Improved Preimage Attacks on Round-Reduced Keccak-384/512

Le He[1], Xiaoen Lin[1], Hongbo Yu[1], and Jian Guo[2]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, China
{he-l17,lxe21,yuhongbo}@tsinghua.edu.cn
[2] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, Singapore
guojian@ntu.edu.sg

Abstract. This paper provides improved preimage analysis on round-reduced Keccak-384/512. Unlike low-capacity versions, Keccak-384/512 outputs from two parts of its state: an entire 320-bit plane and a 64/192-bit truncation of a second plane. Due to lack of degrees of freedom, most existing preimage analysis can only control the first 320-bit plane and achieve limited results. By thoroughly analyzing the algebraic structure of Keccak, this paper proposes a technology named "extra linear dependence", which can construct linear relations between corresponding bits from two planes. To apply the technology, this paper inherits pioneers' attack thoughts that convert output bits to linear or quadratic equations of input variables. When solving the final equation system, those linear relations can lead to extra restricting equations of output, exceeding the limit of matrix rank. As a result, the complexity of preimage attacks on 2-round and 3-round Keccak-384/512 can be decreased to $2^{39}/2^{204}$ and $2^{270}/2^{424}$ Keccak calls respectively, which are all the best known results so far. To support the theoretical analysis, this paper provides the first preimage of all '0' digest for 2-round Keccak-384, which can be obtained in one day with single core on an ordinary PC.

Keywords: Keccak · Preimage attack · Linear dependence

1 Introduction

The Keccak function, designed by Bertoni et al. [1,2], was selected as the winner of SHA-3 competition in 2012 and finally standardized in 2015 by NIST. Since Keccak was proposed in 2008, there have been kinds of security cryptanalysis from the public research community, including preimage [3,4,5], collision [6,7,8], distinguishing [9,10,11], keyed modes [12,13,14], and many other unmentioned security settings. Those advanced attack methods work well even with practical results in low-capacity Keccak: round-reduced Keccak-224/256. Yet for round-reduced Keccak-384/512, due to lack of freedom in message block setting, most methods cannot work as efficiently as they do in low-capacity versions.

In this paper, we mainly focus on preimage attacks on round-reduced Keccak-384/512 — more specifically, linear analysis. Our research is inspired by several creative works as summarized below. In 2016, Guo et al. [15] pioneered a strategy named *linear structure* in preimage attacks on round-reduced Keccak versions. Their idea is to linearize the whole state after several rounds with freedom space partially left. Yet for Keccak-384/512, their linear structures can pass through only 1 round and thus they had to adopt other advanced technologies to achieve good results, which are not required in this paper. Then in 2019, Li and Sun [16] improved the linear structure through a technology named *allocating model*: the first message block aims to generate a restricted middle state satisfying specific conditions, so that the second message block (XORed with the restricted middle state) can obtain extra freedom space in preimage searching. They applied this model merely on round-reduced Keccak-224/256, while it can also be applied on 3-round Keccak-384 (we will give a simple design in Section 3.1). Rajasree [17] made an improvement from another perspective. He noticed the number of degrees of freedom left is much less than the number of (linear) output equations. Thus he allowed non-linear parts to exist in the structure and just constructed output equations on linear parts. This idea would not enlarge the freedom space, but enlarge the space of random constants, which is also a noticeable problem of high-capacity versions. In 2021, He et al. [18] proposed a technology named *zero coefficient*. It refers to some linear-dependent bit pairs in Keccak's state. Using this technology, they successfully satisfied 173 equations with only 162 degrees of freedom, obtaining 11 linear-dependent bit pairs. This value is limited mainly because their analysis object is Keccak-224/256. For Keccak-384/512 (with two output planes), the number can be increased to lane-level.

The latest preimage analysis on round-reduced Keccak-384/512 is Liu et al.'s research [19], mainly from which we inherit the attack frameworks. They also allowed non-linear parts to exist in the structure — unlike [17], they can indeed enlarge the freedom space, but must deal with an entirely-quadratic output state. Fortunately, they found that benefiting from the algebraic structure of Keccak, the *relinearization technique* can be applied to the final equation system. This technique is only suitable for a special case of quadratic equation system, where the number of equations is much larger than the number of different quadratic terms. Using this technique, the freedom space can bring an equivalent gain in preimage searching. In other words, n degrees of freedom can bring a gain of 2^n in preimage searching, identical to the case of linear equation system. Those preimage analysis results on round-reduced Keccak-384/512 mentioned above[3] are summarized in **Table 1**.

Our contributions. This paper proposes a new technology named *extra linear dependence* to improve preimage attacks on round-reduced Keccak-384/512. The technology aims to construct linear relations between corresponding bits from

[3] It is worth mentioning that the best preimage analysis result on 2-round Keccak-384 so far is Kumar et al.'s [20] time-memory trade-offs, with time complexity 2^{89} and memory cost 2^{87}. Yet their attack frame is completely different from linear analysis.

two output planes, so that the final equation system can be solved even when the number of equations is more than the number of variables. Under this case, n degrees of freedom can bring a gain even larger than 2^n in preimage searching. To apply the technology, we inherit (and slightly modify) the latest quadratic structures in [19]. As a result, we successfully construct 128 linear-dependent bit pairs in 2-round Keccak-384/512 and 24 linear-dependent bit pairs in 3-round Keccak-384, which can decrease the searching complexity (guessing times) by $2^{64}/2^{12}$. As for 3-round Keccak-512, due to lack of controllable column sums, *extra linear dependence* can hardly be applied. Yet we still make a progress by rectifying an omission in applying the *relinearization technique* and improving the quadratic structure. To support our analysis, we firstly provide an actual preimage (matching the padding rule) of all '0' digest for 2-round Keccak-384. Comparisons between our results and previous results are displayed in **Table 1**.

Table 1. Summary of preimage analysis on round-reduced Keccak-384/512.

Variant	Guessing Times	Size[a]	Solving Time	Final Complexity	Reference
2-round Keccak-384	2^{129}	\	\	\	[15][b]
	2^{113}	\	\	\	[17][b]
	2^{93}	384	2^{11}	2^{104}	[19][c]
	2^{28}	320	2^{11}	2^{39}	Sect. 5.1
2-round Keccak-512	2^{384}	\	\	\	[15][b]
	2^{321}	\	\	\	[17][b]
	2^{258}	448	2^{12}	2^{270}	[19][c]
	2^{193}	384	2^{11}	2^{204}	Sect. 5.2
3-round Keccak-384	2^{322}	\	\	\	[15][b]
	2^{321}	\	\	\	[17][b]
	2^{271}	460	2^{12}	2^{283}	[19][c]
	2^{258}	460	2^{12}	2^{270}	Sect. 5.3
3-round Keccak-512	2^{482}	\	\	\	[15][b]
	2^{475}	\	\	\	[17][b]
	2^{440}	494	2^{12}	2^{452}	[19][c]
	2^{412}	448	2^{12}	2^{424}	Sect. 5.4
4-round Keccak-384	2^{371}	\	\	\	[17][b]
	2^{366}	175	2^{9}	2^{375}	[19][c]

[a] "Size" is the total number of variables in the equation system after applying the relinearization technique. For better comparison, we will use the same way to calculate "Size" and "Solving Time" is then estimated according to [19].
[b] The cited results refer to guess times instead of Keccak calls, which do not include the complexity of solving the final equation system.
[c] The authors made a mistake in matching Keccak's padding rule. Their results in "Guessing Times" and "Final Complexity" should all be cut down by 2^1.

Organization. This paper starts with some preliminaries and notations about Keccak in Section 2. An overview about the attack thoughts of linear analysis on round-reduced Keccak is given in Section 3. The core technology of extra linear dependence is explained in Section 4. Improved preimage attacks on 2-round and 3-round Keccak-384/512 are provided in Section 5. Conclusions are summarized in Section 6.

2 Preliminaries

This section gives the descriptions about sponge construction, Keccak-f permutation, SHA-3 standard, properties of S-box inversion, and the meanings of the notations used in this paper.

2.1 Sponge Construction

The Keccak function adopts a new iterative construction named *sponge*, which involves three parameters r, c, d and a permutation Keccak-$f[b]$ with $b = r + c$ (as depicted in **Fig. 1**). This construction processes a message in two phases — absorbing phase and squeezing phase. In absorbing phase, the message M (after padding) is split into r-bit blocks. Starting with a b-bit all '0' IV, its first r bits are XORed with the first message block, followed by an execution of Keccak-f. After all message blocks are similarly processed, it comes to the squeezing phase. In squeezing phase, the construction outputs an r-bit digest and mixes its state by executing Keccak-f, repeating until the digest length reaches d. Finally, the digest is truncated to the first d bits.

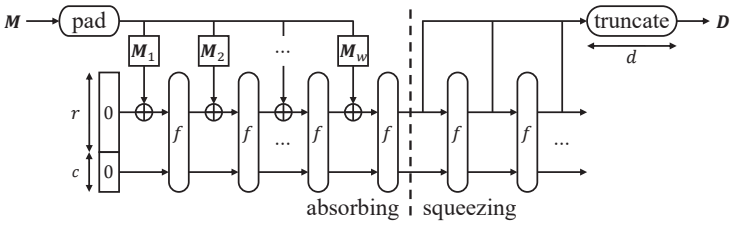

Fig. 1. The sponge construction.

2.2 Keccak-f Permutation

The core of Keccak-f calculation is its b-bit state. In [2], the designers provided seven Keccak-f permutations where $b \in \{25, 50, 100, 200, 400, 800, 1600\}$. NIST finally chose $b = 1600$ as SHA-3 standard [21]. In this paper, we also consider the case of $b = 1600$ only.

In the case of $b = 1600$, the state of Keccak-f can be regarded as 5×5 64-bit lanes (as depicted in **Fig. 2**). Each bit is denoted as $A_{x,y,z}$, where x varies from 0 to 4, y varies from 0 to 4, and z varies from 63 to 0 (counting from the most significant bit) as directed by arrows in **Fig. 2**. The r-bit part of the state piles in order of $A_{0,0,0} \sim A_{0,0,63}, A_{1,0,0} \sim A_{1,0,63}, \ldots, A_{4,0,0} \sim A_{4,0,63}, A_{0,1,0} \sim A_{0,1,63} \ldots$ Furthermore, the designers defined some components of the state (also depicted in **Fig. 2**). Among these components, a momentous one in this paper is "plane", which consists of 5 lanes or 64 rows.

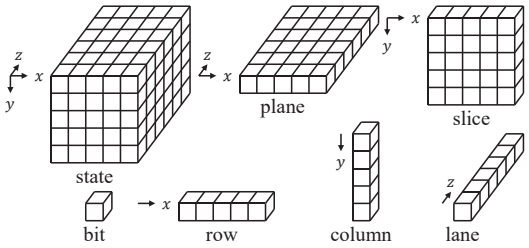

Fig. 2. The state and its components of Keccak-f.

As for the Keccak-f calculation, it consists of 24 rounds of function R, and each R consists of five steps $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$, where:

$$\begin{aligned}
\theta &: A_{x,y,z} = A_{x,y,z} \oplus \bigoplus\nolimits_{j=0\sim 4}(A_{x-1,j,z} \oplus A_{x+1,j,z-1}) \\
\rho &: A_{x,y,z} = A_{x,y,z-r_{x,y}} \\
\pi &: A_{x,y,z} = A_{x+3y,x,z} \\
\chi &: A_{x,y,z} = A_{x,y,z} \oplus (A_{x+1,y,z} \oplus 1) \cdot A_{x+2,y,z} \\
\iota &: A_{0,0,z} = A_{0,0,z} \oplus RC_z
\end{aligned} \tag{1}$$

In the formulas above, "$\oplus$" means bit-wise XOR and "$\cdot$" means bit-wise AND. Indices x and y are calculated modulo 5 and index z is calculated modulo 64. Besides, $r_{x,y}$ refers to a lane-dependent rotation constant as shown in **Table 2**. RC is a round-dependent constant. We omit the details of RC here since those constants do not affect our attack methods.

