IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2021, No. 1, pp. 217–238. DOI:10.46586/tosc.v2021.i1.217-238

Improved Preimage Attacks on 4-Round Keccak-224/256

Le He, Xiaoen Lin and Hongbo Yu

Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
{he-l17,linxe17,yuhongbo}@mail.tsinghua.edu.cn

Abstract. This paper provides an improved preimage attack method on standard 4-round Keccak-224/256. The method is based on the work pioneered by Li and Sun, who design a linear structure of 2-round Keccak-224/256 with 194 degrees of freedom left. By partially linearizing 17 output bits through the last 2 rounds, they finally reach a complexity of $2^{207}/2^{239}$ for searching a 4-round preimage. Yet under their strategy, those 17 bits are regarded as independent bits and the linearization costs a great amount of freedom. Inspired by their thoughts, we improve the partial linearization method where multiple output bits can reuse some common degrees of freedom. As a result, the complexity of preimage attack on 4-round Keccak-224/256 can be decreased to $2^{192}/2^{218}$, which are both the best known theoretical preimage cryptanalysis so far. To support the theoretical analysis, we apply our strategy to a 64-bit partial preimage attack within practical complexity. It is remarkable that this partial linearization method can be directly applied if a better linear structure with more freedom left is proposed.

Keywords: Keccak · SHA-3 · Preimage attack · Linear structure

1 Introduction

SHA family are one of the most widely used hash functions in cryptography and have been standardized as the Federal Information Processing Standards (FIPS) by the National Institute of Standards and Technology (NIST). Till now, there are three generations of SHA standards. SHA-1, designed by the National Security Agency (NSA) in 1995, has been proved to be insecure in collision resistance by Wang et al. in 2005 [WY05]. The SHA-2 series, also designed by NSA in 2001, are still undetermined in security due to their resemblance with SHA-1. Because of this, NIST decided to launch a public competition in 2008 to find a new hash function standard. Through four years of intensive scrutiny, the Keccak function won the competition among 64 initial submissions and was finally standardized as SHA-3 in 2015 [oST15].

The Keccak function, which was designed by Bertoni et al. [BDPV11a, BDPV11b], adopts a new iterative structure named *sponge* instead of the previous Merkle-Damgard structure of SHA-1 and SHA-2. The designers have proved [BDPV08] that the sponge construction has the characteristics of indifferentiability and collision resistance if its inner permutation Keccak-f can be regarded as a pseudo-random permutation. Although Keccak-f cannot be proved to be pseudo-random, the hybrid permutation consisting of 24 rounds of iteration still shows high resistance against any kind of cryptanalysis.

Since Keccak was publicly available in 2008, numerous researches have been conducted from public community. About the pseudo-randomness of Keccak-f, there are mainly two types of distinguishers: differential characteristic distinguishers and zero-sum distinguishers. In [DGPW12], Duc et al. made use of multiple 5-round differential paths and derived

Received: 2020-11-23 Accepted: 2021-01-23 Published: 2021-03-19

an 8-round distinguisher with complexity of about 2^{491}. The zero-sum attack was firstly proposed by Aumasson et al. in 2009 [AM09], reaching a practical attack for 9 rounds and a birthday attack (bounded by 2^{800}) for 15 rounds. In 2011, Boura et al. [BCC11] proposed a full-round zero-sum distinguisher with complexity of 2^{1590} (improved by Duan et al. [DL11] with complexity of 2^{1579}). As for collision attacks, most results depend on the differential trails. In [DV12], Daemen et al. analyzed the differential properties of Keccak and Kölbl et al. went further to study the differential properties of Keccak-f[800] and Keccak-f[1600] in [KMNS13]. In 2012, Dinur et al. [DDS12] proposed a new technology named *target difference algorithm*. Using this technology, they obtained collisions for 4-round Keccak-224/256 within practical complexities. After that, Qiao, Song, Guo et al. [GLL+20, QSLG17, SLG17] improved the technology and finally they were able to construct practical collisions for up to 5-round Keccak-224/256. Besides these, there are also attacks in other security settings. We would not list them here all since they are less relevant to our work.

In this paper, we mainly focus on preimage attacks. Although there exist some other kinds of preimage cryptanalysis [Ber10, MPS13, MS13], most preimage attacks depend on the linear characteristics of Keccak, especially of its nonlinear S-box. In [DMP+15], Dinur et al. proposed an idea of keeping the first round linear and used it to analyze the security of keyed-variant Keccak. Guo et al. [GLS16] further developed this idea by constructing a 3-round linear structure of Keccak-f and obtained a practical distinguisher for up to 11 rounds. Moreover, they adopted a similar 2-round structure and reached good results on searching a preimage for 3-round Keccak, or 4-round Keccak with low parameters such as Keccak Challenge and SHAKE128. However, when applied to standard 4-round Keccak-224/256, the linear structure will be severely affected by the restrictions in the starting state (the last several lanes must be all '0') and thus cannot perform well. In 2019, Li and Sun [LS19] proposed a 2-block model to loosen the restrictions. Under their model, the starting state of the second message block (which is also the ending state of the first message block) only needs to meet 129/193 conditions (for Keccak-224/256), instead of 448/512 all '0'. Therefore, more degrees of freedom can be left in their structure after 2-round entire linearization. Using the new linear structure as well as the new attack model, they succeeded in constructing the first practical preimage for 3-round Keccak-224 and their attack method also performed well on 4-round Keccak-224/256. All those preimage cryptanalysis on 4-round Keccak-224/256 are summarized in **Table 1**.

Table 1: Summary of preimage cryptanalysis on 4-round Keccak-224/256.

Round	Instance	Complexity	Reference
4	Keccak-224	2^{221}	[MPS13]
4	Keccak-256	2^{252}	
4	Keccak-224	2^{213}	[GLS16]
4	Keccak-256	2^{251}	
4	Keccak-224	1st block: 2^{129a} 2nd block: 2^{207b}	[LS19]
4	Keccak-256	1st block: 2^{193a} 2nd block: 2^{239b}	
4	Keccak-224	1st block: 2^{129a} 2nd block: 2^{192b}	This Paper
4	Keccak-256	1st block: 2^{193a} 2nd block: 2^{218b}	

[a] Corrected: when considering the message padding of Keccak, the complexity of the 1st block should be increased by 2^2.
[b] Note: those results do not include the complexities for solving the linear equation system (with a factor $O(n^3)$ where n is the number of linear equations).

Our contributions. Our work is mainly about the last 2 rounds of 4-round Keccak-224/256. In the third round, we propose an efficient partial linearization strategy where

multiple bits can reuse some common degrees of freedom. Under this new strategy, 56 bits located at specific sites can be linearized within 162 degrees of freedom, passing through the third S-box layer and reaching the fourth S-box layer. In the fourth round, we reveal the relation between the 224/256 output bits and the 56 linearized sites. We thoroughly analyze the situations where a linearized site works or not and calculate the expected gain of all 56 sites in total under a random digest. Combining our partial linearization and our expectation analysis with the linear structure inherited from [LS19], a complete preimage attack algorithm for 4-round Keccak-224/256 is constructed with complexity of $2^{192}/2^{218}$. To support our analysis, we present a partial pseudo preimage (the second message block) for 4-round Keccak matching 64 bits of the digest, with practical complexity of only 2^{33}.

Organization. The paper starts with some preliminaries and notations about Keccak in Section 2. In Section 3, we give a brief discussion about Li and Sun's work, especially about their 2-round linear structure. The following Section 4 introduces our partial linearization strategy and our expected gain analysis on 4-round Keccak-256. Section 5 is similar for 4-round Keccak-224. Experimental results of 64-bit partial pseudo preimage attack are presented in Section 6 and conclusions are summarized in Section 7.

2 Preliminaries

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

2.1 Sponge Construction

The Keccak hash function adopts the sponge construction, as depicted in **Figure 1**. It involves three parameters r, c, ℓ and a permutation Keccak-$f[b]$ with $b = r + c$, which will be further described in the next section.

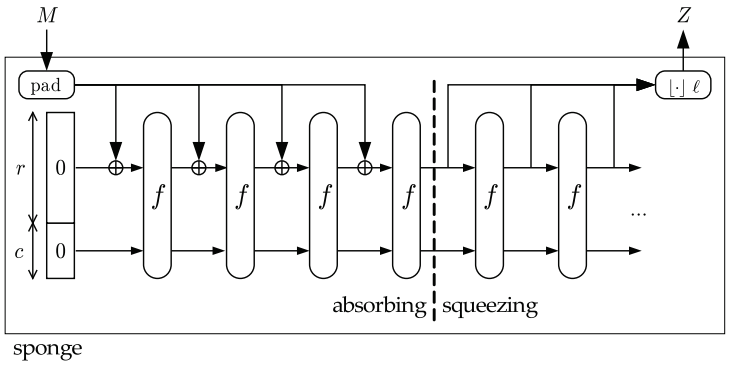

Figure 1: The sponge construction.

This construction processes a message in two phases—absorbing phase and squeezing phase. In the absorbing phase, the message M (after padding) is split into r-bit blocks firstly. Beginning with a b-bit all '0' IV, the first r bits are XORed with the first message block, followed with an execution of Keccak-f. Similar steps are repeated until all message blocks are processed. Then it comes to the squeezing phase. The sponge construction first outputs an r-bit block, then mixes its state by executing Keccak-f, and outputs another r-bit block, repeating until the digest length meets the requirement. Finally, the digest is truncated to the first ℓ bits.

Since the target preimage is only 2-block in our attack model and ℓ is much smaller than r in Keccak-224/256, we just need to focus on the specific sponge construction with Keccak-f executed exactly twice.

2.2 Keccak-f Permutation

The core of Keccak-$f[b]$ is its b-bit inner state. The designers provided seven Keccak-f permutations in [BDPV11b] with $b \in \{25, 50, 100, 200, 400, 800, 1600\}$. NIST finally chose $b = 1600$ as SHA-3 standard. In this paper, we also consider the case of $b = 1600$ only.

