New MILP Modeling: Improved Conditional Cube Attacks on Keccak-Based Constructions

Ling Song[1,2(✉)], Jian Guo[1(✉)], Danping Shi[2(✉)], and San Ling[1(✉)]

[1] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore
{guojian,lingsan}@ntu.edu.sg

[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{songling,shidanping}@iie.ac.cn

Abstract. In this paper, we propose a new MILP modeling to find better or even optimal choices of conditional cubes, under the general framework of conditional cube attacks. These choices generally find new or improved attacks against the keyed constructions based on Keccak permutation and its variants, including Keccak-MAC, KMAC, Keyak, and Ketje, in terms of attack complexities or the number of attacked rounds. Interestingly, conditional cube attacks were applied to round-reduced Keccak-MAC, but not to KMAC despite the great similarity between Keccak-MAC and KMAC, and the fact that KMAC is the NIST standard way of constructing MAC from SHA-3. As examples to demonstrate the effectiveness of our new modeling, we report key recovery attacks against KMAC128 and KMAC256 reduced to 7 and 9 rounds, respectively; the best attack against Lake Keyak with 128-bit key 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 of 256 bits; attack complexity improvements are found generally on other constructions. Our new model is also applied to Keccak-based full-state keyed sponge and gives a positive answer to the open question proposed by Bertoni *et al.* whether cube attacks can be extended to more rounds by exploiting full-state absorbing. To verify the correctness of our attacks, reduced-variants of the attacks are implemented and verified on a PC practically. It is remarked that this work does not threaten the security of any full version of the instances analyzed in this paper.

Keywords: Keccak · SHA-3 · KMAC · Keyak · Ketje · Full-state Conditional cube attack · MILP

1 Introduction

The Keccak hash function family [5] is a proposal designed by Bertoni *et al.* and submitted to the SHA-3 competition [22] in 2008. It was selected as the final winner of the competition in 2012, and subsequently standardized as SHA-3 [29] in

T. Peyrin and S. Galbraith (Eds.): ASIACRYPT 2018, LNCS 11273, pp. 65–95, 2018.
https://doi.org/10.1007/978-3-030-03329-3_3

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, and hence 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.* [17] 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 [27], Song *et al.* 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.* [23] and Dinur *et al.* [12].

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 [4] and KMAC [30]. In [13], 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 [19]. The authenticated encryption schemes Keyak [7] and Ketje [6] 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 [19], and to Ketje [15,20] for up to 7 out of 13 rounds. Differently from the traditional way of reducing the strength of the design by round number, there is a recent attack against full Ketje with tweaked rate size by Fuhr *et al.* [16]. Kravatte [2] is a pseudorandom function by instantiating the Farfalle construction with Keccak-p. Algebraic attacks on Kravatte, including cube attacks, which take advantage of structural properties of Farfalle, were proposed in [9].

Following a similar design strategy used for Keccak-MAC, KMAC [30] 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 most bits of that for Keccak-MAC are known. Similar observations were discovered and made use of in the so-called "Full-State Keyed Duplex (FKD)" [10,21] to improve the efficiency of keyed sponge constructions. It is interesting to note, despite the great similarity between Keccak-MAC and KMAC, there is no existing cryptanalysis result against KMAC to the best of our knowledge. Also, for FKD no cube attack is proposed by exploiting the full-state absorption, as stated by the Keyak designers in [7]:

Whether these attacks can still be extended to more rounds by exploiting full-state absorbing remains an open question.

Our Contributions. Based on the previous works [15,19,20] on conditional cube attacks against Keccak-based keyed constructions, we propose a new Mixed Integer Linear Programming (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 keeping the cube size large enough 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 imposes no unnecessary conditions, hence could be able to find optimal conditional cubes, in terms of cube size and number of conditions, whenever possible. This comes with a few key techniques:

1. We are able to model 2 Keccak rounds together, *i.e.*, Sbox layer of the first round, the linear layer followed by the Sbox layer again of the second round. To do this, we exhaustively list the propagations of variables through the first Sbox layer so to keep the output of the Sbox linear. The second round is dealt in different ways.
   - For normal Keccak-based constructions, we classify the situation of the linear layer in the second round into two cases depending on whether there is spreading of variables and model them each individually.
   - For FKD, we describe column sums of the state after the Sbox layer of the first round with inequalities. With this, the diffusion of the second round can be described precisely with MILP.

   With all these together, we are able to convert all the necessary constraints in the search of conditional cubes into the MILP language.
2. For FKD, 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.

We apply this new MILP modeling to Keccak-based keyed constructions including Keccak-MAC, `KMAC`, Keyak, Ketje, and FKD and find 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 the 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. Details are 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 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 bits. Details are summarized in Table 2.
– Conditional cubes that fully linearize the first two rounds are targeted by our modeling and the open question of FKD is answered by extending cube attacks by one additional round.

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

Target	Key size	Capacity	Rounds	Time (Data)	Reference
`KMAC128`	128	256	7/24	2^{76}	Section 6.1
`KMAC256`	256	512	9/24	2^{147}	
KECCAK-MAC	128	256/512	7/24	2^{72}	[19]
		768	7/24	2^{75}	[20]
		1024	6/24	$2^{58.3}$	
		1024	6/24	2^{40}	Section 5.3
		1024	7/24	2^{111}	[25]

Very recently, another two MILP models [8,25] were proposed for cube-attack-like cryptanalysis [13], together with some new results for keyed KECCAK modes. In particular, 7 rounds of KECCAK-MAC-512 can be attacked. In cube-attack-like cryptanalysis, only the first round is linearized and the idea is to choose cube variables such that they multiply with a small number of key bits in the first round. Therefore, one only needs to pay attention to the diffusion of the linear layer in the first round. Due to this, cube-attack-like cryptanalysis performs well especially when the degrees of freedom is limited, *e.g.*, smaller versions of KETJE. The drawback is that cube-attack-like cryptanalysis is not suitable for constructions with fully unknown internal state, *e.g.*, `KMAC` and KECCAK-based FKD which are our main targets of conditional cube attacks. Whereas, in conditional cube attacks, one has to deal with two rounds in which more degrees of freedom are needed to control the diffusion of cube variables. Also, finding good conditional cubes is more challenging. However, if sufficient degrees of freedom are available, conditional cube attacks can exploit this and provide better attacks. Examples include attacks on all instances of KEYAK, KETJE Major and KETJE Minor.

Organization. The remaining part of the paper is organized as follows. Section 2 gives a detailed description of KECCAK-p based constructions, including KECCAK, `KMAC`, KEYAK and KETJE, followed by an introduction in Sect. 3 to related works. Our new MILP model is presented in Sects. 4 and 5, and applied

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

Target	Key size	Rounds	Time (Data)	Memory	nonce-respected	Reference
Lake KEYAK	128	6/12	2^{37}	-	Yes	[13]
	128	8/12	2^{74}	-	No	[19]
	128	8/12	$2^{71.01}$	-	Yes	Section 6.2
	256	9/14	$2^{137.05}$	-	Yes	
River KEYAK	128	8/12	2^{77}	-	Yes	Section 6.2
KETJE Major	128	7/13	2^{83}	-	Yes	[20]
	128	7/13	$2^{71.24}$	-	Yes	Section 6.2
KETJE Minor	128	7/13	2^{81}	-	Yes	[20]
	128	7/13	$2^{73.03}$	-	Yes	Section 6.2
KETJE SR v1	128	7/13	2^{115}	2^{50}	Yes	[15]
	128	7/13	2^{91}	-	Yes	Section 6.2
FKD[1600]	128	9/-	2^{90}	-	No	Section 6.3
KETJE Jr v1	96	5/13	$2^{36.86}$	2^{18}	Yes	[25]
KETJE Jr v2	96	5/13	$2^{34.91}$	2^{15}	Yes	
KETJE Sr v2	128	7/13	2^{99}	2^{33}	Yes	

to the key recovery attacks of KMAC, KEYAK, KETJE and full-state keyed duplex (FKD) in Sect. 6. Finally, Sect. 7 concludes the paper. Details of cubes are provided in the full version of this paper [26].