Table 2. The offsets of ρ.

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$
$y=0$	0	1	62	28	27
$y=1$	36	44	6	55	20
$y=2$	3	10	43	25	39
$y=3$	41	45	15	21	8
$y=4$	18	2	61	56	14

2.3 SHA-3 Standard

Any Keccak variant can be denoted as Keccak$[r, c, d]$ with bitrate r, capacity c and digest length d. In [21], NIST standardized four SHA-3 versions that have $r = 1600 - 2d$ and $c = 2d$, where $d \in \{224, 256, 384, 512\}$. Therefore, we can use Keccak-d or SHA-3-d to denote a SHA-3 version for short.

The only difference between Keccak-d and SHA-3-d is padding rule: Keccak pads the message by 10^*1 while SHA-3 pads the message by 0110^*1. This means for both Keccak and SHA-3, the last bit of message block M_w must be '1' and for only SHA-3, the penultimate '1' must follow "01". Therefore, matching the padding rule of Keccak[4] or SHA-3 will further increase the searching complexity by 2^1 or 2^3.

2.4 Properties of S-Box Inversion

According to Keccak-f calculation, the digest of Keccak is finally truncated from the state after the last ι step, which is just a simple constant-XOR and can be directly inversed. One step backwards, the state before the last χ step can also be partially recovered from the digest. Inversing the last χ step can effectively decrease the algebraic degree of output equations and make preimage attacks much easier.

Since step χ processes on different rows, it can be regarded as a 5-bit S-box, where input $a_0a_1a_2a_3a_4$ and output $b_0b_1b_2b_3b_4$ are calculated by (the subscript is calculated modulo 5):

$$\begin{cases} b_i = a_i \oplus (a_{i+1} \oplus 1) \cdot a_{i+2} \\ a_i = b_i \oplus (b_{i+1} \oplus 1) \cdot (b_{i+2} \oplus (b_{i+3} \oplus 1) \cdot b_{i+4}) \end{cases} \tag{2}$$

Most properties of S-box inversion have been thoroughly discussed in previous works [15,16,17,18,19]. For simplicity, here we just state some properties related to this paper without any proof.

I. Matching the first 320-bit plane with $b_0b_1b_2b_3b_4$ all known.

In this case, each a_i can be recovered according to equations (2). And any restricting equation on recovered a_i can bring a gain of 2^1 — before setting the restricting equations, those involved a_i must be linearized first.

II. Matching the 64-bit truncation of the second plane with b_0 known.

In this case, there are two ways to set restricting equations. If attackers set $a_0 = b_0$, although the matching probability is only $3/4$, one restricting equation can still bring a gain of $3/4 \div 1/2 \approx 2^{0.58}$. If attackers set $a_0 = b_0$ and meanwhile ensure $a_1 = 1$ or $a_2 = 0$, b_0 must be matched and two restricting equations can bring a gain of 2^1.

III. Matching the 192-bit truncation of the second plane with $b_0b_1b_2$ known.

This case is much more complicated. On one hand, through delicate substitutions, it can be proved that:

[4] In [19], the authors mistook the extra cost for 2^2 with the last two bits "11", which is actually unnecessary.

$$\begin{cases} a_0 \oplus (b_1 \oplus 1) \cdot a_2 = b_0 \\ a_1 \oplus (b_2 \oplus 1) \cdot a_3 = b_1 \end{cases} \tag{3}$$

Therefore, restricting equations on a_0 or $a_0 \oplus a_2$ (depending on b_1) and a_1 or $a_1 \oplus a_3$ (depending on b_2) can always bring a gain of 2^1 each.

On the other hand, our attacks may encounter a special situation that only a_0 and a_4 can be restricted. In this case, attackers can set:

$$\begin{cases} a_0 = b_0 \oplus (b_1 \oplus 1) \cdot b_2 \\ a_4 = 0 \end{cases} \tag{4}$$

Then it can be proved that as long as b_1 and b_2 are randomly matched, b_0 must be simultaneously matched. Therefore, two restricting equations on a_0 and a_4 can always bring a gain of 2^1.

2.5 Notations

From this section on, we will no longer use A to denote the state of Keccak-f, since it cannot accurately show the execution process. Instead, we will use capital Greek letters (in $\{\Theta, P, \Pi, X, I\}$) with a superscript (from 1 to 3) to denote the state exactly after the corresponding step is executed. For examples, Π^2 denotes the state after the second π step, and X^3 denotes the state after the third χ step. In particular, I^0 denotes the initial state of a single Keccak-f (after XORing the message block). The first r bits of I^0 are named "input part" (XORed with r-bit message), and the last c bits of I^0 are named "restricted part" (uncontrollable in coming Keccak-f).

To avoid ambiguity, we will always use three indices in subscript to denote a component of the state. However, we may use "$*$" to indicate all possible values. For examples, $I^1_{*,y,z}$ is a 5-bit row, $I^1_{x,*,z}$ is a 5-bit column, $I^1_{x,y,*}$ is a 64-bit lane, $I^1_{*,y,*}$ is a 320-bit plane, and $I^1_{*,*,z}$ is a 5×5 slice. If the subscript is omitted, it indicates the 1600-bit whole state (like notations above).

Column sum setting is the core issue of preimage analysis on round-reduced Keccak. In this paper, we use S_A with two parameters x, z to denote the sum of a certain column from state A, which is:

$$S_A(x, z) = \bigoplus\nolimits_{y=0\sim4} A_{x,y,z} \tag{5}$$

Similarly, x, z may be replaced by "$*$" to indicate a set of column sums.

3 Overview

This section describes some existing attack thoughts of linear analysis on round-reduced Keccak-384/512, which greatly inspire our research. Developments of the attack framework can be divided into two parts: linear structure with *allocating model*, and quadratic structure with the *relinearization technique*.

3.1 Linear Structure with Allocating Model

In 2016, Guo et al. [15] applied linear analysis in round-reduced Keccak and proposed the basic linear structure. Their idea is to linearize step χ, which is the only non-linear step in function R, so that they can get an entirely-linear state after several rounds. Take their 1-round linear structure for Keccak-384 as an example (as shown in **Fig. 3**).

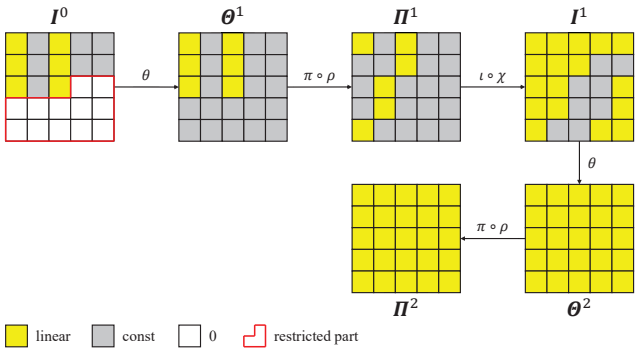

Fig. 3. The 1-round linear structure for Keccak-384 in [15].

Since the forward S-box calculation is $b_i = a_i \oplus (a_{i+1} \oplus 1) \cdot a_{i+2}$, to linearize step χ, attackers must ensure no consecutive linear bits exist in any row before step χ (in state Π^n). However, due to the first θ step, initial variable bits will diffuse messily in Π^1. To control the diffusion, attackers must fix related column sums by setting (linear) equations. For **Fig. 3**, the equations should be[5]:

$$\begin{cases} I^0_{0,0,z} \oplus I^0_{0,1,z} \oplus I^0_{0,2,z} \oplus I^0_{0,3,z} \oplus I^0_{0,4,z} = S_{I^0}(0,z) \\ I^0_{2,0,z} \oplus I^0_{2,1,z} \oplus I^0_{2,2,z} \oplus I^0_{2,3,z} \oplus I^0_{2,4,z} = S_{I^0}(2,z) \end{cases} \tag{6}$$

Then through column sum setting, the number of linear lanes remains 6 in Π^1, and the first χ step is successfully linearized. Yet according to the forward S-box calculation, variables in $\Pi^1_{x,y,*}$ may still diffuse to $I^1_{x-1,y,*}$ or $I^1_{x-2,y,*}$, and finally cover the whole Π^2. Therefore, Guo et al.'s basic linear structure can only pass through 1-round for Keccak-384.

In summary, by setting 384 initial variable bits and fixing 128 column sums, Guo et al. designed a 1-round linear structure for Keccak-384 with 256 degrees of freedom left. Those degrees of freedom were further used to restrict equivalent bits of $\Pi^2_{*,0,*}$, which can be recovered from the digest (cf. Section 2.4). Counting the padding rule, their searching complexity of preimage attack on 2-round Keccak-384 is $2^{384-256+1} = 2^{129}$.

[5] Among any equation system in this paper, z varies from 0 to 63, red indicates linear variables, purple indicates controllable constants (attackers can arbitrarily set) and grey indicates uncontrollable constants (have been determined).

To promote the basic linear structure, a natural idea is to further control the diffusion in the first χ step, which requires extra conditions in the restricted part of I^0. In 2019, Li and Sun [16] constructed an allocating model and solved this problem. By applying such a model, they merely improved preimage attacks on round-reduced Keccak-224/256. For better comparison, here we simply design a 2-round linear structure for Keccak-384 in accordance with their idea (as shown in **Fig. 4**).

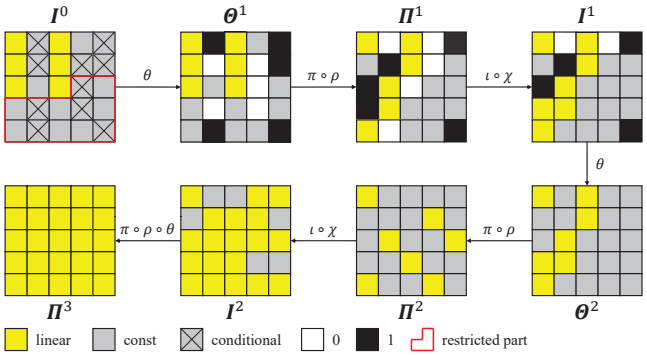

Fig. 4. A 2-round linear structure for Keccak-384 by applying *allocating model*.