In the case of $b = 1600$, the b-bit inner state can be organized as $5 * 5$ 64-bit lanes like **Figure 2**. Each bit is denoted as $A_{x,y,z}$. The triple (x, y, z) denote the indices of a bit, where x varies from 0 to 4, y varies from 0 to 4, and z varies down from 63 to 0 (counting from the most significant bit) as the arrows in **Figure 2** show. The r-bit part of the b-bit state is in order of $A_{0,0,0} \sim A_{0,0,63}, A_{1,0,0} \sim A_{1,0,63}, \ldots, A_{4,0,0} \sim A_{4,0,63}, A_{0,1,0} \sim A_{0,1,63} \cdots$

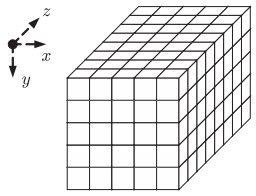

Figure 2: The Keccak-f state.

The Keccak-f permutation 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:

$$
\begin{aligned}
\theta &: A_{x,y,z} = A_{x,y,z} \oplus \bigoplus\nolimits_{i=0\sim4} (A_{x-1,i,z} \oplus A_{x+1,i,z-1}) \\
\rho &: A_{x,y,z} = A_{x,y,(z-r_{x,y})} \\
\pi &: A_{x,y,z} = A_{x+3y,x,z} \\
\chi &: A_{x,y,z} = A_{x,y,z} \oplus (A_{x+1,y,z} \oplus 1) \cdot A_{x+2,y,z} \\
\iota &: A_{0,0,z} = A_{0,0,z} \oplus RC_z
\end{aligned}
$$

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

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 instance can be denoted as Keccak$[r, c, \ell]$ with capacity c, bitrate r and digest length ℓ. In [oST15], NIST standardized several SHA-3 versions that have $r = 1600 - 2\ell$ and $c = 2\ell$, where $\ell \in \{224, 256, 384, 512\}$. Therefore, we can use Keccak-ℓ or SHA-3-ℓ to denote a SHA-3 version for short.

In this paper, we would not present those details such as the padding rule, the message reversing and the big-endian byte order. We just notice that the padding rule will bring

an extra 2-bit (4-bit for SHA-3, instead) restriction and the last 32 bits of the 224-bit digest are corresponding to $z = 0 \sim 31$ of the fourth lane. For more details of Keccak and SHA-3, please refer to [oST15].

2.4 Notations

From this section onwards, we will no longer use the capital letter A to denote the inner state, since it cannot accurately show its execution process. Instead, we will use capital Greek letters (in $\{\Theta, P, \Pi, X, I\}$) with a superscript (from 1 to 4) to express the state exactly **after** the corresponding step is executed. For example, Π^3 denotes the state after π in the third round, and X^4 denotes the state after χ in the fourth round.

The starting state of the first Keccak-f is denoted as IV, and the starting state of the second Keccak-f is denoted as H. The 2-block message is denoted as $M_1 M_2$. The digest of 4-round Keccak-224/256 is truncated from I^4. Notice that ι^{-1} can be easily applied to the digest, recovering corresponding 224/256 bits of X^4. In this paper, "digest" means those bits in I^4 and "output" means those bits in X^4. If attackers are able to match the output, the digest can also be easily matched.

To avoid ambiguity, we will always use three indices in subscript to denote a part of the inner state. However, we may use "$*$" to indicate all legal values. For example, $\Pi^3_{*,y,z}$ is a 5-bit row, $\Pi^3_{x,*,z}$ is a 5-bit column, $\Pi^3_{x,y,*}$ is a 64-bit lane, and $\Pi^3_{*,*,z}$ is a $5 * 5$ slice. If we omit the subscript, it indicates the 1600-bit whole state (just like the notations above). And if we omit the superscript, it means the statement is general in each round.

In the rest of the paper, we may use the word "site" to describe a certain bit in Π^4. We say a site $\Pi^4_{x,y,z}$ is linearized, if it can be expressed by an XOR sum of several bits from H and M_2 (through 3 nonlinear S-box layers).

3 Overview

This section gives an overview about our preimage attack on 4-round Keccak-224/256. First, we briefly recall the 2-round linear structure designed by Li and Sun. Second, we present the outline of our attack model. Thirdly, we take a preview on the strategy of partial linearization and how linearized sites work under different outputs.

3.1 2-Round Linear Structure via an Allocating Approach

In [LS19], the authors provided a 2-round linear structure with 194 degrees of freedom left via an allocating approach. The structure is depicted in **Figure 3**.

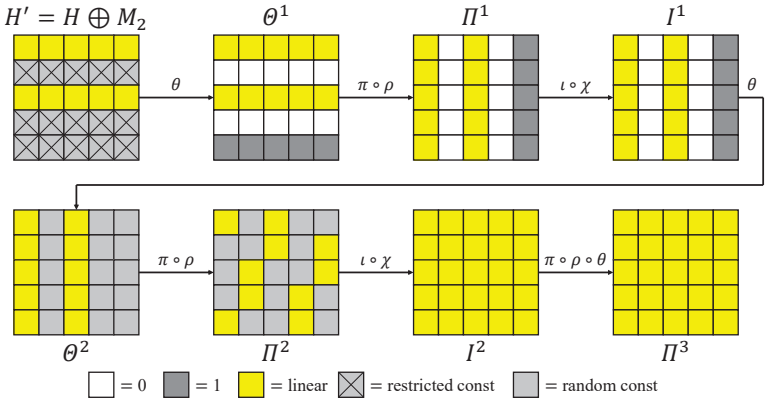

Figure 3: The 2-round linear structure in [LS19].

This structure starts with the ending state H of the first Keccak-f by absorbing M_1. After XORed with M_2, if $H' = H \oplus M_2$ meets $H'_{x,1,z} = H'_{x,3,z} = H'_{x,4,z} \oplus 1$, the first θ step can lead to $\Theta^1_{x,1,z} = \Theta^1_{x,3,z} = 0$ and $\Theta^1_{x,4,z} = 1$ by controlling $H'_{x-1,0,z} \oplus H'_{x-1,2,z} \oplus H'_{x+1,0,z-1} \oplus H'_{x+1,2,z-1} = H'_{x,4,z} \oplus 1$. These constants can keep the number of uncertain bits at the least level after the first nonlinear χ step. In the second θ step, those column sums $I^1_{x,0,z} \oplus I^1_{x,1,z} \oplus I^1_{x,2,z} \oplus I^1_{x,3,z} \oplus I^1_{x,4,z}$ (denoted as $S_{I^1}(x,z)$) will be randomly set to prevent the diffusion of uncertain bits. Finally the structure can pass through the second χ step and reach the third χ step, which is a 2-round linear structure.

Theorem 1. *If the starting state meets* $H'_{x,1,z} = H'_{x,3,z} = H'_{x,4,z} \oplus 1$ *and* $\bigoplus_{x,z} H'_{x,4,z} = 0$, ***Figure 3*** *can construct a 2-round linear structure with 194 degrees of freedom left.*

In [LS19] the authors have actually proven this theorem. Here we just briefly explain their ideas. Θ^1 can be obtained by setting $H'_{x-1,0,z} \oplus H'_{x-1,2,z} \oplus H'_{x+1,0,z-1} \oplus H'_{x+1,2,z-1} = H'_{x,4,z} \oplus 1$, which composes 320 linear equations in total. Among these equations, 160 of them can infer $\bigoplus_{x,z}(H'_{x,0,z} \oplus H'_{x,2,z})$ to equal a certain value, and the other 160 of them also infer $\bigoplus_{x,z}(H'_{x,0,z} \oplus H'_{x,2,z})$ to equal a certain value. Therefore, this part will return 1 degree of freedom and $\bigoplus_{x,z} H'_{x,4,z} = 0$ is required to avoid contradictions. In the second part of 128 column sums, since the sum of all 640 variables is certain, 1 column sum can be inferred from the other 127 column sums, which returns 1 degree of freedom. Overall, this linear structure leaves $640 - 320 - 128 + 2 = 194$ degrees of freedom.

As for the complexity of constructing a qualified H', notice that the message block M_2 can adjust 5 lanes of $H'_{x,1,*}$ and 2/3 lanes of $H'_{x,3,*}$ in Keccak-256/224, leaving 3/2 lanes only. Therefore, 192/128 equations of $H_{x,3,z} = H_{x,4,z} \oplus 1$ need to be met in H and the complexity is $2^{193}/2^{129}$ (including $\bigoplus_{x,z} H_{x,4,z} = 0$). When considering the message padding of Keccak, extra 2 bits (4 bits for SHA-3) are restricted in $H'_{3,1,*}/H'_{3,2,*}$, and thus the complexity should be increased by 2^2, becoming $2^{195}/2^{131}$.

3.2 Outline of 4-Round Preimage Attack

Figure 4 shows the outline of our 4-round preimage attack. The figure takes Keccak-256 as an example, while it is similar for Keccak-224.

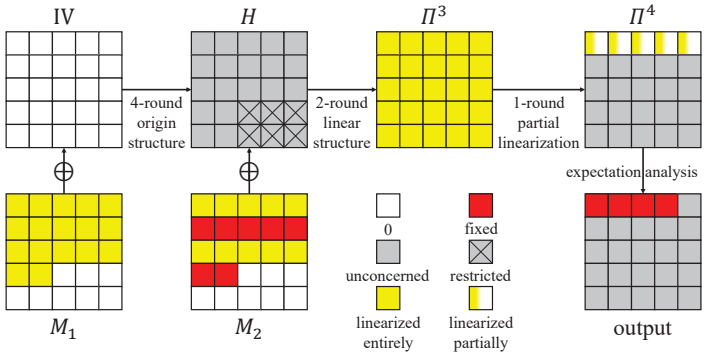

Figure 4: The outline of 4-round preimage attack.

Attackers first find out a message block M_1 to make H meet those 195 conditions. With H determined, 7 lanes are fixed and 10 lanes are varied in message block M_2. By setting 446 linear equations, the inner state after the third π step is still entirely linearized with 194 degrees of freedom left. Then attackers apply a partial linearization strategy to pass through the third χ step in order that several sites in Π^4 can be linearized. It is expected that those linearized sites can bring gains in matching the output, which corresponds to

the state just one step behind. If attackers fail in matching the output, they can reset all random values and find a new solution of the linear equation system, leaving H and M_1 unchanged. This process will be repeated until the output is matched. Finally, the 2-block preimage is $M = M_1 M_2$.

Apparently, the main problems of this outline are how to linearize as many sites as possible, and how to analyze the expected gain of those linearized sites. Before taking a preview on these problems, we first discuss how [LS19] deals with them.