2 Description of KMAC, 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 $l = 0, \cdots, 6$. 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 Fig. 1 with x for the column index and y for the row index.

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 steps $\mathtt{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$, with details as follows.

θ: $a[x][y][z] = a[x][y][z] \oplus \bigoplus_{y=0}^{4} a[x-1][y][z] \oplus \bigoplus_{y=0}^{4} a[x+1][y][z-1]$.

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

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

ρ: $a[x][y][z] = a[x][y][(z - T(x, y))]$, where $T(x, y)s$ are rotation constants.
π: $a[y][2x + 3y][z] = a[x][y][z]$.
χ: $a[x][y][z] = a[x][y][z] \oplus (a[x + 1][y][z] \oplus 1) \cdot a[x + 2][y][z]$.
ι: $a[0][0] = a[0][0] \oplus RC_{i_r}$, where RC_{i_r} is the i_r-th round constant.

Here, '$\oplus$' 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 $\boldsymbol{d}$ bits digest are obtained.

The Keccak hash function follows the sponge construction and takes Keccak-$f[1600]$ as the underlying permutation. In 2015, Keccak was formally standardized by NIST as `SHA-3` [29], based on which more functions, including cSHAKE128, cSHAKE256 and `KMAC`, are derived in the NIST Special Publication 800-185 [30].

`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

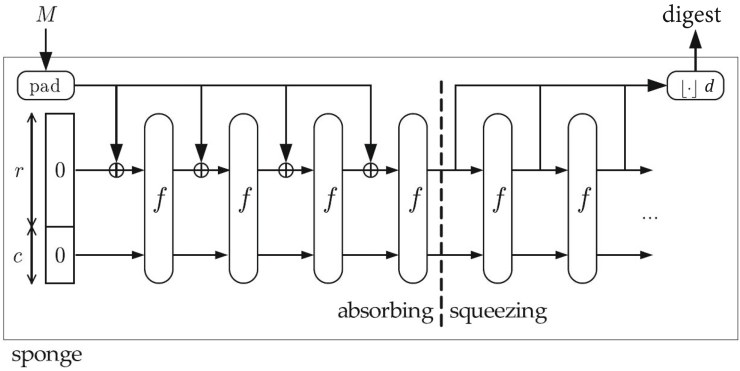

Fig. 2. Sponge construction [3].

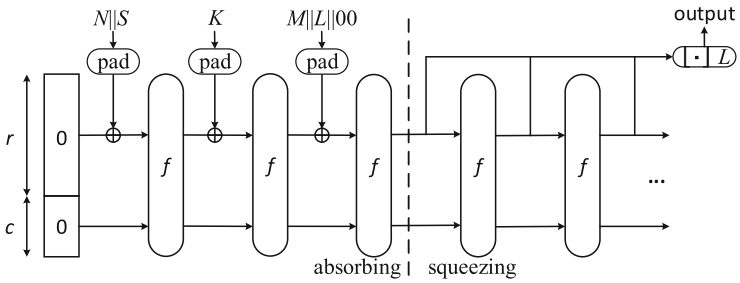

Fig. 3. `KMAC` processing one message block

two variants: `KMAC128` and `KMAC256`, based on KECCAK$[\boldsymbol{c} = 256](M, L)$ and KECCAK$[\boldsymbol{c} = 512](M, L)$, 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`.

KECCAK-MAC [4] is a KECCAK-based MAC where KECCAK directly takes the combination of a key and a message, *i.e.*, $K||M$ as input. The key size is assumed to be 128 bits.

2.3 The Duplex Construction and KEYAK, KETJE

The duplex construction [4] is closely related to the sponge construction, and is mostly used for authenticated encryption. Following variants of the duplex

construction, KEYAK and KETJE [6,7] are two KECCAK-p based authenticated encryption schemes. Figure 4(a) shows the scheme of KEYAK which employs an almost full-state keyed duplex construction [10]. It consists of 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. Note that any attack on Lake KEYAK is also applicable to the three remaining instances.

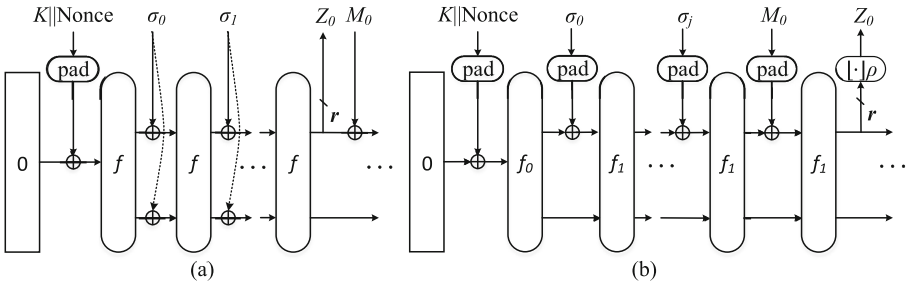

Fig. 4. (a) KEYAK and (b) KETJE, where the finalization is omitted.

Figure 4(b) displays the scheme of KETJE. 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.

Full-state Keyed Duplex (resp. Full-state Keyed Sponge) [10,21] is generalized from duplex (resp. sponge) for better efficiency by allowing full-state absorption. This idea has been applied to KEYAK which absorbs data blocks of length greater than $\boldsymbol{r}$ bits.

2.4 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. Only one-block padded messages are considered in our attacks for KMAC while there is no restriction on message length for attacks on other instances.

For describing the model, we use variables a, b, c, d in lowercase to denote states and the capital ones, namely A, B, C, D to denote their activeness, *i.e.*, a bit is active if it contains cube variables. The demension of the cube is denoted by d, and the number of conditions is denoted by t.

3 Related Works and Motivations

3.1 Cube Attacks

The cube attack, a variant of higher order differential attacks, was introduced by Dinur and Shamir [14] in 2009. It considers the output bit of a cipher as an unknown Boolean polynomial $f(k_0, \cdots, k_{n-1}, v_0, \cdots, v_{m-1})$ where $k_0, \cdots, k_{n-1}$ are secret input variables and $v_0, \cdots, v_{m-1}$ are public input variables. Given a monomial $t_I = v_{i_1} \cdots v_{i_d}$, $I = \{i_1, \cdots, i_d\}$ $(d \leq m)$, 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:

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

where p_{S_I} is called the superpoly of I in f. The basic idea of cube attacks and cube testers [1] is that the sum of the outputs over the cube which contains all possible values for $v_{i_1}, \cdots, v_{i_d}$ (called *cube variables*) 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 variables, and cube testers aim to distinguish p_{S_I} from a random function, *e.g.*, $p_{S_I} = 0$.

In [13], 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 [19], Huang *et al.* developed conditional cube testers for the 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 recall the definition of conditional cube variables and a theorem from [19] below.