Compared to **Fig. 3**, this structure starts with identical initial variable bits (and identical column sum equations) in I^0. However, this structure maintains several lanes of '0' and '1' in Π^1, so that the diffusion in the first χ step can be effectively controlled. Similarly, after setting 192 column sum equations in the second θ step (as enclosed below), this structure can linearize the second χ step with $384 - 128 - 192 = 64$ degrees of freedom left.

$$\begin{cases} I^1_{0,0,z} \oplus I^1_{0,1,z} \oplus I^1_{0,2,z} \oplus I^1_{0,3,z} \oplus I^1_{0,4,z} = S_{I^1}(0,z) \\ I^1_{1,0,z} \oplus I^1_{1,1,z} \oplus I^1_{1,2,z} \oplus I^1_{1,3,z} \oplus I^1_{1,4,z} = S_{I^1}(1,z) \\ I^1_{2,0,z} \oplus I^1_{2,1,z} \oplus I^1_{2,2,z} \oplus I^1_{2,3,z} \oplus I^1_{2,4,z} = S_{I^1}(2,z) \end{cases} \tag{7}$$

Those lanes of '0' and '1' are originally generated in the first θ step. Then according to the calculation of step θ, extra conditions are required in advance:

$$\begin{cases} I^0_{1,0,z} = I^0_{1,1,z} \oplus 1 = I^0_{1,3,z} \oplus 1 = I^0_{1,4,z} \\ I^0_{3,1,z} = I^0_{3,2,z} = I^0_{3,3,z} \\ I^0_{4,0,z} = I^0_{4,1,z} = I^0_{4,4,z} \end{cases} \tag{8}$$

Among above conditions, $I^0_{1,3,z} \oplus 1 = I^0_{1,4,z}$ and $I^0_{3,2,z} = I^0_{3,3,z}$ belong to the restricted part, which attackers cannot control. We name this kind of conditions "restricted conditions". Those restricted conditions can only be satisfied by the output of previous Keccak-f, and thus require an allocating model. In general

cases, even if attackers satisfy all restricted conditions by an exhaustive search, the complexity is still far away from that of preimage searching (neglecting the solving time, temporarily) — for **Fig. 4**, the former is 2^{128}, while the latter is $2^{384-64+1} = 2^{321}$. Therefore, the total searching complexity only depends on the freedom space left in the linear structure.

Another noticeable problem in *allocating model* is the size of random space, which is the total number of different equation systems the linear structure can generate. Let d_1 denote the searching complexity of satisfying all restricted conditions (d_1 can be constant-level), d_2 denote the searching complexity of preimage attack, and d_r denote the size of random space. Usually $d_r < d_2$, under this case attackers are expected to restart the linear structure (generating another qualified I^0) $[d_2/d_r]$ times to find a preimage. Then the total searching complexity of satisfying all restricted conditions is $d_1 \times [d_2/d_r]$ — if $d_r < d_1$, the total searching complexity becomes $[d_1/d_r] \times d_2$ rather than d_2. Therefore, when applying *allocating model*, the size of random space should satisfy $d_r \geq d_1$.

As for the calculation of d_r, it depends on the number of controllable column sums. For **Fig. 4**, $S_{I^1}(0,*)$, $S_{I^1}(1,*)$ and $S_{I^1}(2,*)$ are all controllable, while S_{I^0} must satisfy following relations[6] to generate those lanes of '0' and '1' in Θ^1. Counting controllable $S_{I^0}(1,*)$, the size of random space is $d_r = 2^{64+192} = 2^{256}$.

$$\begin{cases} S_{I^0}(0,z) \oplus S_{I^0}(2,z-1) \oplus I^0_{1,3,z} = \Theta^1_{1,3,z} = 0 \\ S_{I^0}(2,z) \oplus S_{I^0}(4,z-1) \oplus I^0_{3,3,z} = \Theta^1_{3,3,z} = 0 \\ S_{I^0}(3,z) \oplus S_{I^0}(0,z-1) \oplus I^0_{4,4,z} = \Theta^1_{4,4,z} = 1 \end{cases} \tag{9}$$

In summary, by applying *allocating model*, the basic linear structure can be promoted to 2-round for Keccak-384. Under this structure, the total searching complexity of preimage attack on 3-round Keccak-384 is $2^{384-64+1} = 2^{321}$. The model parameters are $d_1 = 2^{128}$, $d_2 = 2^{321}$ and $d_r = 2^{256}$, satisfying $d_1 \leq d_r$.

3.2 Quadratic Structure with the Relinearization Technique

In the previous section, the designs of linear structure all aim to get an entirely-linear state, which is unnecessary in some cases. Like the example in **Fig. 4**, the entirely-linear Π^3 contains 320 restricting equations on $\Pi^3_{*,0,*}$ that can bring a gain of 2^1 each, while the number of degrees of freedom left is only 64. In 2019, Rajasree [17] designed a 2-round partially-linear structure for Keccak-384 (as shown in **Fig. 5**) that achieved a balance between the two values.

In the partially-linear structure, quadratic lanes are allowed to exist, so that the number of linear restricting equations on $\Pi^3_{*,0,*}$ and the number of degrees of freedom left can be matched (both are 64). Calculations of model parameters are very similar to the contents in Section 3.1 — for simplicity, here we directly conclude that the model parameters are $d_1 = 2^{64}$, $d_2 = 2^{321}$ and $d_r = 2^{320}$.

Compared to the entirely-linear structure (**Fig. 4**), Rajasree's partially-linear structure can merely enlarge the size of random space d_r, remaining the searching

[6] Please notice that $S_{I^0}(4,*)$ has actually been determined by conditional constants.

complexity d_2 unchanged. Yet this idea still gives us great inspirations because a larger random space will greatly contribute in the application of *extra linear dependence*.

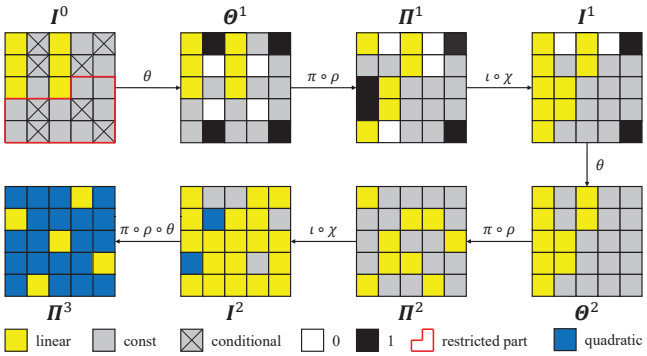

Fig. 5. The 2-round partially-linear structure for Keccak-384 in [17].

However, Rajasree's attack framework has not fully balanced the number of linear restricting equations and the number of degrees of freedom left, because all those 256 quadratic bits in $\Pi^3_{*,0,*}$ are generated by the diffusion of only 128 quadratic terms in I^2. This is exactly the suitable case to apply the *relinearization technique*. The *relinearization technique* is a solving algorithm for special quadratic (or higher-degree) equation systems, where the number of equations is much larger than the number of different quadratic (or non-linear) terms. Take the equation system in **Fig. 6** as an example.

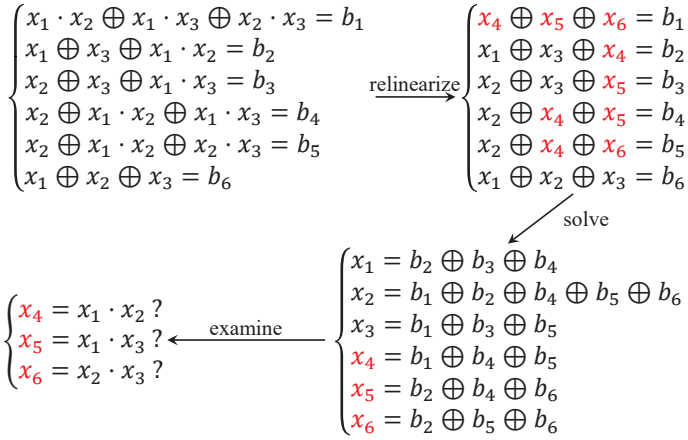

Fig. 6. An example of applying the *relinearization technique*.

The equation system in **Fig. 6** contains 3 variables and 6 equations — if it is a linear equation system, solvers can simply satisfy 3 out of 6 equations and match the others randomly with a probability of 2^{-3}. Although the example is a quadratic equation system, it can be equivalently solved by introducing 3 new variables to replace each different quadratic term. Then the equation system will become entirely-linear and can be simply solved by Gaussian Elimination on a 6×6 matrix. Finally solvers only need to examine whether 3 new variables are correctly matched, and the solving probability is equivalently[7] 2^{-3}.

Generally speaking, for a quadratic equation system that contains v variables and e equations, the *relinearization technique* is suitable only when the number of different quadratic terms q satisfies $q \leq e - v$, and the solving probability is $2^{e-(v+q)+q} = 2^{e-v}$. This means in special scenes, the freedom space can bring an equivalent gain even though the restricting equations are quadratic.

Reviewing Rajasree's attack framework (**Fig. 5**), if we apply the *relinearization technique*, the system parameters will be $v = 64$, $e = 320$ and $q = 128$, which apparently have room for improvements (still far away from $q = e - v$). In 2021, Liu et al. [19] designed a 2-round quadratic structure[8] for Keccak-384 (as shown in **Fig. 7**). Unlike **Fig. 5**, their attack framework can indeed enlarge the freedom space.

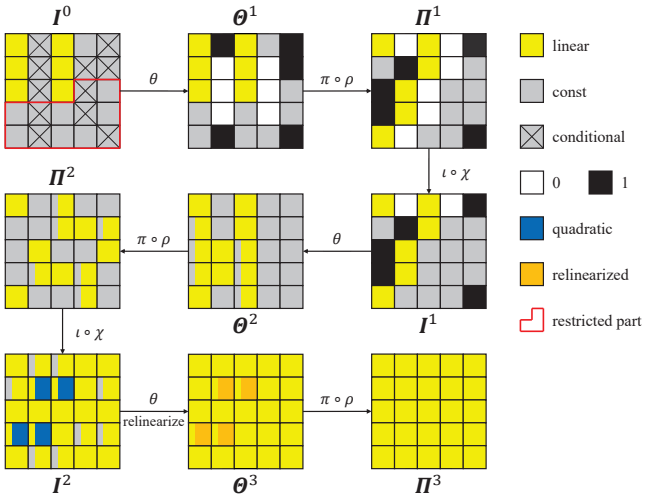

Fig. 7. The 2-round quadratic structure for Keccak-384 in [19].

The core idea of their attack framework is to reduce some column sum equations in $S_{I^1}(1, *)$, fixing only t out of 64 instead:

[7] They are not strictly equivalent because the probability of a random $n \times n$ matrix being singular depends on n.

[8] Without relinearization, the final state Π^3 will be (almost) entirely-quadratic.

$$\begin{cases} I^1_{0,0,z} \oplus I^1_{0,1,z} \oplus I^1_{0,2,z} \oplus I^1_{0,3,z} \oplus I^1_{0,4,z} = S_{I^1}(0,z) \\ I^1_{1,0,z} \oplus I^1_{1,1,z} \oplus I^1_{1,2,z} \oplus I^1_{1,3,z} \oplus I^1_{1,4,z} = S_{I^1}(1,z),\ z = z_1, z_2, \ldots, z_t \\ I^1_{2,0,z} \oplus I^1_{2,1,z} \oplus I^1_{2,2,z} \oplus I^1_{2,3,z} \oplus I^1_{2,4,z} = S_{I^1}(2,z) \end{cases} \tag{10}$$

Then this quadratic structure can leave $64-t$ more degrees of freedom, yet generating $256-4t$ quadratic bits in I^2. Fortunately, by applying the *relinearization technique*, as long as $128-t+256-4t \leq 320$ holds, the freedom space can still bring an equivalent gain without loss. It is derived that $t \geq 13$.

In summary, by reducing 51 column sum equations in $S_{I^1}(1,*)$ (269 in total), Liu et al. designed a 2-round quadratic structure for Keccak-384 with 115 degrees of freedom left. Although they finally obtained a quadratic equation system, the remaining freedom space can still bring an equivalent gain of 2^{115} by applying the *relinearization technique*. The model parameters of their attack framework are $d_1 = 2^{128}$, $d_2 = 2^{270}$ and $d_r = 2^{141}$ (with S_{I^0} all determined and 141 controllable column sums in S_{I^1}). The system parameters of their attack framework are $v = 256$, $e = 320 + 141 = 461$ and $q = 4 \times (64 - 13) = 204$.