3.3 Basic Strategy of Partial Linearization

Actually in [LS19], the authors haven't proposed the concept of "partial linearization" or "expectation analysis", but the idea is the same in a way. Their strategy is to control a single bit by spending 11 degrees of freedom, as shown in **Figure 5**.

In order to control a single bit in Π^4, which corresponds to a bit in Θ^4, 11 bits in X^3 need to be linearized. Notice that Π^3 is still entirely linear because of the 2-round linear structure. Then to linearize a bit $X^3_{x,y,z} = \Pi^3_{x,y,z} \oplus (\Pi^3_{x+1,y,z} \oplus 1) \cdot \Pi^3_{x+2,y,z}$, just ensure either $\Pi^3_{x+1,y,z}$ or $\Pi^3_{x+2,y,z}$ is a constant. By randomly setting 10 bits in Π^3 to be '0' or '1' (just like **Figure 5**), the target bit in Π^4 can be successfully linearized.

Including the equation of the target bit itself, it costs 11 degrees of freedom in total to control a single bit in Π^4. Within 187 out of 194 degrees of freedom, 17 bits of Π^4 are finally controlled. These bits can be sited at any position to the needs of attackers.

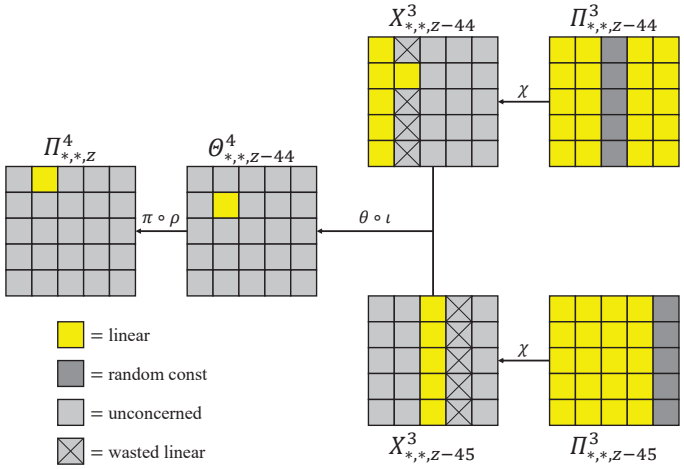

Figure 5: Control a single bit by spending 11 degrees of freedom.

On the other hand, some bits in $\Pi^4_{*,0,*}$ can be directly recovered from the 224/256 output bits. For example, if one row of S-box output is $X_{*,0,z} = 1001?$ ("?" denotes the uncertain bit from the fifth lane), the input may be $\Pi_{*,0,z} = 10000$ or $\Pi_{*,0,z} = 11011$ (denoted as 1?0?? for short) with 2 recovered bits. Suppose attackers control $\Pi^4_{0,0,z} = 1$ to match the output row 1001?. Then the other 4 bits, which are all the sum of 11 bits in I^3, can be regarded as independent Bernoulli variables with probability of 1/2 as long as any of those 11 bits is uncontrolled. Thus among all 16 cases, 2 cases of 10000 and 11011 can lead to 1001?, with probability of 2^{-3} rather than 2^{-4}. From this perspective, 1 recovered bit in $\Pi^4_{*,0,*}$ can bring a gain of 2^1 in matching the output. Although there may be some bad output rows such as 1000? $\leftarrow$?????, attackers can almost always find 17 recovered bits to control. Therefore, the complexity decrease is 2^{17} in [LS19] for 4-round Keccak-224/256.

The advantage of this strategy is that attackers are able to control any site of $\Pi^4_{*,0,*}$. In other words, the average gain of a single site is 2^1, since attackers can always choose

those recovered sites. However, in **Figure 5** those 10 constant bits actually linearize 20 bits in X^3. This strategy directly wastes 9 of them, which means the average usage of a single linear bit is only about 1/2. In the next section, we will take a preview on a new strategy with average usage about 2 and average gain about $2^{0.5}$ to $2^{0.75}$, which can be much more efficient.

3.4 Preview on Freedom Reuse Strategy

In **Figure 5**, a single bit in Π^4 is linearized by 10 constants in Π^3, leading to 20 linear bits with 9 of them wasted. Actually these 9 bits can be used for linearizing another bit in Π^4, just by adding 1 constant. The method is presented in **Figure 6**.

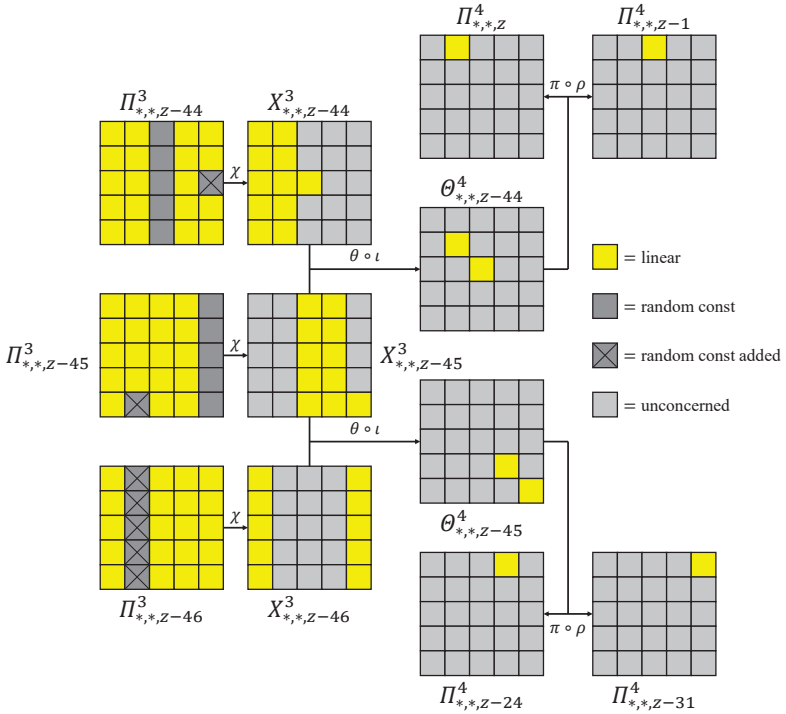

Figure 6: Multiple bits reuse common degrees of freedom.

The origin $\Pi^4_{1,0,z}$ (corresponding to $\Theta^4_{1,1,z-44}$) is determined by two columns $X^3_{0,*,z-44}$ and $X^3_{2,*,z-45}$ with one solitary bit $X^3_{1,1,z-44}$. These bits can be linearized by setting 10 constants, as shown in **Figure 6**. Notice that all the 5 bits in column $\Theta^4_{1,*,z-44}$ actually have been linearized, but only 1 bit can reach $\Pi^4_{*,0,*}$ (passing through the step π). These 10 constants also linearize another two columns $X^3_{1,*,z-44}$ and $X^3_{3,*,z-45}$. By just adding 1 constant to linearize $X^3_{2,2,z-44}$, the bit $\Pi^4_{2,0,z-1}$ (corresponding to $\Theta^4_{2,2,z-44}$) can also be linearized. The freedom reuse strategy can continue by adding 6 constants (1 in $\Pi^3_{*,*,z-45}$ and 5 in $\Pi^3_{*,*,z-46}$), generating another 2 linearized bits in $\Pi^4_{*,0,*}$.

However, the positions of those linearized bits are relatively fixed under this strategy. As the example in **Figure 6** shows, if attackers start with $\Pi^4_{1,0,z}$, following linearized bits must be $\Pi^4_{2,0,z-1}$, $\Pi^4_{3,0,z-24}$, $\Pi^4_{4,0,z-31}$... **Figure 7** further reveals some laws about this strategy. It starts with 5 bits in column $\Pi^3_{x=x_s,*,z=z_s}$ and continues with 5 bits in column $\Pi^3_{x+2,*,z-1}$ as well as 1 solitary bit $\Pi^3_{x+2,x,z}$ (or $\Pi^3_{x+1,x,z}$), linearizing 2 sites in each step. After five steps, the constant column returns to x_s and all z-axes are rotated by an offset of 5, forming one cycle. It's seen that each constant column is related to 4 linearized sites and each solitary constant bit is related to only 1 linearized site.

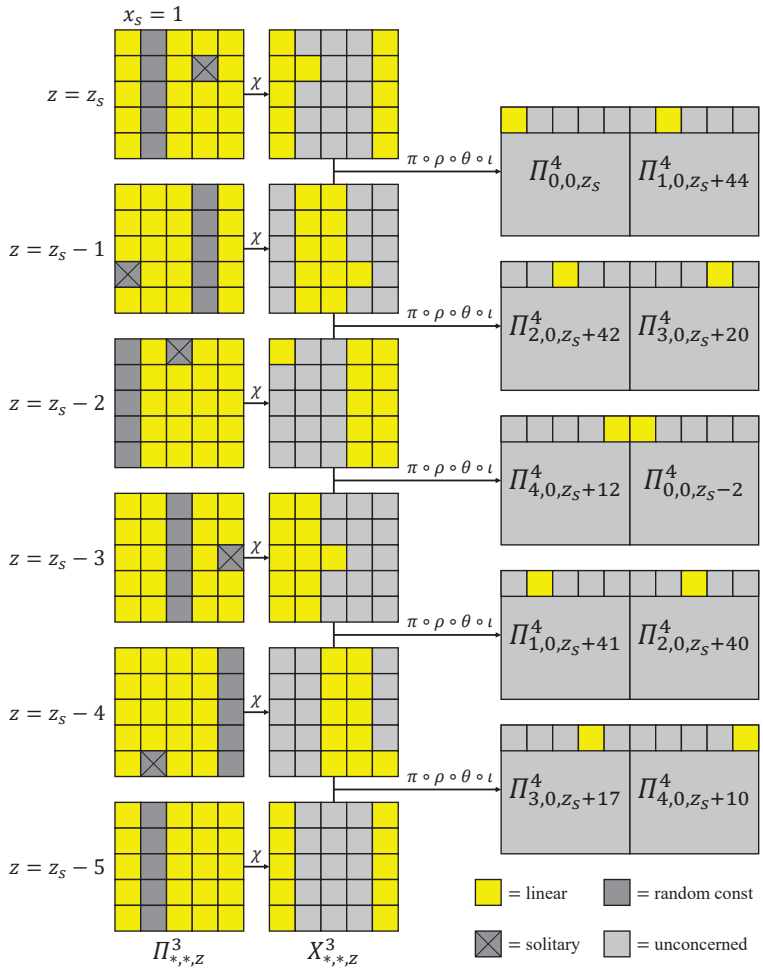

Figure 7: The freedom reuse strategy (one cycle).

Using this freedom reuse strategy, attackers can linearize $28 * 2 = 56$ bits by setting $5 + 28 * 6 = 173$ constants. Those 56 bits will all be located at specific sites once the starting column is chosen. Notice that there are 320 groups of linearized sites in total (by changing the starting column) and attackers are able to go through them all to search one with the greatest gain. However, we still need to analyze the expected gain of a certain group since attackers can only choose one group to build up the linear equation system.

Now we can take a preview on the expected gain analysis. The main principle is that no matter what the output is, we will always use the same group of linearized sites (just for analysis). Take $z_s = 0$ as an example. Suppose one output row is $X^4_{*,0,0} = 1001? \leftarrow 1?0??$. Then the linearized site $\Pi^4_{0,0,0}$ can bring a gain of 2^1 by setting the linear equation $\Pi^4_{0,0,0} = 1$ (this equation also costs 1 degree of freedom), which is a typical situation of a linearized site working. Suppose another output row is $X^4_{*,0,41} = 1000? \leftarrow?????$, which is the worst situation of all solitary bits not working. Then we can do nothing with the linearized site $\Pi^4_{1,0,41}$ expect neglecting the efforts we have made to linearize it. Finally, the gain of this certain group corresponds to the number of effective equations we can establish, which also depends on the number of recovered bits among all linearized sites. Although there may be some bad cases like outputs with lots of 1000?, the expected gain under a random digest can still reach a better result since the number of linearized sites is much greater than 17.

Table 3 shows those recovered bits under all 4-bit outputs, which supports a primary analysis for Keccak-256. Regarding all 56 sites as solitary bits, the expected number of recovered bits is (among those 64 groups starting from $\Pi^3_{1.*,z}$):

$$12 * 0.75 + 11 * 0.75 + 11 * 0.5 + 11 * 0.5 + 11 * 0 = 28.25$$

Under this analysis, the expected total gain of constructing a 4-round preimage for a random 256-bit digest can reach 2^{28}.

Table 3: Recovered bits under different cases.

Output	Input	Output	Input
0000?	?0?0?	1000?	?????
0001?	0101?	1001?	1?0??
0010?	?0???	1010?	?0?0?
0011?	101??	1011?	0011?
0100?	01?0?	1100?	1????
0101?	0001?	1101?	1?0??
0110?	01?0?	1110?	11???
0111?	0111?	1111?	111??
*Total: $\Pi^4_{0,0,z}$: 0.75 $\Pi^4_{1,0,z}$: 0.75 $\Pi^4_{2,0,z}$: 0.5 $\Pi^4_{3,0,z}$: 0.5 $\Pi^4_{4,0,z}$: 0			

*$\Pi^4_{x,0,z}$ corresponds to an input bit of the table. Among all 16 cases, 12 $\Pi^4_{0,0,z}$ are recovered without "?" and thus the total ratio is 0.75. Other ratios are similar.

However, for those linearized sites located at the same row, their efficiency can greatly increase compared with solitary bits from different rows. For example, although $X^4_{*,0,42} =$ 1000? $\leftarrow$?????, we can still get a gain of 2^1 by setting $\Pi^4_{0,0,42} \oplus \Pi^4_{2,0,42} = 1$ ($\Pi^4_{0,0,42}$ is also linearized in the group starting from $\Pi^3_{1.*,0}$). Detailed analysis will be provided in Section 4 and Section 5.

To sum up, this section has given an overview about our preimage attack method: by setting 173 constants in entirely linearized Π^3, 56 bits in Π^4 can be partially linearized, bringing some kinds of benefits in matching the digest. A common doubt is: how can we get a gain greater than 2^{21} since we have set 173 linear equations out of 194 degrees of freedom? Actually those constants need not cost 173 degrees of freedom, which refers to a technology named *zero coefficient*. Moreover, if some linearized sites do not work, we can delete the constants of related solitary bits, which refers to a technology named *freedom return*. These technologies will also be introduced in following sections.

4 Improved Preimage Attack on 4-Round Keccak-256

Preimage cryptanalysis on 4-round Keccak-256 is provided in this section. After introducing how to set 173 constants by spending 162 degrees of freedom, we will display all the details about 173 constant bits and 56 linearized sites. Then we further analyze the expected gain in 3 cases: rows containing 1, 2 or 3 linearized sites. In the third part, we discuss the technology of freedom return, which can make up the lack of freedom. It is finally concluded that attackers can set 38 effective equations with probability of 74.5%, obtaining a complexity decrease of 2^{38} under a random 256-bit digest.

4.1 How to Set 173 Constants within 162 Degrees of Freedom

We have introduced our freedom reuse strategy of linearizing 56 sites in $\Pi^4_{*,0,*}$ by setting 173 constants in Π^3. Those specific constant bits are summarized in **Table 4**.

Table 4: Details about 173 constant bits (example starting from $\Pi^3_{4,*,0}$).

step	1	2	3	4	5	6	7
column	$\Pi^3_{1,*,63}$	$\Pi^3_{3,*,62}$	$\Pi^3_{0,*,61}$	$\Pi^3_{2,*,60}$	$\Pi^3_{4,*,59}$	$\Pi^3_{1,*,58}$	$\Pi^3_{3,*,57}$
solitary	$\Pi^3_{1,4,0}$	$\Pi^3_{2,1,63}$	$\Pi^3_{0,3,62}$	$\Pi^3_{2,0,61}$	$\Pi^3_{4,2,60}$	$\Pi^3_{1,4,59}$	$\Pi^3_{3,1,58}$
step	8	9	10	11	12	13	14
column	$\Pi^3_{0,*,56}$	$\Pi^3_{2,*,55}$	$\Pi^3_{4,*,54}$	$\Pi^3_{1,*,53}$	$\Pi^3_{3,*,52}$	$\Pi^3_{0,*,51}$	$\Pi^3_{2,*,50}$
solitary	$\Pi^3_{0,3,57}$	$\Pi^3_{2,0,56}$	$\Pi^3_{4,2,55}$	$\Pi^3_{1,4,54}$	$\Pi^3_{3,1,53}$	$\Pi^3_{0,3,52}$	$\Pi^3_{2,0,51}$
step	15	16	17	18	19	20	21
column	$\Pi^3_{4,*,49}$	$\Pi^3_{1,*,48}$	$\Pi^3_{3,*,47}$	$\Pi^3_{0,*,46}$	$\Pi^3_{2,*,45}$	$\Pi^3_{4,*,44}$	$\Pi^3_{1,*,43}$
solitary	$\Pi^3_{4,2,50}$	$\Pi^3_{1,4,49}$	$\Pi^3_{3,1,48}$	$\Pi^3_{0,3,47}$	$\Pi^3_{2,0,46}$	$\Pi^3_{4,2,45}$	$\Pi^3_{1,4,44}$
step	22	23	24	25	26	27	28
column	$\Pi^3_{3,*,42}$	$\Pi^3_{0,*,41}$	$\Pi^3_{2,*,40}$	$\Pi^3_{4,*,39}$	$\Pi^3_{1,*,38}$	$\Pi^3_{3,*,37}$	$\Pi^3_{0,*,36}$
solitary	$\Pi^3_{3,1,43}$	$\Pi^3_{0,3,42}$	$\Pi^3_{2,0,41}$	$\Pi^3_{4,2,40}$	$\Pi^3_{1,4,39}$	$\Pi^3_{3,1,38}$	$\Pi^3_{0,3,37}$

The law of each step has been revealed in **Figure 7**, which is $\Pi^3_{x,*,z} \to \Pi^3_{x+2,*,z-1}$ with solitary bits $\Pi^3_{x,x-2,z+1} \to \Pi^3_{x+2,x,z}$. Including the starting column, those constants consist of 29 columns and 28 solitary bits. Although it is only an example starting from $\Pi^3_{4,*,0}$, other situations are actually similar, which shows a characteristic of rotational symmetry. Notice that except $\Pi^3_{2,1,63}$, all solitary bits have the form $\Pi^3_{x+2,x,z}$. We choose $\Pi^3_{2,1,63}$ instead of $\Pi^3_{3,1,63}$ because it can bring 1 degree of freedom, which refers to the technology of zero coefficient.

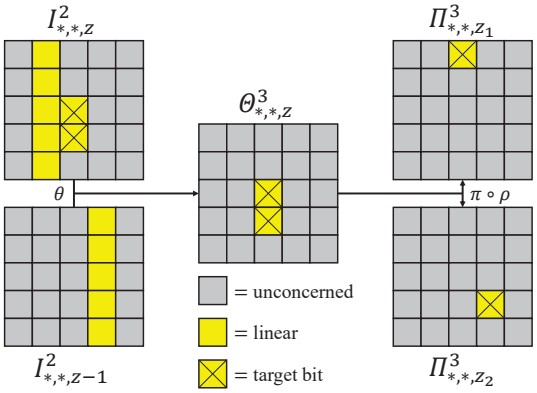

Figure 8: The idea of zero coefficient.

The idea of zero coefficient is shown in **Figure 8**. Passing through the 2-round linear structure, each single bit in I^2 to Π^3 can be expressed by a linear representation of the original 640 variables, which corresponds to a 640-dimensional 0-1 vector. This vector is called the *coefficient* of a bit. It is just the coefficient of the linear equation if we want to set the bit as a constant. The coefficients are still simple in I^2, while they become quite complex in Π^3 because each bit is the sum of 11 bits from I^2. It seems that 173 out of 1600 bits can hardly have linear correlations to bring extra degrees of freedom. However, for those pairs corresponding to identical column in Θ^3, they are XORed by the same group of 10 bits. If we further ensure the remaining 2 bits also have identical coefficients (as shown in **Figure 8**), then the corresponding bits in Π^3 can be linearly correlated and bring 1 degree of freedom. Actually the only possibility that 2 bits from identical column have identical coefficients is zero coefficient. In other words, they must be both '0' or '1' (unnecessary to be equal), not diffused by those 640 variables.

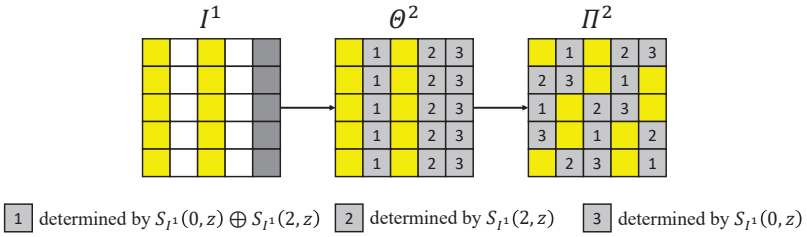

Figure 9: The relation between column sums and constant bits.

As for how to ensure a bit is zero-coefficient, it depends on the random column sum $S_{I^1}(x,z) = I^1_{x,0,z} \oplus I^1_{x,1,z} \oplus I^1_{x,2,z} \oplus I^1_{x,3,z} \oplus I^1_{x,4,z}$ $(x \in \{0,2\})$. Suppose a target bit is $I^2_{x,y,z}$. Then we can fix the corresponding column sum to control the constant bit $\Pi^2_{x+1,y,z}$ or $\Pi^2_{x+2,y,z}$ so that the effect of variable $\Pi^2_{x+2,y,z}$ or $\Pi^2_{x+1,y,z}$ can be offset by logic AND ($\Pi^2_{x,y,z}$ cannot be variable first). The relation between column sums and constant bits is revealed in **Figure 9**.

Take **Figure 8** as an example. If we want to ensure $I^2_{2,2,z}$ and $I^2_{2,3,z}$ are zero-coefficient bits, $\Pi^2_{3,2,z} = 1$ and $\Pi^2_{4,3,z} = 0$ are required separately. Then we need to fix corresponding column sums $S_{I^1}(0,z_1) = 0$ and $S_{I^1}(2,z_2) = 1$ (notice that $I^1_{*,4,*} = 1$) instead of random values. It is seen that 1 pair of zero-coefficient bits will expend 2 random values. Those random values are set to ensure that the attack algorithm can be repeatedly operated until a preimage is found. Actually, neglecting the zero coefficient technology, there exist $127 + 173 = 300$ random values. Those zero-coefficient pairs only expend a few of them, remaining enough space to search a preimage.

We have introduced the principle of zero coefficient technology. It is finally found that for $x_s = 4$, those 173 specific bits in Π^3, corresponding to 173 bits in Θ^3, contain 11 pairs that can both be ensured zero-coefficient. Therefore, those constants can be set within 162 degrees of freedom. All 11 pairs are summarized in **Table 5** with fixed column sums.

Table 5: Pairs of zero-coefficient bits (example with the starting column $\Pi^3_{4,*,0}$).

Pair	Constant Bit	Required Column Sum
$\Theta^3_{0,2,60}$	$\Pi^{3*}_{2,1,63}$	$S_{I^1}(2,35) = 1$
$\Theta^3_{0,3,60}$	$\Pi^3_{3,4,37}$	$^a S_{I^1}(0,50) \oplus S_{I^1}(2,50) = 0$
$\Theta^3_{2,2,3}$	$\Pi^{3*}_{2,0,46}$	$S_{I^1}(0,59) = 0$
$\Theta^3_{2,4,3}$	$\Pi^3_{4,1,0}$	$S_{I^1}(0,1) \oplus S_{I^1}(2,1) = 0$
$\Theta^3_{2,2,62}$	$\Pi^{3*}_{2,0,41}$	$S_{I^1}(0,54) = 0$
$\Theta^3_{2,4,62}$	$\Pi^3_{4,1,59}$	$S_{I^1}(0,60) \oplus S_{I^1}(2,60) = 0$
$\Theta^3_{2,3,42}$	$\Pi^3_{3,3,57}$	$S_{I^1}(2,50) = 1$
$\Theta^3_{2,4,42}$	$\Pi^3_{4,1,39}$	$S_{I^1}(0,40) \oplus S_{I^1}(2,40) = 0$
$\Theta^3_{2,3,47}$	$\Pi^3_{3,3,62}$	$S_{I^1}(2,55) = 1$
$\Theta^3_{2,4,47}$	$\Pi^3_{4,1,44}$	$^b S_{I^1}(0,45) \oplus S_{I^1}(2,45) = 0$
$\Theta^3_{3,0,8}$	$\Pi^3_{0,1,36}$	$S_{I^1}(0,58) = 0$
$\Theta^3_{3,1,8}$	$\Pi^3_{1,4,63}$	$^c S_{I^1}(2,44) = 1$
$\Theta^3_{4,0,29}$	$\Pi^3_{0,3,56}$	$S_{I^1}(0,49) \oplus S_{I^1}(2,49) = 0$
$\Theta^3_{4,3,29}$	$\Pi^3_{3,2,37}$	$S_{I^1}(0,2) = 0$
$\Theta^3_{4,0,34}$	$\Pi^3_{0,3,61}$	$S_{I^1}(0,54) \oplus S_{I^1}(2,54) = 0$
$\Theta^3_{4,3,34}$	$\Pi^3_{3,2,42}$	$S_{I^1}(0,7) = 0$

$\Theta^3_{4,0,25}$ $\Theta^3_{4,4,25}$	$\Pi^{3*}_{0,3,52}$ $\Pi^3_{4,0,39}$	${}^b S_{I^1}(0,45) \oplus S_{I^1}(2,45) = 0$ $S_{I^1}(2,34) = 1$
$\Theta^3_{4,0,30}$ $\Theta^3_{4,4,30}$	$\Pi^{3*}_{0,3,57}$ $\Pi^3_{4,0,44}$	${}^a S_{I^1}(0,50) \oplus S_{I^1}(2,50) = 0$ $S_{I^1}(2,39) = 1$
$\Theta^3_{4,0,35}$ $\Theta^3_{4,4,35}$	$\Pi^{3*}_{0,3,62}$ $\Pi^3_{4,0,49}$	$S_{I^1}(0,55) \oplus S_{I^1}(2,55) = 0$ ${}^c S_{I^1}(2,44) = 1$

*Those bits are from the solitary part in **Table 4**.
^{a,b,c}Identical column sums are involved without contradictions.

Similarly, this example can represent all 64 groups starting from $\Pi^3_{4,*,z_s}$ because of rotational symmetry. However, the result cannot hold for $x_s \neq 4$: when x_s changes, the corresponding bits in Θ^3 will greatly change (because offsets in ρ step relatively change). In other words, situations of $x_s \neq 4$ may bring even more than 11 degrees of freedom. Yet we would not be concerned with those situations for the reason that $x_s = 4$ can actually bring greater gains in expectation compared with other selections. Detailed analysis is provided in the next section.

4.2 Gain Analysis of 56 Linearized Sites

Those 56 linearized sites in $\Pi^4_{*,0,*}$ are summarized in **Table 6**. It is still an example of $z_s = 0$, while it can represent other situations of z_s: when z_s changes, just rotate all entries of the table with the same offset.

Table 6: 56 linearized sites (example with the starting column $\Pi^3_{4,*,0}$).

z	$\Pi^4_{x,0,z}$	z	$\Pi^4_{x,0,z}$	z	$\Pi^4_{x,0,z}$	z	$\Pi^4_{x,0,z}$
0	\	16	$\Pi^4_{2,0,z}\,\Pi^4_{3,0,z}$	32	\	48	$\Pi^4_{0,0,z}$
1	$\Pi^4_{3,0,z}\,\Pi^4_{4,0,z}$	17	\	33	$\Pi^{4*}_{1,0,z}$	49	\
2	\	18	$\Pi^{4*}_{1,0,z}$	34	$\Pi^{4*}_{2,0,z}$	50	\
3	\	19	$\Pi^{4*}_{2,0,z}\,\Pi^4_{3,0,z}$	35	$\Pi^4_{1,0,z}$	51	$\Pi^{4*}_{0,0,z}$
4	$\Pi^{4*}_{3,0,z}\,\Pi^{4*}_{4,0,z}$	20	$\Pi^4_{1,0,z}$	36	$\Pi^4_{2,0,z}$	52	\
5	\	21	$\Pi^4_{2,0,z}\,\Pi^4_{3,0,z}$	37	\	53	$\Pi^4_{0,0,z}\,\Pi^{4*}_{4,0,z}$
6	$\Pi^4_{3,0,z}\,\Pi^4_{4,0,z}$	22	\	38	$\Pi^4_{0,0,z}\,\Pi^{4*}_{1,0,z}$	54	\
7	\	23	$\Pi^{4*}_{1,0,z}$	39	$\Pi^{4*}_{2,0,z}$	55	$\Pi^4_{4,0,z}$
8	\	24	$\Pi^{4*}_{2,0,z}$	40	$\Pi^4_{1,0,z}$	56	$\Pi^{4*}_{0,0,z}$
9	$\Pi^4_{3,0,z}\,\Pi^{4*}_{4,0,z}$	25	$\Pi^4_{1,0,z}$	41	$\Pi^4_{0,0,z}\,\Pi^4_{2,0,z}$	57	\
10	\	26	$\Pi^4_{2,0,z}$	42	\	58	$\Pi^4_{0,0,z}\,\Pi^{4*}_{3,0,z}\,\Pi^{4*}_{4,0,z}$
11	$\Pi^4_{3,0,z}\,\Pi^4_{4,0,z}$	27	\	43	$\Pi^4_{0,0,z}\,\Pi^4_{1,0,z}$	59	\
12	\	28	$\Pi^{4*}_{1,0,z}$	44	\	60	$\Pi^4_{3,0,z}\,\Pi^4_{4,0,z}$
13	\	29	$\Pi^{4*}_{2,0,z}$	45	\	61	$\Pi^{4*}_{0,0,z}$
14	$\Pi^4_{3,0,z}\,\Pi^{4*}_{4,0,z}$	30	$\Pi^4_{1,0,z}$	46	$\Pi^4_{0,0,z}$	62	\
15	\	31	$\Pi^4_{2,0,z}$	47	\	63	$\Pi^4_{0,0,z}\,\Pi^{4*}_{3,0,z}\,\Pi^{4*}_{4,0,z}$

*Those bits are relevant to the solitary part in **Table 4**, and meanwhile irrelevant to the solitary part appearing in **Table 5**.

It is concluded that those 56 bits are distributed in 38 out of 64 rows, which can be divided into 3 classes:

– Rows containing 1 linearized bit, 22 rows in total.
– Rows containing 2 linearized bits, 14 rows in total.
– Rows containing 3 linearized bits, 2 rows in total.

Based on **Table 6**, we can provide a detailed analysis on the expected gain. In Section 3, we have mentioned that the total gain of those 56 linearized sites is corresponding to the number of effective equations: 1 effective equation can exclude half of possible cases (e.g., 16 out of 32) and meanwhile conserve both 2 matched inputs in the remaining cases. Therefore, our analysis mainly focuses on the number of effective equations under 3 classes.

- Rows containing 1 linearized bit.

The equation will be effective if and only if the linearized bit exactly matches those recovered bits. For example, under the output row 1001? ← 1?0??, those $\Pi^4_{0,0,z}$ and $\Pi^4_{2,0,z}$ can bring 1 effective equation and the others cannot. In **Table 3**, we have revealed the probability of one linearized bit matching the recovered bits. According to the result, the expected total number of effective equations from this part is:

$$5 * 0.75 + 9 * 0.75 + 7 * 0.5 + 0 * 0.5 + 1 * 0 = 14$$

- Rows containing 2 linearized bits.

Apparently, there are three situations: both linearized bits can match, exactly one bit can match, and neither of them can match. The second situation is just the same as one solitary bit: setting equation on the matched one and ignoring the other one can bring a gain of 2^1. If 2 linearized bits both match, they can work together because 2 effective equations can exclude three quarters of possible cases and bring a gain of 2^2. It is worth mentioning that the last situation also leads to 1 effective equation. The example is still 1001? ← 1?0??. Since $\Pi^4_{3,0,z}$ and $\Pi^4_{4,0,z}$ are both uncertain in 10000 and 11011, their sum must be certain and the equation of $\Pi^4_{3,0,z} \oplus \Pi^4_{4,0,z} = 0$ can bring a gain of 2^1.

In summary, each row in this class can at least lead to 1 effective equation. Another effective equation can be set if and only if 2 linearized bits both match those recovered bits. As for the probability of 2 bits simultaneously matching, please refer to **Table 7**.

Table 7: Probability of 2 bits matching.

	$x_1 = 0$	$x_1 = 1$	$x_1 = 2$	$x_1 = 3$	$x_1 = 4$
$x_2 = 0$	\	9/16	8/16	6/16	0
$x_2 = 1$	9/16	\	6/16	8/16	0
$x_2 = 2$	8/16	6/16	\	4/16	0
$x_2 = 3$	6/16	8/16	4/16	\	0
$x_2 = 4$	0	0	0	0	\

The expected total number of effective equations from this part is:

$$14 + 2 * 9/16 + 1 * 8/16 + 3 * 4/16 + 8 * 0 \approx 16.5$$

- Rows containing 3 linearized bits.

This class becomes much more complicated. Fortunately, we only need to focus on the situation of ($\Pi^4_{0,0,z}\, \Pi^4_{3,0,z}\, \Pi^4_{4,0,z}$). Notice that $\Pi^4_{4,0,z}$ can never match a recovered bit since those bits from the fifth lane are always uncertain. Thus, the conclusion is no matter how $\Pi^4_{0,0,z}$ and $\Pi^4_{3,0,z}$ match those recovered bits, we can always set 2 effective equations on $\Pi^4_{0,0,z}$, $\Pi^4_{3,0,z}$, $\Pi^4_{0,0,z} \oplus \Pi^4_{4,0,z}$ or $\Pi^4_{3,0,z} \oplus \Pi^4_{4,0,z}$. The total number of effective equations from this part is $2 * 2 = 4$.

Finally, the expected gain of all 56 linearized sites under a random 256-bit digest is:

$$2^{14} * 2^{16.5} * 2^4 = 2^{34.5}$$

It is emphasized again that this result can hold for all 64 groups of 56 linearized sites starting from $\Pi^3_{4,*,z_s}$, while cannot hold for $x_s \neq 4$. In this paper we would not provide any analysis on $x_s \neq 4$ (including Keccak-224) because $x_s = 4$ is the only selection leading to the best results. From the analysis above, we can see that multiple bits located at the same row are much more efficient than several solitary bits. And $x_s = 4$ is exactly the best selection with fewest (22 out of 56) solitary bits.

Although attackers can find 34.5 effective equations, a problem is that the degree of remaining freedom is only 32 (162 out of 194). We will solve this problem in Section 4.3 by using the technology of freedom return. In addition, although the expected gain of a single group is $2^{34.5}$, attackers are actually able to go through all 64 groups to search one with the greatest gain. We will discuss this problem in Section 4.4.

4.3 Freedom Return

In Section 3, we have explained that our strategy is partially linearizing 56 sites in $\Pi^4_{*,0,*}$ by setting 173 constants in Π^3 (spending 162 degrees of freedom). Those linearized sites may work or may not work in different cases. In the previous section, we further analyzed that those 56 sites can compose 34.5 effective equations in average, remaining about one quarter unused. A natural idea is, resetting those constants to linearize used sites only, so that several degrees of freedom can be saved. In other words, unused sites are able to return some degrees of freedom. In this section, we give a discussion on this problem.

Notice that those 173 constants consist of 29 columns and 28 solitary bits. We have mentioned in Section 3 that each column is related to 4 linearized sites. These degrees of freedom can be returned only if 4 sites are simultaneously unused—we regard it as hardly possible since the ratio of unused sites is only 1/4. However, for those solitary bits, each of them is related to only 1 linearized site. If the corresponding 28 linearized sites are partly unused, degrees of freedom can be returned by removing the solitary constant bits.

Take $\Pi^4_{1,0,18}$ (in **Table 6**, $z = 18$) as an example. Suppose the output row is 1001? $\leftarrow$ 1?0??. Then the linearized site remains unused and its corresponding constant bit $\Pi^3_{3,1,38}$ (in **Table 4**, step = 27) can be removed, which will not influence the linearization of all other 55 sites. Take $\Pi^4_{0,0,46}$ (in **Table 6**, $z = 46$) as another example. If this site remains unused, its corresponding constant bit $\Pi^3_{2,0,46}$ (in **Table 4**, step = 19) cannot be removed, because it has been bound with $\Pi^3_{4,1,0}$ (in **Table 5**, line 4). Actually those 22 bits with notation "*" in **Table 6** are exactly all the sites that can return degrees of freedom. The situations of freedom return (or those bits being ineffective) are summarized in **Table 8** with corresponding probabilities. It is concluded that those 22 bits can return 7 degrees of freedom in average, which is enough to make up the lack of freedom.

Table 8: Situations of freedom return (example with the starting column $\Pi^3_{4,*,0}$).

Bit	Return Situation	Prob.	Bit	Return Situation	Prob.
$\Pi^{4*}_{3,0,4}$	Never	0	$\Pi^{4*}_{1,0,18}$	$\Pi^{4*}_{1,0,18}$ doesn't match	0.25
$\Pi^{4*}_{1,0,23}$	$\Pi^{4*}_{1,0,23}$ doesn't match	0.25	$\Pi^{4*}_{2,0,24}$	$\Pi^{4*}_{2,0,24}$ doesn't match	0.5
$\Pi^{4*}_{1,0,28}$	$\Pi^{4*}_{1,0,28}$ doesn't match	0.25	$\Pi^{4*}_{2,0,29}$	$\Pi^{4*}_{2,0,29}$ doesn't match	0.5
$\Pi^{4*}_{1,0,33}$	$\Pi^{4*}_{1,0,33}$ doesn't match	0.25	$\Pi^{4*}_{2,0,34}$	$\Pi^{4*}_{2,0,34}$ doesn't match	0.5
$\Pi^{4*}_{2,0,39}$	$\Pi^{4*}_{2,0,39}$ doesn't match	0.5	$\Pi^{4*}_{0,0,51}$	$\Pi^{4*}_{0,0,51}$ doesn't match	0.25
$\Pi^{4*}_{0,0,56}$	$\Pi^{4*}_{0,0,56}$ doesn't match	0.25	$\Pi^{4*}_{3,0,58}$	Never	0
$\Pi^{4*}_{0,0,61}$	$\Pi^{4*}_{0,0,61}$ doesn't match	0.25	$\Pi^{4*}_{3,0,63}$	Never	0

$\Pi^{4*}_{4,0,4}$	$\Pi^{4*}_{3,0,4}$ matches $\Pi^{4*}_{4,0,4}$ doesn't match	0.5	$\Pi^{4*}_{4,0,9}$	$\Pi^{4}_{3,0,9}$ matches $\Pi^{4*}_{4,0,9}$ doesn't match	0.5
$\Pi^{4*}_{4,0,14}$	$\Pi^{4}_{3,0,14}$ matches $\Pi^{4*}_{4,0,14}$ doesn't match	0.5	$\Pi^{4*}_{2,0,19}$	$\Pi^{4*}_{2,0,19}$ doesn't match $\Pi^{4}_{3,0,19}$ matches	0.25
$\Pi^{4*}_{1,0,38}$	$\Pi^{4}_{0,0,38}$ matches $\Pi^{4*}_{1,0,38}$ doesn't match	3/16	$\Pi^{4*}_{4,0,53}$	$\Pi^{4}_{0,0,53}$ matches $\Pi^{4*}_{4,0,53}$ doesn't match	0.75
$\Pi^{4*}_{4,0,58}$	$\Pi^{4}_{0,0,58}$ matches $\Pi^{4*}_{3,0,58}$ matches $\Pi^{4*}_{4,0,58}$ doesn't match	6/16	$\Pi^{4*}_{4,0,63}$	$\Pi^{4}_{0,0,63}$ matches $\Pi^{4*}_{3,0,63}$ matches $\Pi^{4*}_{4,0,63}$ doesn't match	6/16

4.4 Best Result among 64 Groups

In Keccak-256, the 256-bit output (or digest) consisting of 64 rows shows a characteristic of rotational symmetry. This symmetry ensures that the expected gains of all 64 groups of linearized sites are exactly the same ($2^{34.5}$). However, for a certain output, attackers are able to go through all 64 groups to search one with the greatest gain. For example, a 256-bit output contains 4 rows of 1000? ←????? in average. Then those groups affected by 4 bad rows are apparently less efficient than those groups that rotate 4 bad rows to 26 nonlinearized rows. On the other hand, since there are 64 groups in total, attackers can fix 6 linearized sites and hope that there exists 1 group where 6 linearized sites can all match those recovered bits, leading to a gain increase of about 2^3. Overall, the best result among 64 groups can be much higher than the expected gain of a single group.

The best result among 64 groups can hardly be analyzed theoretically. Therefore, we run a program to provide a practical analysis instead. It is concluded that under 1000000 tests of random digests, 74.5% of them can lead to a gain of 2^{38} (with enough freedom). Notice that to search the best group with the greatest gain, attackers only need to count the number of recovered bits among all linearized sites. Thus the cost of 2^6 searching can be neglected compared with even 1 round of equations solving and Keccak running, and the final complexity is still 2^{218}.

5 Improved Preimage Attack on 4-Round Keccak-224

Preimage cryptanalysis on 4-Round Keccak-224 is provided in this section, which is quite similar to Section 4. The main difference is that the Keccak-224 output contains 32 rows with only 3 certain bits. After discussing the influence of this difference, we will present the expected gain analysis of a specific group of linearized sites. It is finally concluded that attackers can obtain a complexity decrease of 2^{32} with probability of 61.3% under a random 224-bit digest.

5.1 Linearized Sites Matching 3-Bit Output Rows

In Keccak-256, the total gain is corresponding to the number of effective equations, because each equation can exclude half of possible cases and conserve both 2 matched inputs. However, in Keccak-224, things become different for those 3-bit output rows: the number of matched inputs becomes 4 and an effective equation may exclude 1 of them. In this situation, the probability of successfully matching is 3/16 rather than 2^{-3}, with a gain of only 3/2. For example, suppose the output row is 100?? with matched inputs 11010, 00101, 10000 and 11011. Then the equations on $\Pi^4_{0,0,z}$ and $\Pi^4_{2,0,z}$ will both exclude 1 of them. Although equations like $\Pi^4_{0,0,z} \oplus \Pi^4_{2,0,z} = 1$ can conserve all matched inputs with an entire gain of 2^1, it is still less efficient than 2 equations of $\Pi^4_{0,0,z} = 1$ and $\Pi^4_{2,0,z} = 0$

(with a gain of 3). Actually the expected gain of rows containing 2 linearized bits is very complex. But for solitary bits, the expected gain can be easily summarized by **Table 9**.

Table 9: Cases of solitary bits matching 3-bit output rows.

Gain	$\Pi^4_{0,0,z}$	$\Pi^4_{1,0,z}$	$\Pi^4_{2,0,z}$	$\Pi^4_{3,0,z}$	$\Pi^4_{4,0,z}$
2^1	4 out of 8	4 out of 8	0	0	0
$2^{0.5*}$	4 out of 8	0	8 out of 8	0	0
2^0	0	4 out of 8	0	8 out of 8	8 out of 8

*It denotes the gain of $3/2$, since $3/2 * 3/2 \approx 2^1$.

It's seen that there are still a half of linearized sites which can match those recovered bits. However, the efficiency of those linearized sites in the 3-bit part decreases because 1 effective equation may bring a gain of $2^{0.5}$, still costing 1 degree of freedom. Moreover, $\Pi^4_{3,0,z}$ and $\Pi^4_{4,0,z}$ are severely useless in the 3-bit part, even with another linearized bit located at the same row. Therefore, to reach the best result, attackers should choose a specific group so that:

a. Most linearized sites work in the 4-bit part instead of the 3-bit part.

b. Few of $\Pi^4_{3,0,z}$ and $\Pi^4_{4,0,z}$ are located at the 3-bit part.

It is found that $z_s = 6$ can exactly meet the conditions above. The details are shown in **Table 10**.

Table 10: The specific group of 56 linearized sites (with the starting column $\Pi^3_{4,*,6}$).

z	$\Pi^4_{x,0,z}$	z	$\Pi^4_{x,0,z}$	z	$\Pi^4_{x,0,z}$	z	$\Pi^4_{x,0,z}$
0	$\Pi^4_{0,0,z}\,\Pi^{4*}_{3,0,z}\,\Pi^{4*}_{4,0,z}$	16	\	32	$\Pi^4_{2,0,z}$	48	\
1	\	17	$\Pi^4_{3,0,z}\,\Pi^4_{4,0,z}$	33	\	49	$\Pi^4_{0,0,z}\,\Pi^4_{1,0,z}$
2	$\Pi^4_{3,0,z}\,\Pi^4_{4,0,z}$	18	\	34	$\Pi^{4*}_{1,0,z}$	50	\
3	$\Pi^{4*}_{0,0,z}$	19	\	35	$\Pi^{4*}_{2,0,z}$	51	\
4	\	20	$\Pi^4_{3,0,z}\,\Pi^{4*}_{4,0,z}$	36	$\Pi^4_{1,0,z}$	52	$\Pi^4_{0,0,z}$
5	$\Pi^4_{0,0,z}\,\Pi^{4*}_{3,0,z}\,\Pi^{4*}_{4,0,z}$	21	\	37	$\Pi^4_{2,0,z}$	53	\
6	\	22	$\Pi^4_{2,0,z}\,\Pi^4_{3,0,z}$	38	\	54	$\Pi^4_{0,0,z}$
7	$\Pi^4_{3,0,z}\,\Pi^4_{4,0,z}$	23	\	39	$\Pi^{4*}_{1,0,z}$	55	\
8	\	24	$\Pi^{4*}_{1,0,z}$	40	$\Pi^{4*}_{2,0,z}$	56	\
9	\	25	$\Pi^{4*}_{2,0,z}\,\Pi^4_{3,0,z}$	41	$\Pi^4_{1,0,z}$	57	$\Pi^{4*}_{0,0,z}$
10	$\Pi^{4*}_{3,0,z}\,\Pi^{4*}_{4,0,z}$	26	$\Pi^4_{1,0,z}$	42	$\Pi^4_{2,0,z}$	58	\
11	\	27	$\Pi^4_{2,0,z}\,\Pi^4_{3,0,z}$	43	\	59	$\Pi^4_{0,0,z}\,\Pi^{4*}_{4,0,z}$
12	$\Pi^4_{3,0,z}\,\Pi^4_{4,0,z}$	28	\	44	$\Pi^4_{0,0,z}\,\Pi^{4*}_{1,0,z}$	60	\
13	\	29	$\Pi^{4*}_{1,0,z}$	45	$\Pi^{4*}_{2,0,z}$	61	$\Pi^4_{4,0,z}$
14	\	30	$\Pi^{4*}_{2,0,z}$	46	$\Pi^4_{1,0,z}$	62	$\Pi^{4*}_{0,0,z}$
15	$\Pi^4_{3,0,z}\,\Pi^{4*}_{4,0,z}$	31	$\Pi^4_{1,0,z}$	47	$\Pi^4_{0,0,z}\,\Pi^4_{2,0,z}$	63	\

*Those bits are relevant to the solitary part in **Table 4**, and meanwhile irrelevant to the solitary part appearing in **Table 5**.

Notice that the 224-bit output (or digest) does not fulfill the characteristic of rotational symmetry. Thus the expected gains are quite different among 64 groups of linearized sites. Although attackers are still able to go through them all, the best gain will not be larger than of this specific group since most groups cannot meet the strict conditions. Therefore,

in the next section we only need to analyze the expected gain of this specific group, and the result is just the final complexity of preimage attack on 4-round Keccak-224.

5.2 Gain Analysis of 56 Linearized Sites

Since the specific group has rotated most $\Pi^4_{3,0,z}$ and $\Pi^4_{4,0,z}$ to the 4-bit part, the efficiency of those linearized sites in the 3-bit part can greatly increase (about five sixth effective). In other words, the degree of remaining freedom is possibly insufficient for all effective equations. To reach the best result, attackers should set linear equations in order of:

1. The 4-bit part $z = 0 \sim 31$ where an effective equation always brings a gain of 2^1.
2. $\Pi^4_{0,0,z}$ and $\Pi^4_{1,0,z}$ in the 3-bit part which may bring a gain of 2^1.
3. Rows containing 2 linearized bits in the 3-bit part (more efficient).
4. Spend remaining freedom on those solitary bits with a gain of only $2^{0.5}$.

In this section, we also provide the theoretical analysis under this order.

- The 4-bit part $z = 0 \sim 31$ where an effective equation always brings a gain of 2^1.

The principle is the same as Section 4.2 and Section 4.3. Here we just conclude that this part can lead to 19 effective equations and return 3.75 degrees of freedom in average.

- $\Pi^4_{0,0,z}$ and $\Pi^4_{1,0,z}$ in the 3-bit part which may bring a gain of 2^1.

The 3-bit part contains 9 solitary $\Pi^4_{0,0,z}$ and $\Pi^4_{1,0,z}$. Including $\Pi^4_{0,0,59}$ (since $\Pi^4_{4,0,59}$ is almost useless), these 10 bits can lead to 5 effective equations with a gain of 2^1 and return 2 degrees of freedom in average.

Here we return the freedom of those unused bits in advance (5 degrees in total) for convenience. In the next two steps, 1 linearized bit with "*" will cost 2 degrees of freedom.

- Rows containing 2 linearized bits in the 3-bit part (more efficient).

From this step onwards we have to deal with those equations with a gain of only $2^{0.5}$. Up to this step, attackers have set 24 equations with $194 - 162 - 24 + 4 + 2 + 5 = 19$ degrees of remaining freedom in average.

As for those rows containing 2 linearized bits, it is worth mentioning that separately setting 2 equations will never be less efficient than 1 equation. For example, to match the output row 100??, equation $\Pi^4_{0,0,z} = 1$ will exclude half of possible cases and 1 matched input (with a gain of 3/2), while equations $\Pi^4_{0,0,z} = 1$ and $\Pi^4_{4,0,z} = 0$ will exclude three quarters of possible cases and 2 matched inputs (with a gain of 2). **Table 11** summarizes the gains of those most efficient equations under all 8 possible cases.

Table 11: Cases of 2 linearized bits matching 3-bit output rows.

Gain	$\Pi^4_{0,0,z}\,\Pi^4_{1,0,z}$	$\Pi^4_{0,0,z}\,\Pi^4_{2,0,z}$	$\Pi^4_{0,0,z}\,\Pi^4_{4,0,z}$
2^2	2 out of 8	0	0
$2^{1.5}$	2 out of 8	8 out of 8	0
2^1	4* out of 8	0	8 out of 8

*Note: 2 out of 4 only need 1 equation on $\Pi^4_{0,0,z}$.

The table above also provides an analysis for $\Pi^4_{0,0,59}$ and $\Pi^4_{4,0,59}$. Notice that the gain of these 2 linearized sites is at most 2^1, costing 3 degrees of freedom ($\Pi^4_{4,0,59}$ can return freedom). Yet the gain of single $\Pi^4_{0,0,59}$ has reached $2^{0.75}$. Thus it will be more efficient if attackers neglect $\Pi^4_{4,0,59}$ and spend the freedom on another equation with a gain of $2^{0.5}$.

Excluding the entry ($\Pi^4_{0,0,z}\,\Pi^4_{4,0,z}$) of $z = 59$, this part can bring a gain of $2^{4.25}$ and cost 6.25 degrees of freedom in average.

- Spend remaining freedom on those solitary bits with a gain of only $2^{0.5}$.

The remaining freedom is about 12 degrees till this step. Attackers first focus on those $\Pi^4_{2,0,z}$ and unused $\Pi^4_{0,0,z}$ without "*" (costing 1 degree of freedom each). Then attackers spend all remaining freedom on those bits with "*" (costing 2 degrees of freedom each).

There are 3 terms $\Pi^4_{2,0,z}$ and 3 terms $\Pi^4_{0,0,z}$ without "*" in the 3-bit part. Notice that $\Pi^4_{0,0,z}$ may have been used in the second step (1.5 terms in average). Thus attackers are expected to set 4.5 equations without "*" and 3.75 equations with "*" within 12 degrees of remaining freedom. In summary, this part can lead to about 8 effective equations and the expected gain is 2^4.

Finally, the expected gain of this specific group of 56 linearized sites is:

$$2^{19} * 2^5 * 2^{4.25} * 2^4 \approx 2^{32}$$

To support the theoretical gain analysis, we also run a program to provide a practical analysis. It is concluded that under 1000000 tests of random digests, 61.3% of them can lead to a gain of 2^{32}.

6 Experiments on 64-Bit Partial Preimage Attack

We present the result of partial preimage attack in this section. The 64-bit target digest is corresponding to $z = 0 \sim 15$ of those lanes, or to say, 16 4-bit rows. If we adopt previous freedom reuse strategy, at most 17 linearized sites can be located at corresponding rows, still hard to reach a practical complexity. Therefore, we adopt a modified strategy as shown in **Figure 10**, which can linearize 31 sites in $z = 0 \sim 15$ by setting 160 constants.

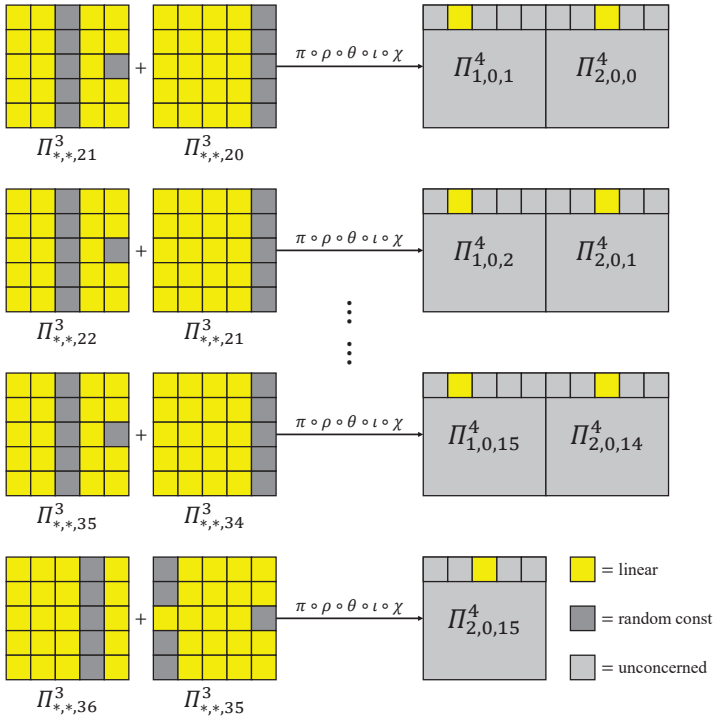

Figure 10: The modified freedom reuse strategy.

The basic idea is to linearize 2 sites by setting 11 constants in each procedure. Excluding 14 duplicate bits, $15 * 2 = 30$ sites can be linearized by $15 * 11 - 14 = 151$ constants. With

another 10 constants (1 duplicate) linearizing $\Pi^4_{2,0,15}$, it costs 160 degrees of freedom (158 combined with zero coefficient) in total. These 31 linearized sites can lead to $15 \sim 31$ effective equations (21 in average). And thus attackers can construct a partial preimage matching 64-bit all '1' digest with a complexity of only 2^{33}.

One instance is presented in **Table 12**.

Table 12: One instance matching 64-bit digest (in big-endian order).

Starting State[a] (the second[b] message block)
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff ffffffffffffffff
Partial Preimage (the second[b] message block)
6b43dfc40739a467 c383cffe561c722d 61b08ad49180b726 9055e2f81e25111f 995ca723a4ad4e52
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
6b43dfc40739a467 c383cffe561c722d 61b08ad49180b726 9055e2f81e25111f 995ca723a4ad4e52
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000
4-Round Digest
ffffc207c85f3d18 ffff7f390a78ac44 ffffdb5599aafd65 fffff81322092859

[a]Note: Although it is a special case with all '0' and all '1' words, in general cases, the complexity is still 2^{33} as long as the starting state meets those 195 conditions.
[b]Note: To construct the first message block linking all '0' IV to the starting state, the complexity is 2^{195}, which is still impractical.

7 Conclusion

In this paper, we provide an improved preimage cryptanalysis on 4-round Keccak-224/256 based on a 2-round linear structure proposed by Li and Sun. Our main idea is partially linearizing as many sites as possible within the 194 degrees of remaining freedom. For this target, we improve the preimage attack algorithm in three aspects:

1. We adopt a freedom reuse strategy where multiple sites can be linearized by some common constants.

2. We propose a zero coefficient technology so that several degrees of freedom can be saved from those constant bits.

3. For those unused sites, we return the corresponding freedom which would not affect the linearization of other used sites.

With these improvements, 56 sites can be linearized within 162 degrees of freedom and return 7 degrees of freedom in average. Then we spend the remaining freedom on setting effective equations which can decrease the complexity of searching a preimage. It is finally analyzed, theoretically and practically, that the expected complexity of preimage attack on 4-round Keccak-224/256 can be decreased to $2^{192}/2^{218}$. So far, the results are both the best preimage cryptanalysis on 4-round Keccak.

It is noted that our attack algorithm is still far from threatening the security of even 5-round Keccak: making use of 2-round linear structure, 1-round partial linearization and 1-round probability analysis, the linear route can hardly pass through the fifth nonlinear S-box layer. However, it should be emphasized that our idea can actually be applied with any linear structure. If a better linear structure is found with more degrees of freedom left, the complexity of preimage attack can further decrease.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant Nos. 2018YFB0803405 and 2017YFA0303903).

References

[AM09] Jean-Philippe Aumasson and Willi Meier. 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.

[BCC11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of Keccak and Luffa. In *FSE 2011*, volume 6733 of *LNCS*, pages 252–269. Springer, Heidelberg, 2011.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, 2008.

[BDPV11a] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions. Submission to NIST (Round 3), 2011. http://sponge.noekeon.org/CSF-0.1.pdf.

[BDPV11b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, version 3.0. Submission to NIST (Round 3), 2011. http://keccak.noekeon.org/Keccak-reference-3.0.pdf.

[Ber10] Daniel J. Bernstein. Second preimages for 6 (7? (8??)) rounds of Keccak? NIST Mailing List, 2010. http://ehash.iaik.tugraz.at/uploads/6/65/NIST-mailing-list_Bernstein-Daemen.txt.

[DDS12] Itai Dinur, Orr Dunkelman, and Adi Shamir. New attacks on Keccak-224 and Keccak-256. In *FSE 2012*, volume 7549 of *LNCS*, pages 442–461. Springer, Heidelberg, 2012.

[DGPW12] Alexandre Duc, Jian Guo, Thomas Peyrin, and Lei Wei. Unaligned rebound attack: Application to Keccak. In *FSE 2012*, volume 7549 of *LNCS*, pages 402–421. Springer, Heidelberg, 2012.

[DL11] Ming Duan and Xuajia Lai. Improved zero-sum distinguisher for full round Keccak-f permutation. Cryptology ePrint Archive, Report 2011/023, 2011. http://eprint.iacr.org/2011/023.

[DMP+15] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 733–761. Springer, Heidelberg, 2015.

[DV12] Joan Daemen and Gilles Van Assche. Differential propagation analysis of Keccak. In *FSE 2012*, volume 7549 of *LNCS*, pages 422–441. Springer, Heidelberg, 2012.

[GLL+20] Jian Guo, Guohong Liao, Guozhen Liu, Meicheng Liu, Kexin Qiao, and Ling Song. Practical collision attacks against round-reduced SHA-3. *Journal of Cryptology*, 33(1):228–270, 2020.

[GLS16] Jian Guo, Meicheng Liu, and Ling Song. Linear structures: Applications to cryptanalysis of round-reduced Keccak. In *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 249–274. Springer, Heidelberg, 2016.

[KMNS13] Stefan Kölbl, Florian Mendel, Tomislav Nad, and Martin Schläffer. Differential cryptanalysis of Keccak variants. In *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 141–157. Springer, Heidelberg, 2013.

[LS19] Ting Li and Yao Sun. Preimage attacks on round-reduced Keccak-224/256 via an allocating approach. In *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 556–584. Springer, Heidelberg, 2019.

[MPS13] Pawel Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced Keccak. In *FSE 2013*, volume 8424 of *LNCS*, pages 241–262. Springer, Heidelberg, 2013.

[MS13] Pawel Morawiecki and Marian Srebrny. A SAT-based preimage analysis of reduced KECCAK hash functions. *Information Processing Letters*, 113(10-11):392–397, 2013.

[oST15] 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.

[QSLG17] Kexin Qiao, Ling Song, Meicheng Liu, and Jian Guo. New collision attacks on round-reduced Keccak. In *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 216–243. Springer, Heidelberg, 2017.

[SLG17] Ling Song, Guohong Liao, and Jian Guo. Non-full sbox linearization: Applications to collision attacks on round-reduced Keccak. In *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 428–451. Springer, Heidelberg, 2017.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 19–35. Springer, Heidelberg, 2005.