Definition 1 ([19]). *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 ([19]). *For* $(n+2)$*-round* Keccak *sponge function* $(n > 0)$*, if there are* p $(0 \leq p < 2^n + 1)$ *conditional cube variables* $v_0, \cdots, v_{p-1}$*, and* $q = 2^{n+1} - 2p + 1$ *ordinary cube variables,* $u_0, \cdots, u_{q-1}$ *(If* $q = 0$*, we set* $p = 2^n + 1$*), then the term* $v_0 v_1 \cdots v_{p-1} u_0 \cdots u_{q-1}$ *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 [19]. Later, the attacks on Keccak-MAC were further improved with better conditional cubes found by an MILP model in [20].

Attack Procedure. In previous works [19,20], the number of conditional cube variables is chosen to be 1, *i.e.*, $p = 1$. Then, over a conditional cube with dimension $d = 2^n$, the cube sum is zero for $(n+1)$-round Keccak sponge function if the conditions are satisfied. For such a conditional cube whose conditions involve t-bit secret information, the $(n+1)$-round attack proceeds in two steps as follows.

1. Guess the t-bit secret information and set the conditions accordingly.
2. Query the $2^d = 2^{2^n}$ outputs and calculate the cube sum. If the cube sum is zero, mark the guess as a candidate for the t-bit secret information.

The attack has a time and data complexity of $2^{d+t} = 2^{2^n+t}$. If t is far less than the output length, the t-bit secret information can be recovered uniquely. There may exist conditions that do not involve any secret information, but only conditions involving secret information affect the complexities. In the following, t is referred to the number of bits of secret information in conditions.

3.3 Linear Structures

In [17], Guo *et al.* 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 in λ with diffusion property, so λ does not diffuse the variables under this setting. Figure 5 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 with orange dots 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 adjacent 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 of variables through χ, while an unknown neighboring constant diffuses the variable in an uncertain way, as denoted by lanes with orange dots where the variable has an uncertain coefficient. This structure has degrees of freedom 512. Also, it can be regarded as a cube of dimension 512 that linearizes the first round.

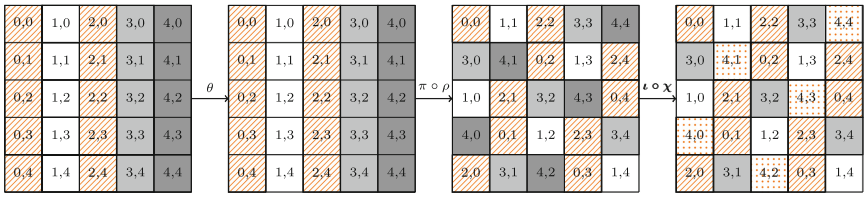

Fig. 5. 1-round linear structure of Keccak-p with the degrees of freedom up to 512, with bits in orange slashes (resp. dots) of degree 1 (resp. at most 1), and gray, light gray and white bits being values 1, 0, and arbitrary unknown constants, respectively.

3.4 Motivations

Through the linear structure, the diffusion effect of variables through χ is illustrated, which enables us to give a full description of χ using MILP. Then we consider the possibility of building a new MILP model for searching conditional cube attacks for Keccak-p based constructions, especially for finding optimal conditional cubes for constructions with fully unknown internal state.

Impact of p. If the number of conditional cube variables p increases by 1, the dimension d of the required cube reduces by 1 but t increases by at least 1. So there is no need to have more than one conditional cube variable for most cases. Therefore, we set $p = 1$ in our attacks on `KMAC`, Keyak and Ketje.

However, multiple conditional cube variables may be useful for analyzing Full-state Keyed Sponge (FKS) or Full-state Keyed Duplex (FKD) [10,21] where full-state message absorption is used. Due to full-state degrees of freedom, a large number of conditional cube variables may exist and even without any condition. The following table shows the comparison between two extreme cases where $p = 1$ and $p = 2^n + 1$, latter of which means all cube variables are conditional cube variables and thus the first two rounds are fully linearized. If p is large

p	Dimension	n_r rounds with zero sum
1	2^n	$n+1$
2^n+1	2^n+1	$n+2$

enough and $2^{d+t} = 2^{2^{n+1}-p+1+t} < 2^{|K|}$, the cube attack can be extended by one round.

For clarity, we define two types of conditional cubes as follows.

Type I. Among all cube variables, there is only one conditional cube variable.
Type II. All cube variables are conditional cube variables, *i.e.*, all the cube variables do not multiply with each other in the first two rounds.

In [7], it is stated that whether cube attacks can be extended to more rounds by exploiting full-state absorbing remains an open question. In this paper, we try to answer the open question by exploiting Type II cubes.

4 Modeling Each Step with MILP

MIL 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.

Cryptanalysis using MILP takes five main steps as shown in Fig. 6. Firstly, one defines variables which are mostly binary for the cryptanalytical problem. Secondly, she identifies links between the variables, which deeply depend on the cryptanalytical problem. Based on the links, she then generates all valid patterns for the variables which can be described with inequalities, using existing methods. In this paper, we use the convex hull method [28] together with a selection algorithm from [24]. Once the cryptanalytical problem is converted to an MILP problem, it can be solved with an MILP solver. Cryptanalysis using other tools such as SAT solvers works in a similar way. Since the last two steps are straightforward, the first three steps are the core part for MILP-based cryptanalysis which will be our focus in Sects. 4 and 5.

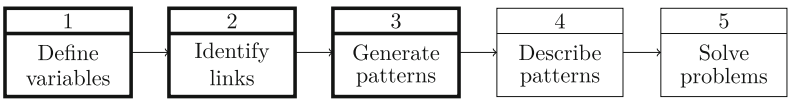

Fig. 6. Workflow of cryptanalysis using MILP

In this section, through a 1-round linear structure of `KMAC` we first show where the conditions come from, and formulate the time complexity of conditional cube attacks. Then we describe each step of the Keccak-p round function using

inequalities. The full model for searching conditional cubes will be introduced in the next section. Note that our modeling is described under the assumption that the internal state of the target constructions is fully unknown. The difference of the model for constructions with partially known internal state will be discussed in Sect. 5.3.

4.1 A 1-Round Linear Structure of KMAC

Suppose the internal state before injecting messages is denoted by $k[x][y]$, $0 \leq x, y < 5$. For convenience, the $\boldsymbol{r}$-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 in Sect. 3.3, lanes with orange slashes denote variables, lanes with orange dots 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 Fig. 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 has a degree of freedom up to 384. A similar 1-round linear structure can also be constructed for KMAC256.

As can be seen, the first round can be linearized without any condition on constants by just excluding neighbouring variables before χ. Let us consider constructing a conditional cube, where 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 of the 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.

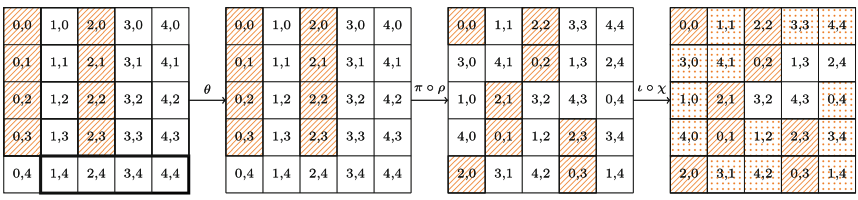

Fig. 7. 1-round linear structure of KMAC128 with the degrees of freedom up to 384, with bits in orange slashes (resp. dots) of degree 1 (resp. at most 1), and white bits being arbitrary unknown constants, respectively.