Calculation of "Size". In [19], the authors used a unique way to calculate the total number of variables in the final quadratic equation system. They regarded the number of initial variable bits as reduced by the column sum equations in S_{I^0} (e.g. $384-128 = 256$ in **Fig. 7**). As a result, "Size" of their attack framework for 3-round Keccak-384 is thus $256 + 204 = 460$. In this paper, we will use the same way to calculate "Size" for better comparison.

4 Extra Linear Dependence

This section discusses *extra linear dependence*, which is the core technology of our improved preimage attacks on round-reduced Keccak-384/512.

4.1 Basic Principle of Extra Linear Dependence

The technology of *extra linear dependence* aims to construct linear-dependent bit pairs in Π^n, so that some pairs of restricting equations can be simultaneously satisfied by only 1 degree of freedom, and the number of equations in the final equation system can exceed the limit of matrix rank. For example, if attackers construct p linear-dependent bit pairs, then p extra restricting equations can be added into the final equation system (the matrix rank is still unchanged), and the complexity reduce of preimage searching is just corresponding to the total gain of p extra restricted bits in Π^n.

Before discussing the construction of multiple linear-dependent bit pairs, we should first explain the basic principle of one bit pair being linear-dependent[9]. The basic principle is revealed in **Fig. 8**.

[9] Actually this basic principle is just identical to the technology of *zero coefficient* [18], which also inspires our research.

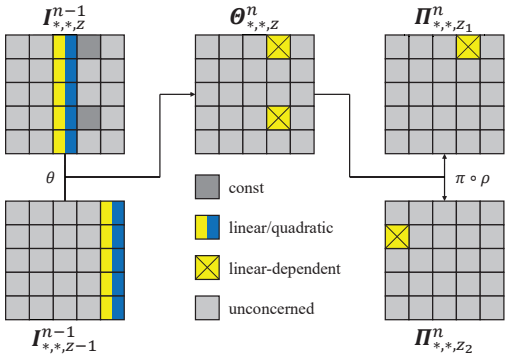

Fig. 8. The basic principle of *extra linear dependence*.

Suppose there are two restricting equations in Π^n. Each equation consists of 11 bits from Θ^n. Then if two restricted bits are permuted from the same column $\Theta^n_{x,*,z}$, 10 out of 11 bits are duplicate in the equation pair. Therefore, as long as 2 unique bits are both constants, linear dependence of the equation pair can be ensured. This relation can be written as:

$$\Theta^n_{x,y_1,z} \oplus \Theta^n_{x,y_2,z} = I^{n-1}_{x,y_1,z} \oplus I^{n-1}_{x,y_2,z} = \text{const} \tag{11}$$

One step backwards, since $I^{n-1}_{x,y,z} = \Pi^{n-1}_{x,y,z} \oplus (\Pi^{n-1}_{x+1,y,z} \oplus 1) \cdot \Pi^{n-1}_{x+2,y,z}$, whether $I^{n-1}_{x,y,z}$ is a constant depends on related bits in Π^{n-1}. If $\Pi^{n-1}_{x+1,y,z}$ or $\Pi^{n-1}_{x+2,y,z}$ is a variable, attackers can prevent the diffusion of variable by ensuring $\Pi^{n-1}_{x+2,y,z} = 0$ or $\Pi^{n-1}_{x+1,y,z} = 1$. However, if $\Pi^{n-1}_{x,y,z}$ has been a variable, $I^{n-1}_{x,y,z}$ is impossible to be a constant. The value control further depends on column sums in $S_{I^{n-2}}$. For example, when constructing the linear-dependent bit pair $I^{n-1}_{3,0,z} \oplus I^{n-1}_{3,3,z} = c(z)$ in **Fig. 8**, suppose the distribution of variables (indicated by red) is:

$$\begin{cases} I^{n-1}_{3,0,z} = \Pi^{n-1}_{3,0,z} \oplus (\Pi^{n-1}_{4,0,z} \oplus 1) \cdot \Pi^{n-1}_{0,0,z} = \Pi^{n-1}_{3,0,z} \\ I^{n-1}_{3,3,z} = \Pi^{n-1}_{3,3,z} \oplus (\Pi^{n-1}_{4,3,z} \oplus 1) \cdot \Pi^{n-1}_{0,3,z} = \Pi^{n-1}_{3,3,z} \end{cases} \tag{12}$$

Then column sums in $S_{I^{n-2}}$ are required to satisfy[10]:

$$\begin{cases} \Pi^{n-1}_{0,0,z} = \Theta^{n-1}_{0,0,z} = S_{I^{n-2}}(4,z) \oplus S_{I^{n-2}}(1,z-1) \oplus I^{n-2}_{0,0,z} = 0 \\ \Pi^{n-1}_{4,3,z} = \Theta^{n-1}_{3,4,z-56} = S_{I^{n-2}}(2,z-56) \oplus S_{I^{n-2}}(4,z-57) \oplus I^{n-2}_{3,4,z-56} = 1 \\ \Pi^{n-1}_{3,0,z} \oplus \Pi^{n-1}_{3,3,z} = \Theta^{n-1}_{3,3,z-21} \oplus \Theta^{n-1}_{2,3,z-15} = I^{n-2}_{3,3,z-21} \oplus S_{I^{n-2}}(2,z-21) \;\backslash\backslash \\ \oplus S_{I^{n-2}}(4,z-22) \oplus S_{I^{n-2}}(1,z-15) \oplus S_{I^{n-2}}(3,z-16) \oplus I^{n-2}_{2,3,z-15} = c(z) \end{cases} \tag{13}$$

[10] Here $c(z)$ is a certain value that can be calculated from two known restricting equations (cf. Section 2.4) on $\Pi^n_{3,0,z_1}$ and $\Pi^n_{0,1,z_2}$. Therefore, it requires an extra condition to match $\Pi^{n-1}_{3,0,z} \oplus \Pi^{n-1}_{3,3,z} = c(z)$.

In this paper, these equations are named "column sum conditions" (distinguished from "column sum equations"). Satisfying column sum conditions does not require any degree of freedom, but requires controllable column sums in the attack framework. Therefore, when applying the technology of *extra linear dependence*, the random space will be greatly compressed, and the calculation of d_r will become a momentous problem.

4.2 Construction of Multiple Linear-Dependent Bit Pairs

The basic principle of *extra linear dependence* has been explained in the previous section. Actually in [18], the authors have proposed a similar technology named *zero coefficient*. Such a technology aimed to find some linear-dependent pairs in a certain equation system to save degrees of freedom. As a result, the authors found 11 linear-dependent pair in 173 equations. This value is limited mainly because their analysis object is Keccak-224/256[11]. Yet for Keccak-384/512, the first two planes in Π^n contains as many as 320 (possibly) linear-dependent bit pairs as summarized below("$\rightarrow$" means the permutation through $\pi \circ \rho$):

$$\begin{cases} (\Theta^n_{3,0,z} \rightarrow \Pi^n_{0,1,z+28}) \oplus (\Theta^n_{3,3,z} \rightarrow \Pi^n_{3,0,z+21}) = \text{const} \\ (\Theta^n_{4,1,z} \rightarrow \Pi^n_{1,1,z+20}) \oplus (\Theta^n_{4,4,z} \rightarrow \Pi^n_{4,0,z+14}) = \text{const} \\ (\Theta^n_{0,2,z} \rightarrow \Pi^n_{2,1,z+3}) \oplus (\Theta^n_{0,0,z} \rightarrow \Pi^n_{0,0,z}) = \text{const} \\ (\Theta^n_{1,3,z} \rightarrow \Pi^n_{3,1,z+45}) \oplus (\Theta^n_{1,1,z} \rightarrow \Pi^n_{1,0,z+44}) = \text{const} \\ (\Theta^n_{2,4,z} \rightarrow \Pi^n_{4,1,z+61}) \oplus (\Theta^n_{2,2,z} \rightarrow \Pi^n_{2,0,z+43}) = \text{const} \end{cases} \tag{14}$$

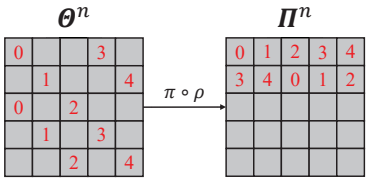

Fig. 9. 320 (possibly) linear-dependent bit pairs.

Therefore, the technology of *extra linear dependence* tends to "construct" rather than "find" linear-dependent bit pairs. When applying *extra linear dependence*, attackers had better:

[11] Due to the algebraic structure of Keccak, any linear-dependent bit pair can never be located in the same plane of Π^n — this leads to the main difference between *zero coefficient* and *extra linear dependence*. Since the output of Keccak-224/256 is generated from merely the first plane of Π^n, *zero coefficient* can only be applied to the linearization process and its improvement is quite limited. However, *extra linear dependence* can directly be applied to the output matching of Keccak-384/512. Then the improvement can be much larger and the construction of linear-dependent bit pairs becomes a worth-thinking problem.

I. Maximize the number of constructed linear-dependent bit pairs to obtain the largest gain.

II. Minimize the number of required column sum conditions to ensure the random space ($d_r \geq d_1$).

From the example in the previous section (equations (13)), it's shown that the construction of a single linear-dependent bit pair requires three column sum conditions. However, by designing specific attack frameworks (depending on the distribution of variables), attackers can construct multiple linear-dependent bit pairs with less than three column sum conditions in average. Take the same bit pair $I^{n-1}_{3,0,z} \oplus I^{n-1}_{3,3,z} = c(z)$ as an example. Suppose the distribution of variables changes to:

$$\begin{cases} I^{n-1}_{3,0,z} = \Pi^{n-1}_{3,0,z} \oplus (\Pi^{n-1}_{4,0,z} \oplus 1) \cdot \Pi^{n-1}_{0,0,z} = \Pi^{n-1}_{3,0,z} \\ I^{n-1}_{3,3,z} = \Pi^{n-1}_{3,3,z} \oplus (\Pi^{n-1}_{4,3,z} \oplus 1) \cdot \Pi^{n-1}_{0,3,z} \end{cases} \tag{15}$$

In this case, although variables cannot diffuse to $I^{n-1}_{3,3,z}$, attackers still need to fix $\Pi^{n-1}_{4,3,z}$ or $\Pi^{n-1}_{0,3,z}$ — otherwise, $I^{n-1}_{3,0,z} \oplus I^{n-1}_{3,3,z} = c(z)$ will become a quadratic equation from the perspective of $S_{I^{n-2}}$. Then since $\Pi^{n-1}_{4,0,z} = 1$ has been required, attackers can choose to fix $\Pi^{n-1}_{0,3,z} = 1$, which involves the same column sums as the former. Finally, attackers can construct 64 linear-dependent bit pairs with only two column sum conditions[12] in average ($I^{n-2}_{4,0,z} = I^{n-2}_{4,4,z}$ is required in the attack framework):

$$\begin{cases} \Pi^{n-1}_{4,0,z} = \Theta^{n-1}_{4,4,z-14} = S_{I^{n-2}}(3, z-14) \oplus S_{I^{n-2}}(0, z-15) \oplus I^{n-2}_{4,4,z-14} = 1 \\ \Pi^{n-1}_{0,3,z} = \Theta^{n-1}_{4,3,z-27} = S_{I^{n-2}}(3, z-27) \oplus S_{I^{n-2}}(0, z-28) \oplus I^{n-2}_{4,0,z-27} = 1 \\ \Pi^{n-1}_{3,0,z} \oplus \Pi^{n-1}_{3,3,z} \oplus \Pi^{n-1}_{4,3,z} \oplus 1 = \Theta^{n-1}_{3,3,z-21} \oplus \Theta^{n-1}_{2,3,z-15} \oplus \Theta^{n-1}_{3,4,z-56} \oplus 1 \ \backslash\backslash \\ = S_{I^{n-2}}(2, z-21) \oplus S_{I^{n-2}}(4, z-22) \oplus S_{I^{n-2}}(1, z-15) \oplus S_{I^{n-2}}(3, z-16) \ \backslash\backslash \\ \oplus S_{I^{n-2}}(2, z-56) \oplus S_{I^{n-2}}(4, z-57) \oplus I^{n-2}_{3,3,z-21} \oplus I^{n-2}_{2,3,z-15} \oplus I^{n-2}_{3,4,z-56} = c(z) \end{cases} \tag{16}$$

Generally speaking, when constructing multiple linear-dependent bit pairs, the effect of *extra linear dependence* highly depends on the attack framework. And after massive attempts on different designs, it is concluded that combining *extra linear dependence* and quadratic structures can exactly reach the lowest final complexity.

4.3 Gain Analysis of Extra Linear Dependence

Since $\Pi^{n}_{*,0,*}$ can be fully recovered from the digest. as long as the final equation system contains a restricting equation on $\Pi^{n}_{x_1,0,z_1}$, linear dependence between $\Pi^{n}_{x_1,0,z_1}$ and $\Pi^{n}_{x_2,0,z_2}$ is equivalent to an extra restricting equation on $\Pi^{n}_{x_2,0,z_2}$ (corresponding to a_{x_2}). As for the gain of restricting equations on different a_i, please refer to Section 2.4.

[12] Please notice that as long as $I^{n-2}_{4,0,z} = I^{n-2}_{4,4,z}$ holds (z varies from 0 to 63), the first line and the second line will correspond to an identical set.

For simplicity, here we generally summarize that:

I. Gains of multiple restricting equations in different rows can be summed up independently.

II. In most cases, one restricting equation can bring a gain of $2^{0.58}$, and two restricting equations in the same row can bring a gain of 2^1.

5 Preimage Attacks on Round-Reduced Keccak-384/512

This section provides improved preimage attacks on 2-round and 3-round Keccak-384/512. For 2-round Keccak-384/512 and 3-round Keccak-384, we modify the quadratic structures designed in [19] to apply *extra linear dependence* and construct lane-level linear-dependent bit pairs. As for 3-round Keccak-512, although *extra linear dependence* can hardly be applied due to lack of controllable column sums, we still make a progress by rectifying an omission in applying the *relinearization technique* and designing an improved quadratic structure.

5.1 Improved Preimage Attack on 2-Round Keccak-384

Our attack framework of improved preimage attack on 2-round Keccak-384 is given in **Fig. 10**.

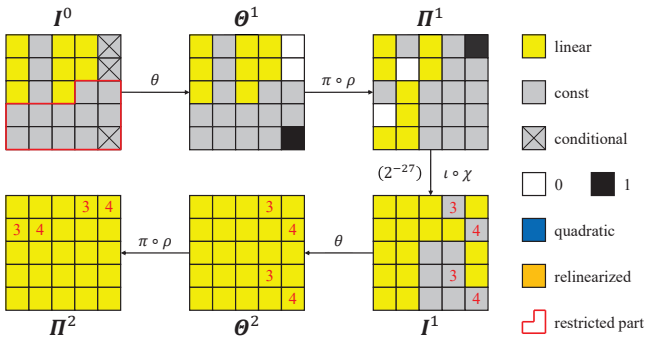

Fig. 10. A modified 1-round quadratic structure for Keccak-384.

Modification of the quadratic structure. Inspired by the design in [19], we set identical 512 variable bits and 192 column sum equations in I^0. 8 linear lanes are then permuted to different locations in Π^1, among which 2 consecutive ones will generate 1 quadratic lane $I^1_{4,4,*}$ through the first χ step. In [19], the authors relinearized $I^1_{4,4,*}$ by introducing new variables, while in this paper, we regard $I^1_{4,4,z} = \Pi^1_{4,4,z} \oplus (\Pi^1_{0,4,z} \oplus 1) \cdot \Pi^1_{1,4,z} = \Pi^1_{4,4,z}$ with a holding probability of 3/4.

Parameters about the relinearization technique. Since $I^1_{4,4,z}$ is regarded as a constant, the *relinearization technique* is unnecessary in our attack framework.

In conclusion, this quadratic structure can leave $512 - 192 = 320$ degrees of freedom but hold with a probability of only $(3/4)^{64} \approx 2^{-27}$. Entire column sum equations are given below. Under this structure, the basic searching complexity (before applying *extra linear dependence*) is $2^{384-320+1} \times 2^{27} = 2^{92}$.

$$\begin{cases} I^0_{0,0,z} \oplus I^0_{0,1,z} \oplus I^0_{0,2,z} \oplus I^0_{0,3,z} \oplus I^0_{0,4,z} = S_{I^0}(0,z) \\ I^0_{2,0,z} \oplus I^0_{2,1,z} \oplus I^0_{2,2,z} \oplus I^0_{2,3,z} \oplus I^0_{2,4,z} = S_{I^0}(2,z) \\ I^0_{3,0,z} \oplus I^0_{3,1,z} \oplus I^0_{3,2,z} \oplus I^0_{3,3,z} \oplus I^0_{3,4,z} = S_{I^0}(3,z) \end{cases} \tag{17}$$

As for the application of *extra linear dependence*, the target linear-dependent bit pairs are (as marked by red numbers in **Fig. 10**):

$$\begin{cases} (\Theta^2_{3,0,z} \to \Pi^2_{0,1,z+28}) \oplus (\Theta^2_{3,3,z} \to \Pi^2_{3,0,z+21}) = I^1_{3,0,z} \oplus I^1_{3,3,z} = c_1(z) \\ (\Theta^2_{4,1,z} \to \Pi^2_{1,1,z+20}) \oplus (\Theta^2_{4,4,z} \to \Pi^2_{4,0,z+14}) = I^1_{4,1,z} \oplus I^1_{4,4,z} = c_2(z) \end{cases} \tag{18}$$

And the distribution of variables in Π^1 is:

$$\begin{cases} I^1_{3,0,z} = \Pi^1_{3,0,z} \oplus (\Pi^1_{4,0,z} \oplus 1) \cdot \Pi^1_{0,0,z} = \Pi^1_{3,0,z} \\ I^1_{3,3,z} = \Pi^1_{3,3,z} \oplus (\Pi^1_{4,3,z} \oplus 1) \cdot \Pi^1_{0,3,z} \\ I^1_{4,1,z} = \Pi^1_{4,1,z} \oplus (\Pi^1_{0,1,z} \oplus 1) \cdot \Pi^1_{1,1,z} = \Pi^1_{4,1,z} \\ I^1_{4,4,z} = \Pi^1_{4,4,z} \oplus (\Pi^1_{0,4,z} \oplus 1) \cdot \Pi^1_{1,4,z} = \Pi^1_{4,4,z} \text{ (prob. : } 3/4) \end{cases} \tag{19}$$

It's found that $\Pi^1_{4,0,*} = 1$, $\Pi^1_{0,3,*} = 0$ and $\Pi^1_{1,1,*} = 0$ involve common column sums. Thus the entire column sum conditions are ($I^0_{4,0,z} = I^0_{4,1,z} = I^0_{4,4,z} \oplus 1$ is required in the quadratic structure):

$$\begin{cases} S_{I^0}(3,z) \oplus S_{I^0}(0,z-1) = I^0_{4,4,z} \oplus 1 \\ \Pi^1_{3,0,z} \oplus \Pi^1_{3,3,z} = \Theta^1_{3,3,z-21} \oplus \Theta^1_{2,3,z-15} = S_{I^0}(2,z-21) \oplus S_{I^0}(4,z-22) \ \backslash\backslash \\ \oplus I^0_{3,3,z-21} \oplus S_{I^0}(1,z-15) \oplus S_{I^0}(3,z-16) \oplus I^0_{2,3,z-15} = c_1(z) \\ \Pi^1_{4,1,z} \oplus \Pi^1_{4,4,z} = \Theta^1_{2,4,z-61} \oplus \Theta^1_{1,4,z-2} = S_{I^0}(1,z-61) \oplus S_{I^0}(3,z-62) \ \backslash\backslash \\ \oplus I^0_{2,4,z-61} \oplus S_{I^0}(0,z-2) \oplus S_{I^0}(2,z-3) \oplus I^0_{1,4,z-2} = c_2(z) \end{cases} \tag{20}$$

The examination shows that the rank of 192 column sum conditions is also 192. Therefore, for any values of $c_1(z)$ and $c_2(z)$ (determined by corresponding restricting equations), 128 linear-dependent bit pairs can be constructed under above column sum settings. Those linear-dependent bit pairs are equivalent to 128 extra restricting equations on $\Pi^2_{0,1,*}$ and $\Pi^2_{1,1,*}$ — according to Section 2.4, the total gain is 2^{64}.

In summary, by modifying the quadratic structure and applying *extra linear dependence*, we decrease the searching complexity of preimage attack on 2-round Keccak-384 from 2^{92} to 2^{28} (corresponding to 2^{39} 2-round Keccak calls). Since

the modified structure does not contain any restricted condition in I^0, *allocating model* is unnecessary for the attack. To support our analysis, **Table 3** presents an actual preimage of all '0' digest (matching the padding rule). Source codes are publicly available at https://github.com/lxe21/2round-Keccak384.

Table 3. An actual preimage for 2-round Keccak-384 (in big-endian order).

Initial State I^0 (one message block)				
65fbd7e20b5fe6b4	0000000000000000	b7fb5afa8f3f1ffb	dd2d29a4b4194993	ffffffffffffffff
9bec84cf16dc95f5	ffffffffd9c96b1	09e053aed207f2d7	dd2d292436194993	ffffffffffffffff
01e8ac92a37c8cbe	ffffffffd9c96b1	be1b097079a8ed2c	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
State Π^2 (the first two planes)				
0000000000008082	0000000000008082	0000000000000000	0000000000008082	0000000000000000
0000000000000000	ffffffffffffffff	d9216fe9e51c940e	5adf53be68c76a5e	0909240404100000
Digest State I^2 (the first two planes)				
0000000000000000	0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	fd21efe9f73c95af	d8214be9e10c940e	5adf53be68c76a5e	f6f6dbfbfbefffff

5.2 Improved Preimage Attack on 2-Round Keccak-512

Our attack framework of improved preimage attack on 2-round Keccak-512 is given in **Fig. 11**.

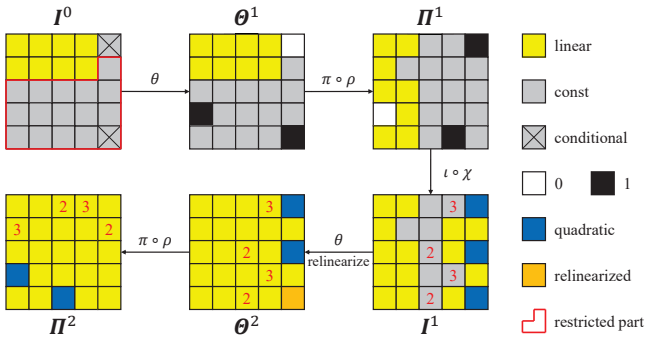

Fig. 11. A modified 1-round quadratic structure for Keccak-512.

Modification of the quadratic structure. Inspired by the design in [19], we set identical 512 variable bits and 256 column sum equations in I^0. 8 linear lanes

are then permuted to different locations in Π^1, among which 6 consecutive ones will generate 3 quadratic lanes $I^1_{4,0,*}$, $I^1_{4,2,*}$ and $I^1_{4,4,*}$ through the first χ step. Thus the *relinearization technique* is applied to solve the final equation system. In [19], the authors relinearized $I^1_{4,0,*}$, $I^1_{4,2,*}$ and $I^1_{4,4,*}$ by introducing 192 new variables. However, there was an omission that the relinearization of $I^1_{4,0,*}$ and $I^1_{4,2,*}$ need not require as many as 128 new variables. Actually by introducing 64 new variables on $I^1_{4,0,*} \oplus I^1_{4,2,*}$, the diffusion of quadratic bits can be prevented in the second θ step, leaving only 2 quadratic lanes[13] that would not affect the digest — both quadratic lanes are not located in the first two planes.

Parameters about the relinearization technique. Using the calculating way in [19], the number of variables is $v = 512-256 = 256$. Since each row $\Pi^2_{*,1,z}$ contains two restricting equations that can bring a gain of 2^1 (cf. equations (3)), counting 320 restricting equations on recovered $\Pi^2_{*,0,*}$, the number of equations is $e = 320 + 128 = 448$. And the number of different quadratic terms has been revealed to be $q = 128$, satisfying $q \leq e - v$.

In conclusion, this quadratic structure can leave $512 - 256 = 256$ degrees of freedom. Entire column sum equations are given below. Under this structure, the basic searching complexity is $2^{512-256+1} = 2^{257}$.

$$\begin{cases} I^0_{0,0,z} \oplus I^0_{0,1,z} \oplus I^0_{0,2,z} \oplus I^0_{0,3,z} \oplus I^0_{0,4,z} = S_{I^0}(0,z) \\ I^0_{1,0,z} \oplus I^0_{1,1,z} \oplus I^0_{1,2,z} \oplus I^0_{1,3,z} \oplus I^0_{1,4,z} = S_{I^0}(1,z) \\ I^0_{2,0,z} \oplus I^0_{2,1,z} \oplus I^0_{2,2,z} \oplus I^0_{2,3,z} \oplus I^0_{2,4,z} = S_{I^0}(2,z) \\ I^0_{3,0,z} \oplus I^0_{3,1,z} \oplus I^0_{3,2,z} \oplus I^0_{3,3,z} \oplus I^0_{3,4,z} = S_{I^0}(3,z) \end{cases} \tag{21}$$

As for the application of *extra linear dependence*, the target linear-dependent bit pairs are (as marked by red numbers in **Fig. 11**):

$$\begin{cases} (\Theta^2_{3,0,z} \to \Pi^2_{0,1,z+28}) \oplus (\Theta^2_{3,3,z} \to \Pi^2_{3,0,z+21}) = I^1_{3,0,z} \oplus I^1_{3,3,z} = c_1(z) \\ (\Theta^2_{2,4,z} \to \Pi^2_{4,1,z+61}) \oplus (\Theta^2_{2,2,z} \to \Pi^2_{2,0,z+43}) = I^1_{2,2,z} \oplus I^1_{2,4,z} = c_2(z) \end{cases} \tag{22}$$

And the distribution of variables in Π^1 is:

$$\begin{cases} I^1_{3,0,z} = \Pi^1_{3,0,z} \oplus (\Pi^1_{4,0,z} \oplus 1) \cdot \Pi^1_{0,0,z} = \Pi^1_{3,0,z} \\ I^1_{3,3,z} = \Pi^1_{3,3,z} \oplus (\Pi^1_{4,3,z} \oplus 1) \cdot \Pi^1_{0,3,z} \\ I^1_{2,2,z} = \Pi^1_{2,2,z} \oplus (\Pi^1_{3,2,z} \oplus 1) \cdot \Pi^1_{4,2,z} \\ I^1_{2,4,z} = \Pi^1_{2,4,z} \oplus (\Pi^1_{3,4,z} \oplus 1) \cdot \Pi^1_{4,4,z} \end{cases} \tag{23}$$

Similarly, $\Pi^1_{4,0,*} = 1$ and $\Pi^1_{0,3,*} = 0$ can be ensured by 64 common column sum conditions of $S_{I^0}(3,z) \oplus S_{I^0}(0,z-1) = I^0_{4,4,z} \oplus 1$. For the linear dependence on $I^1_{2,4,*}$, attackers can choose to fix $\Pi^1_{3,4,*} = 1$, which requires 64 column sum

[13] This idea is quite similar to Rajasree's [17] that allows quadratic parts to exist.

conditions of $S_{I^0}(4,z) \oplus S_{I^0}(1,z-1) = I^0_{0,3,z} \oplus 1$. Above column sum conditions have also ensured that:

$$\begin{cases} \Pi^1_{3,2,z} = \Theta^1_{4,3,z-8} = S_{I^0}(3,z-8) \oplus S_{I^0}(0,z-9) \oplus I^0_{4,3,z-8} \\ \qquad\quad = I^0_{4,4,z-8} \oplus 1 \oplus I^0_{4,3,z-8} \\ \Pi^1_{4,2,z} = \Theta^1_{0,4,z-18} = S_{I^0}(4,z-18) \oplus S_{I^0}(1,z-19) \oplus I^0_{0,4,z-18} \\ \qquad\quad = I^0_{0,3,z-18} \oplus 1 \oplus I^0_{0,4,z-18} \\ \Pi^1_{2,4,z} = \Theta^1_{4,2,z-39} = S_{I^0}(3,z-39) \oplus S_{I^0}(0,z-40) \oplus I^0_{4,2,z-39} \\ \qquad\quad = I^0_{4,4,z-39} \oplus 1 \oplus I^0_{4,2,z-39} \end{cases} \tag{24}$$

In other words, 128 common column sum conditions can totally fix 6 lanes that would affect the construction of 128 linear-dependent bit pairs. Thus the entire column sum conditions are ($I^0_{4,0,z} = I^0_{4,4,z} \oplus 1$ is required in the quadratic structure)[14]:

$$\begin{cases} S_{I^0}(3,z) \oplus S_{I^0}(0,z-1) = I^0_{4,4,z} \oplus 1 \\ S_{I^0}(4,z) \oplus S_{I^0}(1,z-1) = I^0_{0,3,z} \oplus 1 \\ \Pi^1_{3,0,z} \oplus \Pi^1_{3,3,z} = \Theta^1_{3,3,z-21} \oplus \Theta^1_{2,3,z-15} = S_{I^0}(2,z-21) \oplus S_{I^0}(4,z-22) \;\backslash\backslash \\ \oplus I^0_{3,3,z-21} \oplus S_{I^0}(1,z-15) \oplus S_{I^0}(3,z-16) \oplus I^0_{2,3,z-15} = c_1(z) \\ \Pi^1_{2,2,z} \oplus (\Pi^1_{3,2,z} \oplus 1) \cdot \Pi^1_{4,2,z} \oplus \Pi^1_{2,4,z} = \Theta^1_{3,2,z-25} \oplus (\Pi^1_{3,2,z} \oplus 1) \cdot \Pi^1_{4,2,z} \;\backslash\backslash \\ \oplus \Pi^1_{2,4,z} = S_{I^0}(2,z-25) \oplus S_{I^0}(4,z-26) \oplus I^1_{3,2,z-25} \oplus (I^0_{4,3,z-8} \oplus I^0_{4,4,z-8}) \;\backslash\backslash \\ \cdot (I^0_{0,3,z-18} \oplus I^0_{0,4,z-18} \oplus 1) \oplus I^0_{4,2,z-39} \oplus I^0_{4,4,z-39} \oplus 1 = c_2(z) \end{cases} \tag{25}$$

The examination shows that the rank of 256 column sum conditions is also 256. Therefore, for any values of $c_1(z)$ and $c_2(z)$ (determined by corresponding restricting equations), 128 linear-dependent bit pairs can be constructed under above column sum settings. Those linear-dependent bit pairs are equivalent to 128 extra restricting equations on $\Pi^2_{0,1,*}$ and $\Pi^2_{4,1,*}$ — according to equations (4), the total gain is 2^{64}. To avoid contradictions, attackers can construct the final equation system by 320 restricting equations on $\Pi^2_{*,0,*}$, 64 restricting equations on $\Pi^2_{1,1,*}$ or $\Pi^2_{1,1,*} \oplus \Pi^2_{3,1,*}$ (the number of equations becomes $e = 384$) and 128 extra restricting equations on $\Pi^2_{0,1,*}$ and $\Pi^2_{4,1,*}$.

In summary, by modifying the quadratic structure and applying *extra linear dependence*, we decrease the searching complexity of preimage attack on 2-round Keccak-512 from 2^{257} to 2^{193} (corresponding to 2^{204} 2-round Keccak calls). Although the modified structure does not contain any restricted condition in I^0, *allocating model* is still necessary for the attack because the quadratic structure cannot support a random space over 2^{193}. The model parameters are $d_1 = 2^0$, $d_2 = 2^{193}$ and $d_r = 2^0$ (with S_{I^0} all determined). Under this model, each turn of preimage searching will start with a randomized middle state.

[14] Please notice that $S_{I^0}(4,*)$ has actually been determined by conditional constants.

5.3 Improved Preimage Attack on 3-Round Keccak-384

Our attack framework of improved preimage attack on 3-round Keccak-384 is given in **Fig. 12**.

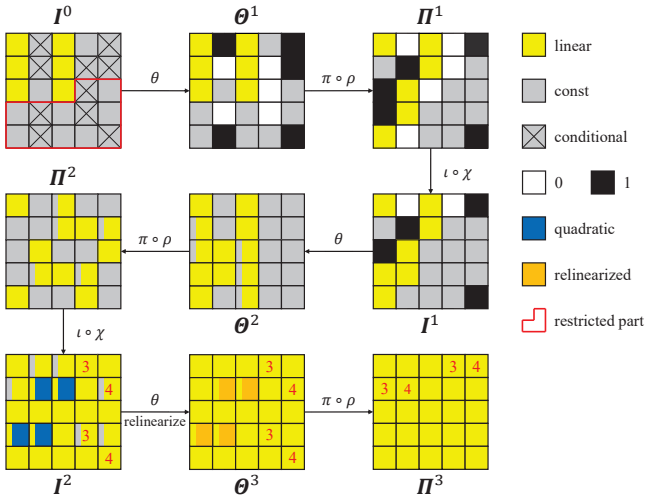

Fig. 12. A modified 2-round quadratic structure for Keccak-384.

Modification of the quadratic structure. We have introduced the design in [19] in Section 3.2. Their idea is to partially set $t = 13$ out of 64 column sum equations in $S_{I^1}(1.*)$ and solve the final equation system by the *relinearization technique*, so that the quadratic structure can leave 51 more degrees of freedom. The main body (as well as the relinearization part) of our modified structure is just the same as the previous one. However, we release those extra conditions in $I^0_{1,2,*}$. This change will influence the linear lane $\Pi^2_{3,4,*}(\leftarrow \Theta^2_{0,3,*})$, which would not generate any quadratic term in I^2 through the second χ step. Therefore, our modified structure can support a larger random space (enlarged by 2^{64}) for preimage searching, which means more controllable column sums can be fixed in the application of *extra linear dependence*.

Parameters about the relinearization technique. Using the calculating way in [19], the number of variables is $v = 384 - 128 = 256$, the number of equations is $e = 320 + 141 = 461$ (the column sum equations in S_{I^1} are counted into this part), and the number of different quadratic terms is $q = 4 \times (64 - 13) = 204$, satisfying $q \leq e - v$.

In conclusion, this quadratic structure can leave $384 - 128 - 192 + 51 = 115$ degrees of freedom. Entire column sum equations are given below. Under this structure, the basic searching complexity is $2^{384-115+1} = 2^{270}$.

$$\begin{cases} I^0_{0,0,z} \oplus I^0_{0,1,z} \oplus I^0_{0,2,z} \oplus I^0_{0,3,z} \oplus I^0_{0,4,z} = S_{I^0}(0,z) \\ I^0_{2,0,z} \oplus I^0_{2,1,z} \oplus I^0_{2,2,z} \oplus I^0_{2,3,z} \oplus I^0_{2,4,z} = S_{I^0}(2,z) \\ I^1_{0,0,z} \oplus I^1_{0,1,z} \oplus I^1_{0,2,z} \oplus I^1_{0,3,z} \oplus I^1_{0,4,z} = S_{I^1}(0,z) \\ I^1_{1,0,z} \oplus I^1_{1,1,z} \oplus I^1_{1,2,z} \oplus I^1_{1,3,z} \oplus I^1_{1,4,z} = S_{I^1}(1,z),\ z = z_1, z_2, \ldots, z_t \\ I^1_{2,0,z} \oplus I^1_{2,1,z} \oplus I^1_{2,2,z} \oplus I^1_{2,3,z} \oplus I^1_{2,4,z} = S_{I^1}(2,z) \end{cases} \tag{26}$$

Since the modified structure contains several restricted conditions in I^0, before applying *extra linear dependence*, attackers should calculate the number of controllable column sums they can fix (limited by $d_r \geq d_1$). To generate those lanes of '0' and '1' in Θ^1, I^0 must satisfy[15]:

$$\begin{cases} I^0_{1,0,z} = I^0_{1,1,z} \oplus 1 = I^0_{1,3,z} \oplus 1 = I^0_{1,4,z} \\ I^0_{3,1,z} = I^0_{3,2,z} = I^0_{3,3,z} \\ I^0_{4,0,z} = I^0_{4,1,z} = I^0_{4,4,z} \\ S_{I^0}(0,z) \oplus S_{I^0}(2,z-1) = I^0_{1,3,z} \\ S_{I^0}(2,z) \oplus S_{I^0}(4,z-1) = I^0_{3,3,z} \\ S_{I^0}(3,z) \oplus S_{I^0}(0,z-1) = I^0_{4,4,z} \oplus 1 \end{cases} \tag{27}$$

Counting controllable $S_{I^0}(1,*)$, the number of controllable column sums is $64 + 141 = 205$. And since there are 128 restricted conditions in I^0 ($d_1 = 2^{128}$), attackers can at most fix $205 - 128 = 77$ of all controllable column sums.

As for the application of *extra linear dependence*, the target linear-dependent bit pairs are (as marked by red numbers in **Fig. 12**):

$$\begin{cases} (\Theta^3_{3,0,z} \to \Pi^3_{0,1,z+28}) \oplus (\Theta^3_{3,3,z} \to \Pi^3_{3,0,z+21}) = I^2_{3,0,z} \oplus I^2_{3,3,z} = c_1(z) \\ (\Theta^3_{4,1,z} \to \Pi^3_{1,1,z+20}) \oplus (\Theta^3_{4,4,z} \to \Pi^3_{4,0,z+14}) = I^2_{4,1,z} \oplus I^2_{4,4,z} = c_2(z) \end{cases} \tag{28}$$

And the distribution of variables in Π^2 is:

$$\begin{cases} I^2_{3,0,z} = \Pi^2_{3,0,z} \oplus (\Pi^2_{4,0,z} \oplus 1) \cdot \Pi^2_{0,0,z} = \Pi^2_{3,0,z} \\ I^2_{3,3,z} = \Pi^2_{3,3,z} \oplus (\Pi^2_{4,3,z} \oplus 1) \cdot \Pi^2_{0,3,z} \\ I^2_{4,1,z} = \Pi^2_{4,1,z} \oplus (\Pi^2_{0,1,z} \oplus 1) \cdot \Pi^2_{1,1,z} \\ I^2_{4,4,z} = \Pi^2_{4,4,z} \oplus (\Pi^2_{0,4,z} \oplus 1) \cdot \Pi^2_{1,4,z} = \Pi^2_{4,4,z} \end{cases} \tag{29}$$

Unlike the constructions in previous sections, here $\Pi^2_{3,3,*}$ and $\Pi^2_{4,1,*}$ may be variables because those column sum equations in $S_{I^1}(1,*)$ are partially set.

$$\begin{cases} \Pi^2_{3,3,z} = \Theta^2_{2,3,z-15} = S_{I^1}(1,z-15) \oplus S_{I^1}(3,z-16) \oplus I^1_{2,3,z-15} \\ \Pi^2_{4,1,z} = \Theta^2_{2,4,z-61} = S_{I^1}(1,z-61) \oplus S_{I^1}(3,z-62) \oplus I^1_{2,4,z-61} \end{cases} \tag{30}$$

[15] Please notice that $S_{I^0}(4,*)$ has actually been determined by conditional constants.

Therefore, to construct linear dependence on $I^2_{3,3,z}$ (or $I^2_{4,1,z}$), the column sum $S_{I^1}(1, z-15)$ (or $S_{I^1}(1, z-61)$) must be fixed in the final equation system. It is summarized that one column sum equation on $S_{I^1}(1, z)$ can correspond to two extra restricting equations on $\Pi^3_{0,1,z+43}$ and $\Pi^3_{1,1,z+17}$.

Other involved bits can be similarly fixed by related column sums. Here we fix $\Pi^2_{0,3,z} = 1$ and $\Pi^2_{0,1,z} = 0$. Thus the entire column sum conditions are:

$$\begin{cases} \Pi^2_{4,0,z} = \Theta^2_{4,4,z-14} = S_{I^1}(3, z-14) \oplus S_{I^1}(0, z-15) \oplus I^1_{4,4,z-14} = 1 \\ \Pi^2_{0,3,z} = \Theta^2_{4,0,z-27} = S_{I^1}(3, z-27) \oplus S_{I^1}(0, z-28) \oplus I^1_{4,0,z-27} = 1 \\ \Pi^2_{3,0,z} \oplus \Pi^2_{3,3,z} \oplus \Pi^2_{4,3,z} \oplus 1 = \Theta^2_{3,3,z-21} \oplus \Theta^2_{2,3,z-15} \oplus \Theta^2_{3,4,z-56} \oplus 1 \ \backslash\backslash \\ = S_{I^1}(2, z-21) \oplus S_{I^1}(4, z-22) \oplus S_{I^1}(1, z-15) \oplus S_{I^1}(3, z-16) \ \backslash\backslash \\ \oplus S_{I^1}(2, z-56) \oplus S_{I^1}(4, z-57) \oplus I^1_{3,3,z-21} \oplus I^1_{2,3,z-15} \oplus I^1_{3,4,z-56} = c_1(z) \\ \Pi^2_{0,1,z} = \Theta^2_{3,0,z-28} = S_{I^1}(2, z-28) \oplus S_{I^1}(4, z-29) \oplus I^1_{3,0,z-28} = 0 \\ \Pi^2_{1,4,z} = \Theta^2_{3,1,z-55} = S_{I^1}(2, z-55) \oplus S_{I^1}(4, z-56) \oplus I^1_{3,1,z-55} = 0 \\ \Pi^2_{4,1,z} \oplus \Pi^2_{1,1,z} \oplus \Pi^2_{4,4,z} = \Theta^2_{2,4,z-61} \oplus \Theta^2_{4,1,z-20} \oplus \Theta^2_{1,4,z-2} \ \backslash\backslash \\ = S_{I^1}(1, z-61) \oplus S_{I^1}(3, z-62) \oplus S_{I^1}(3, z-20) \oplus S_{I^1}(0, z-21) \ \backslash\backslash \\ \oplus S_{I^1}(0, z-2) \oplus S_{I^1}(2, z-3) \oplus I^1_{2,4,z-61} \oplus I^1_{4,1,z-20} \oplus I^1_{1,4,z-2} = c_2(z) \end{cases} \tag{31}$$

Since there are only $t = 13$ column sum equations in $S_{I^1}(1, *)$, attackers can construct at most 26 linear-dependent bit pairs by above column sum settings. For the largest gain, attackers can apply a wise strategy to assemble 24 among 26 (and abandon the rest two) extra restricting equations in 12 rows. The details are given in **Table 4**.

Table 4. The choice of 13 column sum equations in $S_{I^1}(1, *)$.

Fixed Column Sum	Corresponding Linear-Dependent Bit Pairs	
$S_{I^1}(1, 21)$	$\Pi^3_{0,1,0} \oplus \Pi^3_{3,0,57} = \text{const}$	\
$S_{I^1}(1, 47)$	$\Pi^3_{0,1,26} \oplus \Pi^3_{3,0,19} = \text{const}$	$\Pi^3_{1,1,0} \oplus \Pi^3_{4,0,58} = \text{const}$
$S_{I^1}(1, 9)$	$\Pi^3_{0,1,52} \oplus \Pi^3_{3,0,45} = \text{const}$	$\Pi^3_{1,1,26} \oplus \Pi^3_{4,0,20} = \text{const}$
$S_{I^1}(1, 35)$	$\Pi^3_{0,1,14} \oplus \Pi^3_{3,0,7} = \text{const}$	$\Pi^3_{1,1,52} \oplus \Pi^3_{4,0,46} = \text{const}$
$S_{I^1}(1, 61)$	$\Pi^3_{0,1,40} \oplus \Pi^3_{3,0,33} = \text{const}$	$\Pi^3_{1,1,14} \oplus \Pi^3_{4,0,8} = \text{const}$
$S_{I^1}(1, 23)$	$\Pi^3_{0,1,2} \oplus \Pi^3_{3,0,59} = \text{const}$	$\Pi^3_{1,1,40} \oplus \Pi^3_{4,0,34} = \text{const}$
$S_{I^1}(1, 49)$	$\Pi^3_{0,1,28} \oplus \Pi^3_{3,0,21} = \text{const}$	$\Pi^3_{1,1,2} \oplus \Pi^3_{4,0,60} = \text{const}$
$S_{I^1}(1, 11)$	$\Pi^3_{0,1,54} \oplus \Pi^3_{3,0,47} = \text{const}$	$\Pi^3_{1,1,28} \oplus \Pi^3_{4,0,22} = \text{const}$
$S_{I^1}(1, 37)$	$\Pi^3_{0,1,16} \oplus \Pi^3_{3,0,9} = \text{const}$	$\Pi^3_{1,1,54} \oplus \Pi^3_{4,0,48} = \text{const}$
$S_{I^1}(1, 63)$	$\Pi^3_{0,1,42} \oplus \Pi^3_{3,0,35} = \text{const}$	$\Pi^3_{1,1,16} \oplus \Pi^3_{4,0,10} = \text{const}$
$S_{I^1}(1, 25)$	$\Pi^3_{0,1,4} \oplus \Pi^3_{3,0,61} = \text{const}$	$\Pi^3_{1,1,42} \oplus \Pi^3_{4,0,36} = \text{const}$
$S_{I^1}(1, 51)$	$\Pi^3_{0,1,30} \oplus \Pi^3_{3,0,23} = \text{const}$	$\Pi^3_{1,1,4} \oplus \Pi^3_{4,0,62} = \text{const}$
$S_{I^1}(1, 13)$	\	$\Pi^3_{1,1,30} \oplus \Pi^3_{4,0,24} = \text{const}$

The examination shows that the construction of above 24 linear-dependent bit pairs requires 72 column sum conditions without duplication, and the rank of all column sum conditions is also 72. Therefore, for any values of $c_1(z)$ and $c_2(z)$ (determined by corresponding restricting equations), 24 linear-dependent bit pairs can be constructed. Those linear-dependent bit pairs are equivalent to 24 extra restricting equations on $\Pi^2_{0,1,*}$ and $\Pi^2_{1,1,*}$ (assembled in 12 rows) — according to Section 2.4, the total gain is 2^{12}.

In summary, by modifying the quadratic structure and applying *extra linear dependence*, we decrease the searching complexity of preimage attack on 3-round Keccak-384 from 2^{270} to 2^{258} (corresponding to 2^{270} 3-round Keccak calls). Since the modified structure contains restricted conditions in I^0, *allocating model* is necessary for the attack. The model parameters are $d_1 = 2^{128}$, $d_2 = 2^{258}$ and $d_r = 2^{205-72} = 2^{133}$.

5.4 Improved Preimage Attack on 3-Round Keccak-512

Our attack framework of improved preimage attack on 3-round Keccak-512 is given in **Fig. 13**.

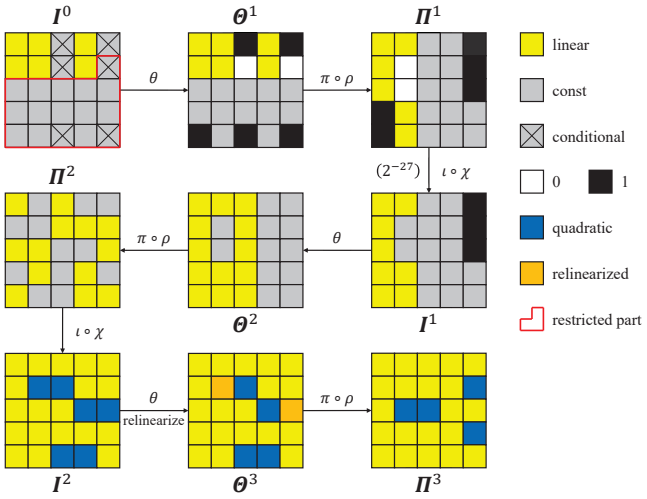

Fig. 13. An improved 2-round quadratic structure for Keccak-512.

Improvement of the quadratic structure. The design in [19] was similar to **Fig. 7**, which partially set $t = 54$ out of 128 column sum equations in $S_{I^1}(0.*)$ and $S_{I^1}(1.*)$. However, we have revealed in Section 5.2 that they left an omission in relinearization and thus their design can apparently be improved. By rectifying

the omission, we design an improved quadratic structure that passes through an entirely different route.

Our quadratic structure starts with 384 variable bits and 192 column sum equations in I^0. 6 linear lanes are then permuted to different locations in Π^1, among which 2 consecutive ones will generate 1 quadratic lane $I^1_{4,0,*}$ through the first χ step. Similarly, we regard $I^1_{4,0,z} = \Pi^1_{4,0,z} \oplus (\Pi^1_{0,0,z} \oplus 1) \cdot \Pi^1_{1,0,z} = \Pi^1_{4,0,z}$ with a holding probability of 3/4. In the second θ step, we abandon all column sum equations in $S_{I^1}(1,*)$, which generates 384 quadratic bits (6 entire lanes) in I^2. Fortunately, the relinearization of 384 quadratic bits only requires 256 new variables on $I^2_{1,1,*}$, $I^2_{2,1,*} \oplus I^2_{2,4,*}$, $I^2_{3,2,*} \oplus I^2_{3,4,*}$ and $I^2_{4,2,*}$. By introducing those 256 new variables, the diffusion of quadratic bits can be prevented in the third θ step, leaving only 4 quadratic lanes that would not affect the digest — although there is a quadratic lane $\Pi^3_{4,1,*}$ located in the first two planes, it doesn't matter because those 128 restricting equations for matching the 192-bit truncation only involve a_0, a_1, a_2 and a_3 (cf. equations (3)).

Parameters about the relinearization technique. Using the calculating way in [19], the number of variables is $v = 384 - 192 = 192$, the number of equations is $e = 320 + 128 + 64 = 512$ (the column sum equations in S_{I^1} are counted into this part), and the number of different quadratic terms has been revealed to be $q = 256$, satisfying $q \leq e - v$.

In conclusion, our quadratic structure can leave $384 - 256 = 128$ degrees of freedom but hold with a probability of only $(3/4)^{64} \approx 2^{-27}$. Entire column sum equations are given below. Due to lack of controllable column sums, *extra linear dependence* can hardly be applied. Thus under this structure, the final searching complexity is $2^{512-128+1} \times 2^{27} = 2^{412}$.

$$\begin{cases} I^0_{0,0,z} \oplus I^0_{0,1,z} \oplus I^0_{0,2,z} \oplus I^0_{0,3,z} \oplus I^0_{0,4,z} = S_{I^0}(0,z) \\ I^0_{1,0,z} \oplus I^0_{1,1,z} \oplus I^0_{1,2,z} \oplus I^0_{1,3,z} \oplus I^0_{1,4,z} = S_{I^0}(1,z) \\ I^0_{3,0,z} \oplus I^0_{3,1,z} \oplus I^0_{3,2,z} \oplus I^0_{3,3,z} \oplus I^0_{3,4,z} = S_{I^0}(3,z) \\ I^1_{0,0,z} \oplus I^1_{0,1,z} \oplus I^1_{0,2,z} \oplus I^1_{0,3,z} \oplus I^1_{0,4,z} = S_{I^1}(0,z) \end{cases} \tag{32}$$

Moreover, to generate those lanes of '0' and '1' in Θ^1, I^0 must satisfy[16]:

$$\begin{cases} I^0_{2,0,z} = I^0_{2,1,z} \oplus 1 = I^0_{2,4,z} \\ I^0_{4,0,z} = I^0_{4,1,z} \oplus 1 = I^0_{4,4,z} \\ S_{I^0}(1,z) \oplus S_{I^0}(3,z-1) = I^0_{2,4,z} \oplus 1 \\ S_{I^0}(3,z) \oplus S_{I^0}(0,z-1) = I^0_{4,4,z} \oplus 1 \\ S_{I^0}(4,z) \oplus S_{I^0}(1,z-1) = I^0_{0,4,z} \oplus 1 \end{cases} \tag{33}$$

Therefore, the column sums in S_{I^0} are all determined and the total number of controllable column sums is 64 (from $S_{I^1}(0,*)$).

[16] Please notice that $S_{I^0}(2,*)$ and $S_{I^0}(4,*)$ have actually been determined by conditional constants.

In summary, by rectifying the omission in relinearization and improving the quadratic structure, we decrease the searching complexity of preimage attack on 3-round Keccak-512 to 2^{412} (corresponding to 2^{424} 3-round Keccak calls). Since our quadratic structure contains restricted conditions in I^0, *allocating model* is necessary for the attack. The model parameters are $d_1 = 2^{64}$, $d_2 = 2^{412}$ and $d_r = 2^{64}$.

6 Conclusion

In this paper, we provide improved preimage analysis on round-reduced Keccak-384/512. The core of our preimage attacks is linear analysis. We inherit existing attack frameworks from previous researches and improve the preimage analysis results in two aspects:

I. By applying a new technology named *extra linear dependence*, we construct lane-level linear-dependent bit pairs between two output planes without spending degrees of freedom (by compressing the random space instead).

II. By changing the form of relinearization, we design an improved quadratic structure and reduce the number of variables in the final equation system.

As a result, the complexity of preimage attacks on 2-round Keccak-384/512 and 3-round Keccak-384/512 is decreased to $2^{39}/2^{204}$ and $2^{270}/2^{424}$ Keccak calls respectively, which are all the best known results so far. Remarkably, our work firstly performs practical preimage attacks on 2-round Keccak-384.

It is noted that our attack algorithm is still far from threatening the security of full-round Keccak. However, the idea of constructing extra linear dependence may be applicable to linear analysis on other cryptographic functions.

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponge functions. Submission to NIST (Round 3) (2011). https://sponge.noekeon.org/CSF-0.1.pdf
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: The Keccak reference, version 3.0. Submission to NIST (Round 3) (2011). https://keccak.noekeon.org/Keccak-reference-3.0.pdf
3. Bernstein, D.J.: Second preimages for 6 (7? (8??)) rounds of Keccak? NIST Mailing List (2010). https://ehash.iaik.tugraz.at/uploads/6/65/NIST-mailing-list_Bernstein-Daemen.txt
4. Morawiecki, P., Srebrny, M.: A SAT-based preimage analysis of reduced KECCAK hash functions. Information Processing Letters **113**(10-11), 392–397 (2013). https://doi.org/10.1016/j.ipl.2013.03.004
5. Morawiecki, P., Pieprzyk, J., Srebrny, M.: Rotational cryptanalysis of round-reduced Keccak. In: Moriai, S. (eds.) FSE 2013. LNCS, vol. 8424, pp. 241–262. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-662-43933-3_13
6. Dinur, I., Dunkelman, O., Shamir, A.: New attacks on Keccak-224 and Keccak-256. In: Canteaut, A. (eds.) FSE 2012. LNCS, vol. 7549, pp. 442–461. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34047-5_25

7. Qiao, K., Song, L., Liu, M., Guo, J.: New collision attacks on round-reduced Keccak. In: Coron, J.S., Nielsen, J. (eds.) EUROCRYPT 2017, Part III. LNCS, vol. 10212, pp. 216–243. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-56617-7_8
8. 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, Heidelberg (2017). https://doi.org/10.1007/978-3-319-63715-0_15
9. Aumasson, J.P., Meier, W.: Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. NIST Mailing List (2009). https://131002.net/data/papers/AM09.pdf
10. Boura, C., Canteaut, A., De Cannière, C.: Higher-order differential properties of Keccak and Luffa In: Joux, A. (eds.) FSE 2011. LNCS, vol. 6733, pp. 252–269. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21702-9_15
11. Duan, M., Lai, X.: Improved zero-sum distinguisher for full round Keccak-f permutation. Cryptology ePrint Archive, Report 2011/023 (2011). https://eprint.iacr.org/2011/023
12. Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In: 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
13. 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. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 259–288. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-56614-6_9
14. Song, L., Guo, J., Shi, D., Ling, S.: New MILP modeling: improved conditional cube attacks on Keccak-based constructions. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 65–95. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-030-03329-3_3
15. Guo, J., Liu, M., Song, L.: Linear structures: applications to cryptanalysis of round-reduced Keccak. In: Cheon, J., 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
16. Li, T., Sun, Y.: Preimage attacks on round-reduced Keccak-224/256 via an allocating approach. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 556–584. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-17659-4_19
17. Rajasree, M.S.: Cryptanalysis of round-reduced KECCAK using non-linear structures. In: Hao, F., Ruj, S., Sen Gupta, S. (eds.) INDOCRYPT 2019. LNCS, vol. 11898, pp. 175–192. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-35423-7_9
18. He, L., Lin, X. Yu, H.: Improved preimage attacks on 4-round Keccak-224/256. IACR Transactions on Symmetric Cryptology **2021**(1), 217–238 (2021). https://doi.org/10.46586/tosc.v2021.i1.217-238
19. Liu, F., Isobe, T., Meier, W., Yang, Z.: Algebraic Attacks on Round-Reduced Keccak. In: Baek, J., Ruj, S. (eds.) ACISP 2021. LNCS, vol. 13083, pp.91–110. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-90567-5_5
20. Kumar, R., Mittal, N., Singh, S.: Cryptanalysis of 2 round Keccak-384. In: Chakraborty, D., Iwata, T. (eds.) Progress in Cryptology – INDOCRYPT 2018.

LNCS, vol. 11356, pp. 120–133. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-030-05378-9_7

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 (2015). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf