] = v_{16} + v_{17}, a[1][2][6] = a[1][4][6] = v_{18}, a[1][2][7] = a[1][4][7] = v_{19},$ $a[1][2][8] = a[1][4][8] = v_{20}, a[1][2][9] = a[1][4][9] = v_{21}, a[1][3][10] = a[1][4][10] = v_{22},$ $a[1][2][11] = a[1][4][11] = v_{23}, a[1][3][13] = a[1][4][13] = v_{24}, a[1][2][15] = a[1][3][15] = v_{25},$ $a[2][2][0] = v_{26}, a[2][3][0] = v_{27}, a[2][4][0] = v_{26} + v_{27}, a[2][2][4] = a[2][4][4] = v_{28},$ $a[2][2][5] = v_{29}, a[2][3][5] = v_{30}, a[2][4][5] = v_{29} + v_{30}, a[2][2][6] = a[2][4][6] = v_{31},$ $a[2][2][7] = v_{32}, a[2][3][7] = v33, a[2][4][7] = v_{32} + v_{33}, a[2][2][8] = v_{34}, a[2][3][8] = v_{35},$ $a[2][4][8] = v_{34} + v_{35}, a[2][2][14] = a[2][4][14] = v_{36}, a[2][2][15] = a[2][3][15] = v_{37},$ $\mathbf{a[3][2][0]} = \mathbf{a[3][4][0]} = \mathbf{v_0}, a[3][2][1] = v_{38}, a[3][3][1] = v_{39}, a[3][4][1] = v_{38} + v_{39},$ $a[3][2][2] = v_{40}, a[3][3][2] = v_{41}, a[3][4][2] = v_{40} + v_{41}, a[3][2][3] = v_{42}, a[3][3][3] = v_{43},$ $a[3][4][3] = v_{42} + v_{43}, a[3][2][7] = a[3][3][7] = v_{44}, a[3][2][8] = a[3][4][8] = v_{45}, a[3][2][9] = v_{46},$ $a[3][3][9] = v_{47}, a[3][4][9] = v_{46} + v_{47}, a[3][2][10] = a[3][4][10] = v_{48},$ $a[3][2][11] = a[3][4][11] = v_{49}, a[3][2][13] = a[3][4][13] = v_{50}, a[4][2][2] = a[4][4][2] = v_{51},$ $a[4][1][3] = v_{52}, a[4][2][3] = v_{53}, a[4][4][3] = v_{52} + v_{53}, a[4][2][4] = a[4][4][4] = v_{54},$ $a[4][2][5] = a[4][4][5] = v_{55}, a[4][1][6] = v_{56}, a[4][2][6] = v_{57}, a[4][3][6] = v_{58},$ $a[4][4][6] = v_{56} + v_{57} + v_{58}, a[4][3][7] = a[4][4][7] = v_{59}, a[4][1][8] = a[4][2][8] = v_{60},$ $a[4][1][11] = v_{61}, a[4][2][11] = v_{62}, a[4][4][11] = v_{61} + v_{62}, a[4][1][12] = v_{63}, a[4][2][12] = v_{64},$ $a[4][4][12] = v_{63} + v_{64}, a[4][1][13] = v_{65}, a[4][2][13] = v_{66}, a[4][4][13] = v_{65} + v_{66},$ $v_{16} + v_{17} + v_{34} + v_{35} + v_{59} = 0, v_8 + v_{16} + v_{23} + v_{26} + v_{27} + v_{57} + 1 = 0,$ $v_{24} + v_{27} + v_{28} + v_{60}.$
27 bit conditions can be derived from B, V and C
B, V and C

`---- ---- 8E-F 51C- -E3F | ---- ---- 4--- ---- -1C- | 8E-F 9FCF DE3F 5FFF -E3F`
`---- 9483 2A7C D481 283E | --8- -2-- ---- 222- 8-8- | BCFF DCDF 7E7F FCBF BC3F`
`---- ---- 1E5F C--- 173F | -1-- -2-- ---- -3-- 8--- | 1C5F DC5F 5E7F D63F 173F`
`---- ---- 3E2F C-D- -F2F | -1-- ---- ---- 212- 8-C- | 3E2F DEDF 7E3F CEFF -F2F`
`---- ---- BE1C --83 BF3C | ---- ---- ---- A1-- --8- | BE1C 1E9F BE3F BFBF BF3C`