Unlike the linear structure proposed in [17], all the constants before χ of the first round are not controllable because of the unknown initial state. 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 conditional cubes fulfilling the requirement that there is no multiplication (a.k.a. AND operation) with any conditional cube variables in the second round. However, if part of constants meets certain conditions, then it can be guaranteed that the conditional cube variable do not multiply with any variable in the second round and thus conditional cubes can be constructed. This is where bit conditions come from for conditional cubes.

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. Hence the overall complexity to recover the internal state is around $\lceil \frac{|b|}{t} \rceil \cdot 2^{2^n+t}$. 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 the 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

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.

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, V and H are used for the modeling. 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 active 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 to $H[x][y][z]$. The number of bit conditions is denoted by t.

Note, we are to model two layers 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 it is practical to solve.

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, $c[x][y][z] = b[x][y][z] \oplus (1 \oplus b[x+1][y][z]) \cdot b[x+2][y][z]$. 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 $B[x][y][z] + B[x+1][y][z] \leq 1$. However, the value of input constants influences the diffusion of variables through χ and further influences the second round, as shown in Fig. 5. 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. To make it clear, we explain some rows of Table 3. The first two rows mean that if both $b[x+1][y][z]$ and $b[x+2][y][z]$ are constants, then the constants can be any value and $c[x][y][z]$ will inherit the same activeness from $b[x][y][z]$. The third row means that if $b[x+2][y][z]$ is active and $b[x][y][z], b[x+1][y][z]$ are constants but the value of $b[x+1][y][z]$ is uncertain, then $c[x][y][y]$ contains uncertain propagation from $b[x+2][y][z]$ and its algebraic degree is at most 1. On the contrary, if the value of $b[x+1][y][z]$ is restricted to 1 (resp. 0) as in the fourth (resp. fifth) row, $c[x][y][z]$ turns to be inactive (active) definitely. The fifth row can be ignored since it costs a bit condition but still diffuses the variable from $b[x+2][x][y]$ to $c[x][y][z]$, making the second round denser. The remaining rows can be explained similarly. Next, we generate a set of inequalities (see Table 7 in Appendix B) to describe these 0–1 patterns.

Table 3. Diffusion of variables through χ, where coordinates $[y][z]$s are omitted and symbol '*' denotes arbitrary value.

$B[x]$	$B[x+1]$	$B[x+2]$	$V[x+1]$	$V[x+2]$	$H[x+1]$	$H[x+2]$	$C[x]$
0	0	0	*	*	*	*	0
1	0	0	*	*	*	*	1
0	0	1	0	0	*	*	1
0	0	1	1	0	1	*	0
0	0	1	1	0	0	*	1[a]
0	1	0	0	0	*	*	1
0	1	0	0	1	*	0	0
0	1	0	0	1	*	1	1
1	0	1	0	0	*	*	1
1	0	1	1	0	*	*	1

[a] This row can be excluded

4.3 Modeling the Linear Layer

The linear layer λ consists of three steps: θ, ρ and π, the latter two of which just change the positions of the state bits. Hence, we focus on modeling θ. θ adds two columns to a bit. If both columns have even parity, then the bit does not change at all after θ. If all columns have even parity, then it is said that the state is in the column parity kernel (CP-kernel). While the original column parity is defined on values, in the context of cube attacks, it refers to activeness.

Following [25], we introduce $F[x][z]$ and $G[x][z]$ to describe the parity of a column in the state.

- The column is not active, *i.e.*, there is no variable: $G[x][z] = 0, F[x][z] = 0$;
- The column is active and the column sum is active: $G[x][z] = 1, F[x][z] = 0$;
- The column is active and the column sum is inactive: $G[x][z] = 0, F[x][z] = 1$;

As can be seen, $G[x][z] = 1$ indicates that the column sum contains variables, and only constants otherwise. If $G[x][z] = 0$ for all columns, then the cube lies in the CP-kernel. $F[x][z] = 1$ means that the column contains variables but the variables sum to certain constant, by consuming one bit degree of freedom. Suppose $A[x][y][z], y = 0, \cdots, 4$ stands for the activeness of column (x, z), then the patterns of $A[x][y][z], y = 0, \cdots, 4$ and $F[x][z]$, $G[x][z]$ fall into a set of $1 + 5 + (32 - 6) \times 2 = 58$ discrete points in $\mathbb{R}^7$. The inequalities model this set are derived and listed in Table 8.

The activeness of the output of θ now can be calculated from $A[x][y][z]$ and $G[x][z]$. Assume $B[x][y][z]$ denotes the activeness of θ's output (elsewhere $B[x][y][z]$ denotes the activeness of the output of the linear layer). Then $B[x][y][z] = 1$ if any of $A[x][y][z], G[x-1][z]$ and $G[x+1][z-1]$ is 1; otherwise $B[x][y][z] = 0$. This can be modeled by the following inequalities.

$$\begin{aligned} B[x][y][z] - A[x][y][z] \geq 0, \\ B[x][y][z] - G[x-1][z] \geq 0, B[x][y][z] - G[x+1][z-1] \geq 0, \\ A[x][y][z] + G[x-1][z] + G[x+1][z-1] - B[x][y][z] \geq 0. \end{aligned} \tag{1}$$

If only cubes in the CP-kernel are of interest, set $G[x][z] = 0$ and inequalities in (1) can be replaced with $B[x][y][z] = A[x][y][z]$. In this way, the model of the linear layer is simplified.

5 Modeling the Search for Conditional Cubes

This section presents a full model for searching conditional cubes of both types. The conditional cube requires conditional cube variables not to multiply with any variable even in the second round, which means their neighboring bits before the second χ should be constants. For the Type I, we could fix the positions of the conditional cube variable (we place the same variable at two bit positions in the same column of the initial state) and focus only on it and its neighboring bits. Whereas for Type II, attention should be paid to the diffusion of all variables in the second round. Due to this difference for the second round, our model for searching conditional cubes of both types will be constructed separately.

5.1 Model for Searching Conditional Cubes of Type I

Modeling the Second Round. The neighboring bits of the conditional cube variable before the second χ should be constants. Suppose these neighboring bits are denoted by s_i. According to the round function R, each neighboring 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 one bit degree of freedom.

We introduce one more dummy variable S_i for s_i to indicate which case happens, where $S_i = 0$ for Case 1 and $S_i = 1$ for Case 2. Case 1 is simple, while for Case 2 one needs to pay attention to "uncertain propagations" or lanes with orange dots in Fig. 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 propagations.

Similarly, all possible patterns of S_i and its related bits can be enumerated as shown in Table 4 and the set of inequalities are provided in Table 9. Specifically, if $c[x][y][z]$ is required in calculating s_i, the inequalities in Table 9 are added to the MILP model.

Table 4. Influence of conditional cube variables in the second round. Symbol '*' denotes arbitrary value.

S_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

Modeling the Search for Conditional Cubes. The following constraints are generated for searching conditional cubes of Type I.

1. Constraints for the linear layer of the first round, according to Sect. 4.3;
2. Constraints for the nonlinear layer of the first round, according to Table 7;
3. Constraints for the conditional cube variable in the first round. If a input bit $b[x][y][z]$ of χ involves the *conditional* cube variable, then we fix its neighboring bits to constants such that it does not diffuse to other positions. It requires

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

4. Constraints for the conditional cube variable in the second round, according to Table 9;
5. Constraint for the dimension. If a 2^n-dimensional conditional cube is required, then set

$$\sum A[x][y][z] - \sum F[x][z] - \sum S_i = 2^n, \tag{3}$$

   where $\sum F[x][z] + \sum S_i$ is the number of consumed degrees of freedom.
6. Objective. The objective is to minimize bit conditions. That is

$$\text{Minimize}: \quad \sum V[x][y][z]. \tag{4}$$

Besides, there may exist additional constraints. For example, the last $\boldsymbol{c}$ 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.

5.2 Model for Searching Conditional Cubes of Type II

Modeling the Second Round. For Type II conditional cubes, all the cube variables should not multiply with each other in the second round. Therefore the diffusion of each cube variable in the second round becomes indispensable and must be modeled. Beside the activeness of the input of the second round, the diffusion of cube variables also depends on the activeness of column sums which is the core part to be modeled.

Recall that we start from b, the input of χ in the first round and $c = \chi(b)$. Let $d = \lambda(c)$ by omitting the ι step of the first round, and $D[x][y][z]$ denotes the activeness of d. From the algebraic expression of χ, namely, $c[x][y][z] = b[x][y][z] \oplus (1 \oplus b[x+1][y][z]) \cdot b[x+2][y][z]$, it is known that if $B[x][y][z] = 1$, then $C[x][y][z] = 1$. If the sum of column (x, z) of b is inactive, then in what circumstance the sum of column (x, z) of c is also active? This is what we need to explore. Note that, columns with an inactive sum do not diffuse to other columns, which is beneficial to the linearization of the second round.

Suppose $G_1[x][z] = 1$ means the sum of column (x, z) in b is active and $G_1[x][z = 0]$ otherwise. Let $G_2[x][z]$ play the same role for c. With $G_2[x][z]$ and $C[x][y][z]$, the linear layer in the second round can be modeled just as the linear layer in the first round. To make the second round linear, we only need to add the constraint $D[x][y][z] + D[x+1][y][z] \leq 1$. So the only problem unsolved is to model the activeness of columns of c.

The value of $G_2[x][z]$ is influenced by three columns of b at $(x, z), (x+1, z)$ and $(x+1, z)$. This is the most complex relation to be modeled in this paper. Specifically, variables at position (x, y, z) of b propagate to position (x, y, z) of c for sure; variables at positions $(x+1, y, z)$ and $(x+2, y, z)$ of b may diffuse to position (x, y, z) of c. The sum of column (x, z) of c is inactive, *i.e.*, $G_2[x][z] = 0$ only if all the following three conditions hold.

- $G_1[x][z] = 0$.
- No variable in column $(x+1)$ of b propagates to column (x, z) of c.
- (a) No variable in column $(x+2)$ of b propagates to column (x, z) of c, or (b) all the variables in column $(x+2)$ of b propagate to column (x, z) of c and $G_1[x+2][z] = 0$[1].

In the following, the three conditions will be analyzed in detail individually.

1. The effect of variables in column (x, z). $C[x][z] = 1$ if $B[x][z] = 1$, so $G_2[x][z] = 1$ if $G_1[x][z] = 1$.
2. The effect of variables in column $(x+1, z)$ of b depends on conditions in column $(x+2, z)$. If there is any uncertain propagation of variables from column $(x+1, z)$, $G[x][z] = 1$. Additionally, $P[x][y][z]$ is introduced where $P[x][y][z] = 1$ if the variable at $(x+1, y, z)$ is propagated to (x, y, z) with an uncertain coefficient and $P[x][y][z] = 0$ otherwise. The relation of $P[x][y][z]$ and $B[x+1][y][z], V[x+2][y][z]$ is described in the following table.

$P[x]$	$B[x+1]$	$V[x+2]$	inequalities
0	0	*	$-P[x] + B[x+1] \geq 0$
1	1	0	$-P[x] - V[x+2] \geq -1$
0	1	1	$P[x] - B[x+1] + V[x+2] \geq 0$

The effect of column $(x+1, z)$ to column (x, z) is denoted by $M[x][z]$ where $M[x][z] = 1$, *i.e.*, there exist uncertain propagations of variables from column $(x+1, z)$ if any $P[x][y][z], y = 0, \cdots, 4$ is 1. This can be described with inequalities in (5).

$$\begin{aligned} M[x][z] - P[x][y][z] &\geq 0, y = 0, \cdots, 4. \\ \sum_y P[x][y][z] - M[x][z] &\geq 0. \end{aligned} \tag{5}$$

3. The effect of variables in column $(x+2, z)$ of b is relatively complicated. As shown previously, there are two cases that column $(x+2, z)$ of b does not affect $G_2[x][z]$. To identify these two cases, we introduce $Q_1[x][y][z], Q_2[x][y][z]$, $N_1[x][z], N_2[x][z]$ and $N_3[x][z]$. $Q_1[x][y][z]$ and $N_1[x][z]$ play similar roles as $P[x][y][z]$ and $M[x][z]$, *i.e.*, $N_1[x][z] = 1$ if there is uncertain propagation from column $(x+2, z)$.
$Q_2[x][y][z] = 1$ if the variable at $(x+2, y, z)$ of b is propagated to (x, y, z) of

[1] The reason why the modeling for the effects of column $(x+2, z)$ and column $(x+1, z)$ are different lies in the following fact. If the constant on the right side of a cube variable consumes a condition, we can constrained the constant to 0 directly, since 1 is worse under all circumstance as shown in Table 3. On the contrary, if the condition is imposed to the constant on the left side of a cube variable, the constant can be restricted to either 0 or 1 and no one has an absolute advantage over the other.

c for sure. Let $N_2[x][z] = 0$ if and only if $\sum_y Q_2[x][y][z] = 0$. Let $N_3[x][z] = 0$ if $\sum_y Q_2[x][y][z] = \sum_y B[x+2][y][z]$, *i.e.*, all variables in column $(x+2, z)$ of b are diffused to column (x, z) of c.
$Q_1[x][y][z]$ and $Q_2[x][y][z]$ can be modeled as shown in the following table.

$Q_1[x]$	$Q_2[x]$	$B[x+2]$	$V[x+1]$	$H[x+1]$	Inequalities
0	0	0	0	*	$-Q_1[x] - Q_2[x] + B[x+2] \geq 0$
0	0	0	1	*	$Q_1[x] - B[x+2] + V[x+1] \geq 0$
1	0	1	0	*	$-Q_1[x] - V[x+1] \geq -1$
0	1	1	1	0	$Q_1[x] + Q_2[x] - B[x+2] + H[x+1] \geq 0$
0	0	1	1	1	$-Q_2[x] - H[x+1] \geq -1$

The relation between $N_1[x][z], N_2[x][z]$ and $Q_1[x][y][z], Q_2[x][y][z]$ can also be described in the same way as in (5). To model $N_3[x][z]$, a large integer I is used to express the IF-ELSE logic that $N_3[x][z] = 0$ if $\sum_y Q_2[x][y][z] = \sum_y B[x+2][y][z]$ as long as I is larger than 5, say 100. The exact inequalities are shown in (6).

$$\begin{aligned} \sum_y Q_2[x][y][z] - \sum_y B[x+2][y][z] + I \cdot N_3[x][z] &\leq I - 1, \\ \sum_y Q_2[x][y][z] - \sum_y B[x+2][y][z] + I \cdot N_3[x][z] &\geq 0. \end{aligned} \tag{6}$$

According to our model, $(N_1[x][z], N_2[x][z], N_3[x][z]) = (0, 0, *)$ indicates the first case, and $(N_1[x][z], N_2[x][z], N_3[x][z]) = (0, 1, 0)$ stands for the second case.

As can be derived from the above analysis, when (a) $(M[x][z], N_1[x][z], N_2[x][z], N_3[x][z]) = (0, 0, 0, *)$, or (b) $(M[x][z], N_1[x][z], N_2[x][z], N_3[x][z]) = (0, 0, 1, 0)$, and $G_1[x+2][z] = 0$, $G_2[x][z] = G_1[x][z]$; otherwise $G_2[x][z]$ is 1. The inequalities in Table 10 can be used to model this property.

Modeling the Search for Conditional Cubes. After introducing special techniques for modeling the column parity of the state in the second round, we can build the whole model for searching conditional cubes that linearize the first two rounds. Note that we start from the input of χ in the first round.

1. Describe the column parity of b using $G_1[x][z], F_1[x][z]$, according to Table 8.
2. Constraints for χ in the first round, according to Sect. 4.2;
3. Constraints for modeling the column parity of c, according to this subsection.
4. Constraints for the linear layer in the second round, according to Sect. 4.3;
5. Constraints for χ in the second round, *i.e.*, $D[x][y][z] + D[x+1][y][z] \leq 1$.

6. Constraint for the dimension. If a $(2^n + 1)$-dimensional conditional cube is required, then set

$$\sum C[x][y][z] - \sum F_1[x][z] = 2^n + 1, \tag{7}$$

where $\sum F_1[x][z]$ is the number of consumed degrees of freedom.
7. Objective. The objective is to minimize bit conditions. That is

$$\text{Minimize}: \quad \sum V[x][y][z]. \tag{8}$$

5.3 Discussion and Comparison

Model for Constructions with Partially Known Internal State. While minimal conditions means optimal conditional cubes for Keccak-p-based construction with fully unknown internal state, such as KMAC, it is not the case if the internal state is partially known even though the number of conditions involving the key is still minimized. Note that the conditions are imposed on certain input bits of the first χ and each bit involves some key information. For Keccak-p-based construction with partially unknown internal state, t bit conditions do not necessarily contain t-bit key information. For example, in the 64-dimensional cube of Ketje SR v1, there are 27 bit conditions all of which involve the key but contain only 26-bit information of the key due to dependency.

Comparison with the Existing MILP Model. Recently, Li *et al.* proposed an MILP model for searching cubes of Type I [20]. Their model sets every $b[x][y][z]$ to a constant if it relates to the neighboring bits of the conditional variable in the first two rounds. 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 neighboring bits of the conditional variable in the second round. As a result, more conditional cubes can be found with a greater range of dimension. As demonstrated in Table 5, better conditional cubes are found using our model under the same setting. In particular, given the dimension, our model returns conditional cubes with much fewer bit conditions. For example, the 32-dimensional conditional cube of Keccak-MAC-512 in [20] requires 24 bit conditions involving the key, while using our model, the number of bit conditions can be only 3 ($n = 5$ and $t = 3$), which reduces the time complexity of attacking 6-round Keccak-MAC-512 from $2^{58.3}$ [20] to $\lceil \frac{|k|}{t} \rceil \cdot 2^{2^n+t} = \lceil \frac{128}{3} \rceil \cdot 2^{2^5+3} \approx 2^{40}$. Our cube of Keccak-MAC-512 is provided in Table 5. Moreover, our models cover both types of conditional cubes while Li *et al.*'s model aims for only Type I conditional cubes.

6 Applications

In this section, we apply our models to conditional cubes attacks on KMAC, Keyak and Ketje where Type I cubes are used. In order to extend the cube attacks on Keccak-p based constructions with full-state absorption, we exploit Type II cubes.

Table 5. 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	[20]
	97	8	This
	65	**2**	
Keccak-MAC-512	32	24	[20]
	32	**3**	This
	50	24	

6.1 Conditional Cube Attacks on KMAC

In this subsection, techniques described in Sects. 4 and 5.1 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.

Cube Attack on KMAC128. For KMAC128, the capacity is 256, which covers only four lanes. By placing the conditional cube variable at two bits in a column of a[2], 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 shown in Table 6. This cube is selected from the CP-kernel and has dimension 16, and the conditional cube variable is placed at $a[0][0][0], a[0][1][0]$. 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. 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 remaining 15 ordinary cube variables can be extracted from $A[x][y][z]$, $0 \leq x, y < 5, 0 \leq z < 64$ 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.

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. Once the internal state is recovered, the key can be derived directly.

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

[2] There is an exception that no conditional cube can be found when the conditional variable is placed in lanes $(1, 0), (1, 1)$.

Table 6. A conditional cube of KMAC in the CP-kernel. Positions of cube variables are derived from a 5×5 array of lanes in hexadecimal using the little-endian format where '0' is replaced with '−'.

Positions of cube variables				
4----------2---1|	----------|	----------1|	----------|	----------
66------41-28-11|	----------|	---------1-----1|	----------|	----------
26------414-8-1-|	----------|	---------1------|	----------|	----------
24--------4---1-|	----------|	----------|	----------|	----------
----------|	----------|	----------|	----------|	----------
The conditional cube variable: $a[0][0][0] = a[0][1][0] = v_0$				
Ordinary cube variables				
$a[0][1][4] = v_1$,	$a[0][1][24] = a[0][2][24] = v_6$,	$a[0][1][61] = v_{11}$,		
$a[0][2][4] = v_2$,	$a[0][1][30] = a[0][2][30] = v_7$,	$a[0][2][61] = v_{12}$,		
$a[0][3][4] = v_1 + v_2$,	$a[0][1][57] = a[0][2][57] = v_8$,	$a[0][3][61] = v_{11} + v_{12}$,		
$a[0][1][15] = a[0][2][15] = v_3$,	$a[0][1][58] = v_9$,	$a[0][0][62] = a[0][1][62] = v_{13}$,		
$a[0][0][17] = a[0][1][17] = v_4$,	$a[0][2][58] = v_{10}$,	$a[2][0][0] = a[2][1][0] = v_{14}$,		
$a[0][2][22] = a[0][3][22] = v_5$,	$a[0][3][58] = v_9 + v_{10}$,	$a[2][1][24] = a[2][2][24] = v_{15}$.		
Conditions				
$b[0][3][36] = k'[0][3][36] + 1$, $b[2][3][36] = k'[2][3][36]$,				
$b[4][0][0] = k'[4][0][0] + 1$, $b[1][0][0] = k'[1][0][0]$.				

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 then 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-p, 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 $\boldsymbol{r}$-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 [26], and the order of the lanes recovered are displayed in Fig. 8. The total time complexity of recovering the whole internal state is $2^6 \cdot 2^{64}(1 \cdot 2^4 + 3 \cdot 2^3 + 6 \cdot 2^2) = 2^{76}$.

Cube Attack on KMAC256. KMAC256 has a capacity of 512 bits 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

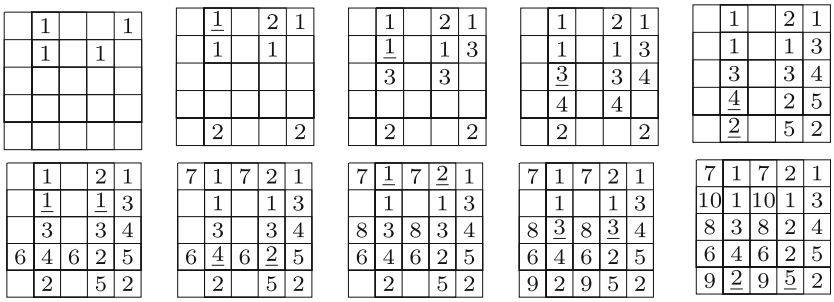

Fig. 8. The lanes recovered using ten z-axis equivalent conditional cubes. The underline means bits of these lanes are involved in conditions but they are already known.

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 the algebraic degree. As a result, 9 rounds of KMAC256 can be attacked.

Choice of the Conditional Cube Variable. When we place the conditional cube variable at two bit positions of the same column in a, the obtained cubes generally 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 CP-kernel so that it does not diffuse even in the second round, leading to a small set of constraints for the conditional cube variable. As studied in [11], 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 at two bit positions of a column of $a[x][y][z]$, $0 \le y < 3$ can then be exploited to recover the remaining lanes.

To recover the whole internal state, three z-axis equivalent conditional cubes as shown in [26] are used and lanes recovered in each cube are displayed in Fig. 9. As can be learned from the figure, the time complexity of the internal state recovery is $2^6 \cdot 2^{128}(2^{12} + 2^{11} + 2^3) = 2^{146.58}$ calls of 9-round KMAC256.

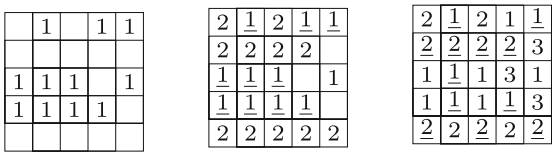

Fig. 9. The lanes recovered using three z-axis equivalent conditional cubes. The underline means bits of these lanes are involved in conditions but they are already known.

6.2 Conditional Cube Attacks on KEYAK and KETJE

This subsection considers conditional cube attacks of KEYAK and KETJE under the nonce respect setting, *i.e.*, the cube variables are placed among the positions where the nonce is loaded, and suppose there is no associated data to be processed.

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 blue positions denote padded or encoded bits and white positions are the nonce. This means that the cube variable should be placed in white lanes. Unlike `KMAC`, the internal state of both KEYAK and KETJE is known except the key part. Due to the dependence of key bits in conditions, our model may not guarantee optimal solutions.

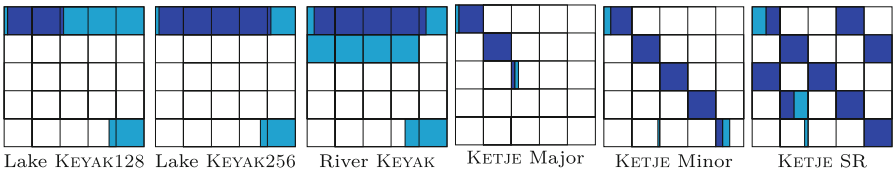

Fig. 10. Key pack of KEYAK and KETJE where the blue part means the key, the light blue part denotes padded or encoded bits and the white part is the nonce.

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 [26]), 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 [26]), 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 [26], these 12 bit conditions involve 11 bits key information), the key recovery attack of 8-round River KEYAK costs a data and time complexities $2^{11} \cdot 2^{64} + 2^{10} \cdot 2^{64} \cdot 6 + 2^{128-71} = 2^{77.00}$.

KETJE **Major.** Using a 64-dimensional cube with 3 bit conditions involving the key (see [26]), 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 [26]), 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 better attacks than the existing ones in [15]. 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 [26], involve 26 bits key information) and the time and data complexities are $2^{26} \cdot 2^{64} \cdot 2 + 2^{128-54} = 2^{91.00}$. Therefore, KETJE instances using KECCAK-$p^\star$ are stronger than those instances using KECCAK-p under our attacks.

6.3 Conditional Cube Attacks on Full-State Keyed Duplex

In this subsection, we consider conditional cube attacks on KECCAK-p based FKD (or FKS) which provides full-state degrees of freedom. We assume that the first data block is absorbed after the application of the underlying permutation, as in KEYAK. Therefore, the internal state before injecting the first data block is fully unknown. This is not a nonce-respected attack since the cube will be constructed on the full-state data block.

For convenience, FKD with KECCAK-$p[\boldsymbol{b}, n_r]$ as the underlying permutation is denoted by FKD[$\boldsymbol{b}$]. A direct application of linear structures shows that 512-dimensional Type II cubes for FKD[1600] can be constructed by constraining 960 bits to certain constants. However, in key/state recovery attacks the number of bit conditions allowed is limited. In this subsection, we apply our model for searching Type II cubes of FKD[$\boldsymbol{b}$], and try to find some useful cubes with a small number of bit conditions.

When the number of bit conditions is set to 0, Type II cubes of FKD[1600] can be found with dimension at least 48. If the dimension is set to 65, a Type II cube with 25 bit conditions is found, as shown in [26]. Since the first two rounds are linearized, the cube sum of 8-round KECCAK is zero. Thus, this cube can be used to attack 8-round FKD[1600] by recovering the internal state in a similar way to the attack on `KMAC`. As long as the rate $\boldsymbol{r}$ is greater than 320 bits, a 9-round attack of FKD[1600] can be achieved by partially reversing the last round. The time complexity is about $2^{65+25} = 2^{90}$. For more experimental results, please refer to Appendix A.

Compared with cube attacks on KECCAK-p based constructions where $\boldsymbol{r}$-bit messages are absorbed, cube attacks on FKD[1600] can be extended to one

more round by exploiting the full-state absorption. With this, the open question proposed by the KEYAK designers in [7] now is answered.

The idea of full-state absorption has already been applied to KEYAK which absorbs data blocks of more than $\boldsymbol{r}$ bits each but less than $\boldsymbol{b}$ bits. For example, Lake KEYAK processes data blocks of 1536 bits, less than 1600 bits. A simple way to adapt our attack on 9-round FKD[1600] to Lake KEYAK is to find a Type II cube with dimension 129 (65+64). However, such a cube with increased dimension could not be found in a practical amount of time. Therefore, the extended attack does not apply to Lake KEYAK.

6.4 Experimental Verification and Codes

Since the attacks in this paper are impractical with current computation power, the correctness of the attacks is verified on 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 Type I conditional cube attacks: one based on the 16-dimensional toy cube in Table 6 for fast verification, and the other based on a 32-dimensional cube for attacking 7-round `KMAC256` (or 6-round `KMAC128`). A conditional cube attack on 7-round FKD[1600] is also implemented with a 32-dimensional cube of Type II. Note that this cube has three bit conditions which are set intentionally; otherwise, there can be no condition. The correctness of our attacks are confirmed by these three experiments. The source codes for experimental verification are available via http://team.crypto.sg/VerificationCodesConCube.zip. The codes for building our models are available through http://team.crypto.sg/modelConCube.zip.

7 Conclusions

In the paper, we proposed new MILP models for searching two types of 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 Type I conditional cubes into account and we proposed a model for searching Type II conditional cube for the first time. With the new models, conditional cubes with desired dimensions and least bit conditions were found for `KMAC`. As a result, key recovery attacks of 7-round `KMAC128`, 9-round `KMAC256` can be mounted respectively. To the best of our knowledge, these are the first cryptanalysis results against `KMAC`. Using our model, we solve the open question of FKD by extending the conditional cube attack by one additional round. The application of our model to KEYAK and KETJE gives rise to new attacks or better attacks with reduced complexities. Specifically, the number of rounds attacked against Lake KEYAK with 128-bit keys is improved from 6 to 8 in the nonce-respected setting and 9 rounds of Lake KEYAK can be attacked when using 256-bit keys; attack complexities are reduced generally on other constructions.

Acknowledgement. Ling Song and Danping Shi are partially supported by the Fundamental Theory and Cutting Edge Technology Research Program of Institute of Information Engineering, CAS (Grant No. Y7Z0251103), Youth Innovation Promotion Association CAS, the National Natural Science Foundation of China (Grants No. 61802399, 61802400, 61732021, 61772519 and 61472415) and Chinese Major Program of National Cryptography Development Foundation (Grant No. MMJJ20180102).

A Experimental Details

The model for searching Type II cubes for FKD[1600] has 37440 inequalities on 15040 variables, which is about 1.8 times of the model for searching Type I conditional cubes. Even though the search for Type I conditional cubes takes seconds or minutes, the solving time of the model for Type II cubes increases exponentially. We solve the model for finding Type II conditional cubes with Gurobi optimizer [18] on a server with 64 cores at 2.3 GHz, and Gurobi could not finish the optimization in a practical amount of time.

Type II cubes for FKD[1600] can be found with dimension $d \geq 65$. However, for FKD[800], when we set the number of conditions $t \leq 62$ and the objective to maximize the dimension, Gurobi shows after running 8 days that the dimension falls in $[62, 94]$, but to extend the attack by one more round, a 65-dimensional Type II conditional cube is required.

B Inequalities

Table 7. Inequalities modeling the non-linear operation χ in the first round, where coordinates $[y][z]$s are omitted.

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

Table 8. Inequalities modeling the parity of a column

$$
\begin{aligned}
-F[x][z] - G[x][z] &\geq -1 \\
-A[x][0][z] + F[x][z] + G[x][z] &\geq 0 \\
-A[x][1][z] + F[x][z] + G[x][z] &\geq 0 \\
-A[x][2][z] + F[x][z] + G[x][z] &\geq 0 \\
-A[x][3][z] + F[x][z] + G[x][z] &\geq 0 \\
-A[x][4][z] + F[x][z] + G[x][z] &\geq 0 \\
A[x][0][z] + A[x][1][z] + A[x][2][z] + A[x][3][z] + A[x][4][z] - 2F[x][z] - G[x][z] &\geq 0
\end{aligned}
$$

Table 9. Inequalities modeling the non-linear operation χ in the second round

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

Table 10. Inequalities modeling the column parity of the input of the second round.

$$
\begin{aligned}
G_2[x][z] - G_1[x][z] &\geq 0 \\
G_2[x][z] - N_1[x][z] &\geq 0 \\
G_2[x][z] - M[x][z] &\geq 0 \\
-G_2[x][z] + G_1[x][z] + M[x][z] + N_1[x][z] + N_2[x][z] &\geq 0 \\
G_2[x][z] - G_1[x+2][z] - N_2[x][z] &\geq -1 \\
G_2[x][z] - N_2[x][z] - N_3[x][z] &\geq -1 \\
-G_2[x][z] + G_1[x][z] + G_1[x+2][z] + M[x][z] + N_1[x][z] + N_3[x][z] &\geq 0
\end{aligned}
$$

References

1. Aumasson, J., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In: Handschuh, H., Lucks, S., Preneel, B., Rogaway, P. (eds.) Symmetric Cryptography, 11.01. - 16.01.2009. Dagstuhl Seminar Proceedings, vol. 09031. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany (2009). http://drops.dagstuhl.de/opus/volltexte/2009/1944/

2. Bertoni, G., Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., Van Keer, R.: Farfalle: parallel permutation-based cryptography. IACR Trans. Symmetric Cryptol. **2017**(4), 1–38 (2017). https://tosc.iacr.org/index.php/ToSC/article/view/801
3. 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
4. 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.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28496-0_19
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak Reference, January 2011. http://keccak.noekeon.org, version 3.0
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR Submission: Ketje v2. Candidate of CAESAR Competition, September 2016
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR Submission: Keyak v2. Candidate of CAESAR Competition, September 2016
8. Bi, W., Dong, X., Li, Z., Zong, R., Wang, X.: Milp-aided cube-attack-like cryptanalysis on Keccak keyed modes. Designs, Codes and Cryptography, August 2018. https://doi.org/10.1007/s10623-018-0526-x
9. Chaigneau, C., Fuhr, T., Gilbert, H., Guo, J., Jean, J., Reinhard, J., Song, L.: Key-recovery attacks on full kravatte. IACR Trans. Symmetric Cryptol. **2018**(1), 5–28 (2018). https://doi.org/10.13154/tosc.v2018.i1.5-28
10. Daemen, J., Mennink, B., Van Assche, G.: Full-state keyed duplex with built-in multi-user support. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 606–637. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_21
11. Daemen, J., Van Assche, G.: Differential propagation analysis of Keccak. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 422–441. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34047-5_24
12. Dinur, I., Dunkelman, O., Shamir, A.: Improved practical attacks on round-reduced Keccak. J. Cryptol. **27**(2), 183–209 (2014). https://doi.org/10.1007/s00145-012-9142-5
13. 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.) EUROCRYPT 2015, Part I. LNCS, vol. 9056, pp. 733–761. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_28
14. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_16
15. 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
16. Fuhr, T., Naya-Plasencia, M., Rotella, Y.: State-recovery attacks on modified Ketje Jr. IACR Trans. Symmetric Cryptol. **2018**(1), 29–56 (2018). https://tosc.iacr.org/index.php/ToSC/article/view/843
17. Guo, J., Liu, M., Song, L.: Linear structures: applications to cryptanalysis of round-reduced Keccak. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 249–274. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_9
18. Gurobi: Gurobi Optimizer. http://www.gurobi.com/

19. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional cube attack on reduced-round Keccak sponge function. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 259–288. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_9
20. Li, Z., Bi, W., Dong, X., Wang, X.: Improved conditional cube attacks on Keccak keyed modes with MILP method. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 99–127. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_4
21. 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.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 465–489. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_19
22. NIST: SHA-3 COMPETITION. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html (2007–2012)
23. Qiao, K., Song, L., Liu, M., Guo, J.: New collision attacks on round-reduced Keccak. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 216–243. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56617-7_8
24. Sasaki, Y., Todo, Y.: New algorithm for modeling S-box in MILP based differential and division trail search. In: Farshim, P., Simion, E. (eds.) SecITC 2017. LNCS, vol. 10543, pp. 150–165. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69284-5_11
25. Song, L., Guo, J.: Cube-Attack-like cryptanalysis of round-reduced Keccak using MILP. To appear in IACR Trans. Symmetric Cryptol. **2018**(3) (2018). https://eprint.iacr.org/2018/810
26. Song, L., Guo, J., Shi, D., Ling, S.: New MILP Modeling: Improved Conditional Cube Attacks on Keccak-based Constructions. Cryptology ePrint Archive, Report 2017/1030 (2017). https://eprint.iacr.org/2017/1030
27. Song, L., Liao, G., Guo, J.: Non-full Sbox linearization: applications to collision attacks on round-reduced Keccak. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 428–451. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_15
28. 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.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_9
29. 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
30. The U.S. National Institute of Standards and Technology: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash. NIST Special Publication 800–185, 21 December 2016. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf