Allocating Rotational Cryptanalysis based Preimage Attack on 4-round Keccak-224 for Quantum Setting

Runsong Wang[1], Xuelian Li[1†], Juntao Gao[2*†], Hui Li[1†] and Baocang Wang[2†]

[1]School of Mathematics and Statistics, Xidian University, Xi'an, 710000, Shaanxi, China.

[2*]School of Telecommunication and Engineering, Xidian University, Xi'an, 710000, Shaanxi, China.

*Corresponding author(s). E-mail(s): jtgao@mail.xidian.edu.cn;
Contributing authors: wangrs12@stu.xidian.edu.cn;
xuelian202@163.com; hui0921@stu.xidian.edu.cn;
bcwang@xidian.edu.cn;
†These authors contributed equally to this work.

Abstract

In this paper, we aim to present a quantum setting oriented preimage attack against 4-round Keccak-224. An important technique we called the allocating rotational cryptanalysis takes the preimage attack into the situation of 2-block preimage recovery. With the conditions on the middle state proposed by Li et al., we use the generic quantum preimage attack to deal with the finding of first preimage block. By using the newly explored propagation of rotational relations, we significantly increase the number of eigenpoints at the end of 4-round modified Keccak-f from 0 to 32, and therefore improving the accuracy of determining the rotational number for a certain rotational counterpart in the quantum setting by more than 10 orders of magnitude. On the basis of the above, we design an efficient unitary oracle operator with only twice calling of the 4-round modified Keccak-f, which costs half of previous results, to mark a rotational counterpart of the second preimage block in order that the second preimage block can be found indirectly from a quickly generated specified search space. As a result on the 4-round

KECCAK-224: In the classical setting, the preimage attack with the complexity decreased to $\mathbf{2^{218}}$ is better than the result based on the pioneered rotational cryptanalysis. In the quantum setting, the amplitude amplification driven preimage attack with a complexity of $\mathbf{2^{110}}$ is by far the best dedicated quantum preimage attack. Additionally, the SKW algorithm is applied to the dedicated quantum preimage attack against the 4-round KECCAK-224 for the first time, which is exponentially easier to implement in quantum circuit than the former, with a complexity of $\mathbf{2^{111}}$.

Keywords: KECCAK, Rotational Cryptanalysis, Linearization of Keccak-$\boldsymbol{f}$, Preimage Attack, Amplitude Amplification, SKW Algorithm

1 Introduction

In view of a series of breakthrough attack methods [1–4] proposed by wang et al. for MD5, SHA-1 and other hash function standards, the National Institute of standards and Technology (NIST) announced the public collection of the next generation encryption hash standard SHA-3 in 2007. The KECCAK sponge function [5] became the candidate scheme of SHA-3 competition in 2008 [6], and was finally selected as the final standard of SHA-3 in 2012. In 2015, KECCAK was officially recognized as a federal information processing standard (FIPS) by NIST as SHA-3 [7]. Since KECCAK was published in 2008, the properties of its hash function have been widely studied. Some progress has been made in cryptanalysis, including KECCAK's keyed mode and keyless mode. Here, we mainly review the research progress of preimage attack against the round-reduced KECCAK.

In the field of the classical cryptanalysis, many good conclusions have emerged from the discussion on the preimage recovery of round reduced KECCAK. In the [8], Bernstein et al. proposed a preimage attack for the round-reduced KECCAK, which requires higher computation and memory consumption compared with the parallel exhaustive search, to reach up to 8-round. For the frst time, Morawiecki et al. applied the rotational cryptanalysis to the preimage recovery against 3-round / 4-round KECCAK in [9]. By tracking the propagation of the bits relations between the state array and its rotational counterparts, they provided an indirect idea to find the preimage. After that, many cryptologists have turned to the algebraic attacks to solve the preimage for a given hash value. Guo et al. proposed a new linear structure for KECCAK in [10]. They introduced the linearization for the permutation of 3-round KECCAK, and demonstrated the 3-round / 4-round preimage attack strategy with the linear structure. In [11], Li et al. proposed a technique called cross-linear structures, which helps them to obtain several linear equations by guessing the value of a linear polynomial, to improved the works [10]. Later in [12], Li et al. demonstrated a new preimage attack technology against 3-round / 4-round KECCAK called the allocating approach, which finds a 2-block preimage

instead of a 1-block one. By establishing a set of weaker constraints on the middle state compared into those brought by the initial values and the hash values, they succeeded in reducing the complexity of finding the preimage in two separate stages. Besides taking the approach of solving the linear system, in recent years, there have been many efforts to save more degrees of freedom by using the method of solving the nonlinear system to carry out the preimage attack. In [13], Liu et al. made full use of the equations derived from hash value by constructing Boolean quadratic system. They use the relinearization technique to solve the quadratic system, and thus improving the preimage attack on Keccak-384/512. Later in [14], Dinur et al. gave the specific complexity of polynomial method for solving multivariate equation systems [15], and applied this method to the preimage attack on round-reduced Keccak. The latest relevant research result is [16], Wei et al. revisited the Crossbred algorithm for Solving Boolean multivariate quadratic (MQ) systems, and shown that in the case of $D = 2$ the Crossbred algorithm is superior to brute force theoretically and practically under the feasible memory costs. By solving the new constructed Boolean MQ system, Wei et al. successfully reached the preimage attacks on Keccak-224/256 up to 4 rounds, and obtained the best attack complexity so far.

The development of quantum cryptanalysis is accompanied by the iterative updating of quantum computing hardware and software. It is more urgent than ever to evaluate the security of traditional cryptographic primitives in quantum setting, and this has become the consensus of the cryptographic community. In recent years, the efforts related to the quantum cryptanalysis are mainly classified into two models[17–19]: $Q1$ and $Q2$ models. Under the $Q2$ model, the attacker can not only use the quantum computer to gain the advantage of quantum computing, but also use it to perform online superposition queries to the target primitives. Among the many $Q2$ model studies, the works [20] with milestone significance is proposed by Leander et al., which used the Simon's algorithm to build the unitary oracle operator for the Grover's algorithm to greatly reduce the effective key-length of FX-construction. However, the motivation of the designers of cryptographic primitives to disclose interfaces supporting online superposition queries to an attacker is questionable. Therefore, more recently, cryptologists began to carry out cryptanalysis[21–24] under the more reasonable $Q1$ model, where is assumed that the attacker is equipped with a quantum computer that can only provide offline computing and can not access the target primitives in superposition. Unfortunately, one of the most actively researched subjects in the classical cryptanalysis — SHA-3 hash standards are rarely discussed in the quantum setting. In order to analyze the ability of SHA-3 to resist the preimage recovery in the quantum setting, Amy et al. in [25] evaluated the attack cost of the generic quantum preimage attack which is assumed to run on a surface code based fault-tolerant quantum computer. The latest research about dedicated quantum preimage attack against the round-reduced Keccak is shown in [26], Wang et al. constructed a

feasible method to bring the rotational cryptanalysis technology into the quantum setting, and achieved better results than the generic quantum preimage attack.

Our contributions. To obtain a quantum setting oriented dedicated preimage attack against 4-round KECCAK-224, this paper mainly makes contributions in following three aspects.

- One critical technique we used in the preimage attack is the allocating rotational cryptanalysis which introduces the idea of two-block attack on standard KECCAK-224 into the rotational cryptanalysis and makes a great improvements on the propagation of the rotational relations. Specifically, we mainly focus on a state array that satisfies two sets of constraints which were once used by Li et al. in [12] to linearize KECCAK-f. By exploring how the 4-round KECCAK-f changes the rotational relations between this kind of state array and the corresponding rotational counterparts, we gain a whole new way of the propagation of rotational relations which has up to 32 eigenpoints at the end of the 4-round KECCAK and takes the results of Morawiecki et al. in [9] a big step forward.
- In order that the newly found propagation of rotational relations and 32 eigenpoints can be used to mount a improved preimage attack, we develope a good method to quickly expand this kind of state array into a large enough specified search space for finding the second preimage block indirectly. More specifically, we theoretically relate the second set of constraints to the state array that serves as the input to the second hash process. It is worth noting that the second set of constraints is used to limit the output of 1-round KECCAK-f. Therefore, our process of building the specified search space does not involve the running of KECCAK-f, reducing a significant amount of computational overhead.
- Taking above work to the quantum setting and obtaining a good performance dedicated quantum preimage attack on the 4-round KECCAK-224 is dependent on the design of an efficient unitary oracle operator which can mark the rotational counterpart of the second preimage block from other random state arrays with a high accuracy. By using the newly obtained 32 eigenpoints, we first designed a quantum search algorithm with high accuracy for finding the rotational number of the rotational counterpart. Through embedding this quantum algorithm into the design of the unitary oracle operator, we successfully reduced the computational complexity of the unitary oracle operator to only twice calling of the 4-round modified KECCAK-f while improving its accuracy. Our efficient unitary oracle operator working with different quantum search algorithm will have different advantageous tendencies.

The best known preimage attacks on the 4-round KECCAK-224 have been summarized in the following Table 1. In the classical setting, the classical counterpart of our preimage attack decreases the complexity to 2^{218}, which is better than the result based on the pioneered rotational cryptanalysis [9],

but still not up to the excellent work of wang et al. in [16]. In the quantum setting, by making the amplitude amplification work with the unitary oracle operator, we gain the best dedicated quantum preimage attack on the 4-round Keccak-224 with a complexity of 2^{110} by far. Compared to results of wang et al. in the [26] and the generic quantum preimage attack, our result has roughly 6 times and 8 times better attack effect, respectively. More importantly, we succeeded for the first time in bringing the SKW quantum algorithm, which is exponentially easy to build on the quantum circuits, into the preimage attack against the 4-round Keccak-224 with a complexity of 2^{111}. Compared to the generic quantum preimage attack, our dedicated preimage attack based on the SKW algorithm has roughly 4 times better attack effect. Note that due to the low accuracy of the unitary oracle operator designed by Wang et al. in [26] when marking the rotational counterpart of the preimage, the SKW algorithm cannot be applied to their attack against 4-round Keccak-224.

Table 1: The best-known preimage attacks on the 4-round Keccak-224.

Setting	Rounds	Variants	Time	RAM	qRAM	Reference
Classical	4	224	2^{221}	Negligible	0	[9]
	4	224	1^{st}block:2^{129} 2^{nd}block:2^{218}	Negligible	0	Sect.3.4
	4	224	2^{182}	Negligible	0	[16]
Quantum	4	224	2^{113}	0	Negligible	Grover Search
	4	224	$2^{112.57}$	0	Negligible	[26]
	4	224	1^{st}block:2^{98} 2^{nd}block:2^{110}	0	Negligible	Sect.4.2(The amplitude amplification)
	4	224	1^{st}block:2^{99} 2^{nd}block:2^{111}	0	Negligible	Sect.4.2(The SKW algorithm)

2 Preliminaries

Here, we give some backgrounds for a better presentation of the contents shown later. We first define some necessary notations to unify the meaning of symbols throughout our work. Then, reviewing the detail of the hash function — Keccak, the core permutation of the SHA-3, is indispensable. Next, we recall the core points of the rotational cryptanalysis and the linearization of Keccak-f proposed in [9] and [12], respectively. After summarizing the search tools in the quantum setting, we end this section by discussing the common technique — the uncompute trick and the controversial technique — qRAM.

2.1 Notations

r	The rate of a sponge function.
c	The capatity of a sponge function.
b	The width of a KECCAK-f permutation in bits.
n_r	The number of rounds for a KECCAK-f permutation.
Rnd	The round function of a KECCAK-f permutation.
$\theta, \rho, \pi, \chi, \iota$	The five step mappings of Rnd.
M^i	The message state of i^{th} block.
A^i	The messaged state of i^{th} block, which is the result of XORing the latest output state with the i^{th} message block M_i.
$\overleftarrow{A}^i$	The rotational counterpart of the i^{th} messaged state A_i.
H	The hash value.
Θ^i	The output state of the θ in the i^{th} round, and similarly have P^i, Π^i, X^i, and I^i.
$A^i_{x,y,z}$	The bit value at the position (x, y, z) of the messaged state A^i. This method will be applied to any state array.
Γ	The specified search space for searching the rotational counterparts of preimage A^2.

2.2 The Keccak hash function

Figure 1 shows the sponge construction adopted in the KECCAK hash. There

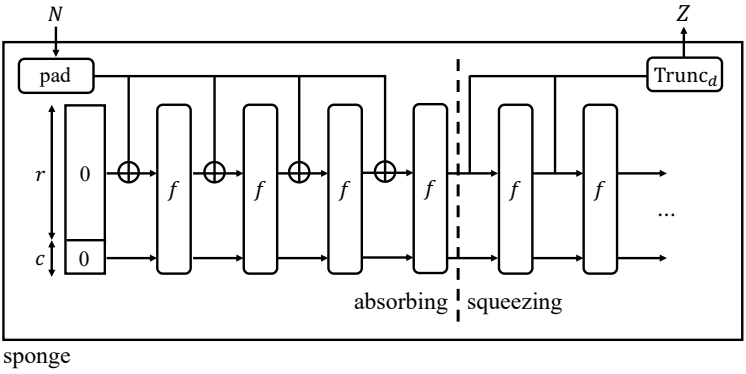


Fig. 1: Sponge construction.

are two processing phases(the absorbing phase and the sequeezing phase) in generating the hash values. In the absorbing phase, the padded r bits message block M^i will be XORed the latest hash block H^{i-1}(H^0 is the all '0' IV), and then inputed to the permutation KECCAK-f. The absorbing behavior will keep until all the message blocks M^i are processed. In the sequeezing phase, each time of executing the permutation KECCAK-f will return an r bits value as one part of the hash value H until H is fully constructed.

As the core of the KECCAK hash, above mentioned permutation KECCAK-f maps $b \in \{25, 50, 100, 200, 400, 800, 1600\}$ bits string to another b bits string. Here, we keep up with the parameter b adopted in SHA-3 standard, that is, $b = 1600$. The b bits inner state of KECCAK-f can be organized as a $5 \times 5 \times 64$ state array. Each bit is denoted as $A_{x,y,z}$, where $0 \leq x, y < 5$, and $0 \leq z < 64$. Figure 2 gives a clear view for the array construction, and several parts of the state array used in our work are also listed here.

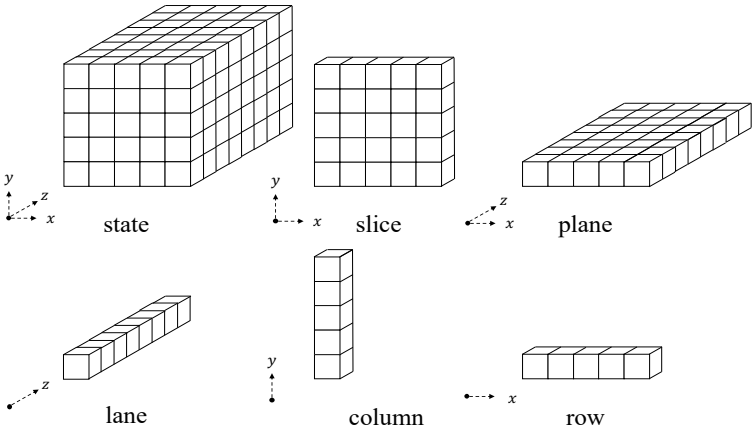


Fig. 2: State array and the parts of that.

The permutation KECCAK-f consists of 24 rounds of function Rnd which is consisted of 5 sub-step θ, ρ, π, χ and ι as follows:

$$
\begin{aligned}
&R = \iota \circ \chi \circ \pi \circ \rho \circ \theta, \\
&\theta : A_{x,y,z} = A_{x,y,z} \oplus \bigoplus_{j=0\sim4} A_{x-1,j,z} \bigoplus_{j=0\sim4} A_{x+1,j,z-1}, \\
&\rho : A_{x,y,z} = A_{x,y,z} \lll T_{x,z}, \\
&\pi : A_{y,2x+3y,z} = A_{x,y,z}, \\
&\chi : A_{x,y,z} = A_{x,y,z} \oplus (A_{x+1,y,z} \oplus 1) \cdot A_{x+2,y,z}, \\
&\iota : A_{0,0,z} = A_{0,0,z} \oplus RC_z^i,
\end{aligned}
$$

where $'\oplus'$ denotes the bitwise XOR operation, $' \lll'$ denotes rotating the bits $A_{x,y,z}$ forward by $T_{x,z}$(the offset) bits, $'\cdot'$ denotes the bitwise AND operation and RC_z^i denotes the z^{th} bit value of the round constant RC in the corresponding round.

2.3 The rotational cryptanalysis

Rotational cryptanalysis, improved by morawiecki et al., is an increasingly important cryptanalysis technique in recent years. In this kind of technique, a main concept is the rotational counterpart $\overleftarrow{A}$ which is obtained by rotating every bit $A_{x,y,z}$ of state array A forward by a certain number n called the rotational number along the lane $A_{x,y}$. The main idea of the rotational cryptanalysis is tracking the propagation of rotational relations(a set of probability values):

$$P_{x,y,z}[A_{x,y,z} \neq \overleftarrow{A}_{x,y,z+n}];\ \ 0 \leqslant x, y < 5,\ 0 \leqslant z < 64$$

between the state array A and the rotational counterpart $\overleftarrow{A}$ which are setted as the inputs of the round-reduced KECCAK simultaneously.

Morawiecki et al. have shown how the bitwise operations involved in KECCAK change the rotational relations in [9]. Let α, β be the input bits and *out* be the output bit, the bitwise AND operation changes the rotational relations according to $P_{out} = \frac{1}{2}(P_a + P_b - P_a P_b)$ and the bitwise XOR operation changes the rotational relations according to $P_{out} = P_a + P_b - 2P_a P_b$. The remaining bitwise opertors don't affect the rotational relations. To delay the rotational relations being independent($P_{x,y,z}[A_{x,y,z} \neq \overleftarrow{A}_{x,y,z+n}] \to 1/2$), Morawiecki et al. used rotational counterpart $\overleftarrow{A}$ as the input of the modified KECCAK that does not involve the step ι.

After applying same rounds KECCAK and modified KECCAK on the state array A and rotational counterpart $\overleftarrow{A}$ respectively, in the case of that there are several tracked rotational relations $P_{x,y,z}$ equal to 0 or 1, these coordinates (x, y, z) can be used as the eigenpoints to mount a preimage attack; in the situation of that a certain tracked rotational relations $P_{x,y,z}$ deviates slightly from 0.5, this coordinates (x, y, z) can be used as the key to make a distinguisher for the KECCAK-f permutation.

2.4 The linearization of Keccak-f

In the work of [12], Li et al. introduced a good method to linearize the permutation Keccak-f in the 2-block preimage attack against KECCAK-224/256. According to the inherent relationship between the hash block H^{i-1} and the next messaged block A^i, where the latest r bits in H^{i-1} and A^i are identical, Li et al. build a set of middle constraints for the 4-round KECCAK-224 as follows:

$$H'_{3,3,z} \oplus 1 = H'_{3,4,z}, H'_{4,3,z} \oplus 1 = H'_{4,4,z}, \bigoplus_{x,z} H'_{x,4,z} = 0, \tag{1}$$

where $0 \leqslant x < 5$ and $0 \leqslant z < 64$. Backward to the first preimage block A^1, the constraints 1 is much weaker than that of a 224-bit hash value, resulting in a lower complexity to search one of the proper preimage. Forward to the second preimage block A^2, the constraints 1 ensures the existence of preimage

A^2 featured by the following constraints:

$$A^2_{x,1,z} = A^2_{x,3,z} = A^2_{x,4,z} \oplus 1; \bigoplus_{x,z} A^2_{x,4,z} = 0, \tag{2}$$

where $0 \leq x < 5$ and $0 \leq z < 64$. As shown in the Figure 3, Li et al. use the

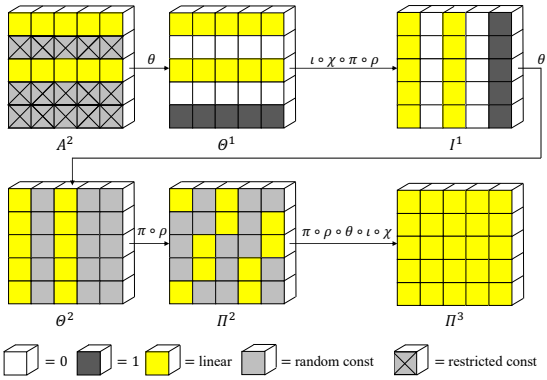


Fig. 3: The 2.5 rounds linear structure in [12].

constraints 2 to linearize the KECCAK-f up to 2.5 rounds while preserving up to 194 degrees of freedom which are used to establish a linear algebraic system to solve the preimage A^2 at low cost.

2.5 The quantum search

As an extension of the Grover's search algorithm [27], the amplitude amplification [28] is expected to find one of the solutions for a certain search problem with providing a quadratic speedup. Given a database $\mathcal{D} = \{x_1, \cdots, x_{2^n}\}$ that containing $N = 2^n$ elements, a boolean function $c(x) : \{0,1\}^n \to \{0,1\}$ is modeled to divide the database $\mathcal{D}$ into *good* subset $\mathcal{A} : \{x|c(x) = 1\}$ and *bad* subset $\mathcal{B} : \{x|c(x) = 0\}$. In quantum setting, we can design an oracle U_c to act function c on a superposition state $\sum_{x=1}^{N} \alpha_x |x\rangle$ in $\mathcal{O}(1)$ time, where $\alpha_{x'}$ denotes the probability amplitude corresponding to the state x'. Work on the equal superposition state $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{\alpha=1}^{N} |\alpha\rangle$, with the help of the phase kickback protocol, every time applying the oracle U_c followed by a diffusion operator $U_{\psi^\perp} = -I + 2|\psi\rangle\langle\psi|$, a step known by the Grover iteration $G = U_{\psi^\perp} U_c$, the amplitudes of all the elements in subset $\mathcal{A}$ will be be slightly amplified. After repeating this process about $t = \sqrt{N/|\mathcal{A}|}$ times, we can obtain a solution $x \in \mathcal{A}$ theoretically by measuring the superposition state.

In [29], Shenvi et al. presented a quantum walk based search algorithm called SKW algorithm, which can also be viewed as a rotation in a two-dimensional subspace. Compared with the amplitude amplification, in addition

to providing a quadratic speedup, that can exponentially reduce the implementation difficulty of the search algorithm. The only drawback of SKW algorithm is that the probability of finding a correct solution is about $\frac{1}{2}$.

2.6 The uncompute trick and qRAM

Here, we first talk the uncompute trick. The no-deleting theorem[30] tells us that there is no single-qubit unitary operator which can set an arbitrary qubit state to $|0\rangle$. Based on this point, one common technique that always be used to carry out a certain computation on a quantum register and then retrieve the initial state $|0\rangle$ is the uncompute trick, which can be found in the designs of many quantum attacks [21–23]. Specifically, For the given classifier $c(x)$ mentioned in previous section 2.5, there exists a corresponding classifier oracle U_c to enact the mapping $|x\rangle \rightarrow |c(x)\rangle$ which can be represented more precisely as $|x\rangle|0\rangle|0\rangle$ to $|x\rangle|g(x)\rangle|c(x)\rangle$, where the $|g(x)\rangle$ is a *garbage state*. Through the uncompute trick, we can return the garbage state $|g(x)\rangle$ to the initial state $|0\rangle$, in order that the later computations, such as the next Grover iteration, will not be interrupted. See [31] for a more detailed discussion.

As an important question of loading data into a quantum computer, qRAM is non-trivial to implement. In [32, 33], Giovannetti et al. and Park et al. focu on the possible memory structure of quantum random memory access. Given that the attackers assumed in cryptanalysis are powerful, many works such as [21–23] are based on an assumption that the attackers are equiped quantum computer with qRAM. Later, a negligible cost of qRAM is also required in our allocating rotational cryptanalysis based preimage attack.

3 The allocating rotational cryptanalysis

In this section, we focus on the presentation of new cryptanalysis technique which we called the allocating rotational cryptanalysis. We first propose new observations of the rotational cryptanalysis based preimage attack on 4-round KECCAK-224/256 both in the classical setting and quantum setting. Subsequently, the allocating rotational cryptanalysis will be introduced to mount a new 2-block preimage attack, and the main idea of the attack is elaborated in detail. We are committed to showing that how to weaken the operators θ and χ on the first two rounds of second hash process under a specified search space where the rotational counterparts of preimage can be found. Then, one way to reduce the complexity of precomputation is shown in order that the complexity of preimage attack based on the allocating rotational cryptanalysis is low enough. Finally, the theoretical preimage attack against 4-rounds KECCAK-224 is shown at the end of this section.

3.1 New observations

When using rotational cryptanalysis [9] to find the preimage of 4-round KECCAK, which only considing 1-block preimage, the attack complexity is expected

to be one-sixty-fourth of that of random preimage attack. However, since there are not enough eigenpoints after the propagation of 3.5 rounds, attacker needs to perform a large number of 4-round KECCAK to verify the correctness of guessed rotational counterpart. Especially, in the cases of variants KECCAK-256 and KECCAK-224, the advantages of rotational cryptanalysis driven preimage attacks over generic preimage attack are not obvious. Under the quantum counterpart of rotational cryptanalysis [26], the remaining eigenpoints after the propagation of 3.5 rounds are used to build the oracle operator to mark the possible rotational number of guessed rotational counterpart. The fewer eigenpoints, the lower accuracy of the oracle operator as a classifier, and resulting in a higher computing cost in the qunatum preimage attack. In short, if we can obtain as many eigenpoints as possible after the propagation of 3.5-round or even 4-round KECCAK, we can make an improvement on the preimage attacks in both classical and quantum setting. Given that the research about 1-block preimage attack based on the rotational cryptanalysis seems to have entered a bottleneck, here we combine the linearization of Keccak-f with the rotational cryptanalysis to present a new 2-block preimage attack as follows, which can obtain more eigenpoints after the propagation of 4-round KECCAK in the online phase.

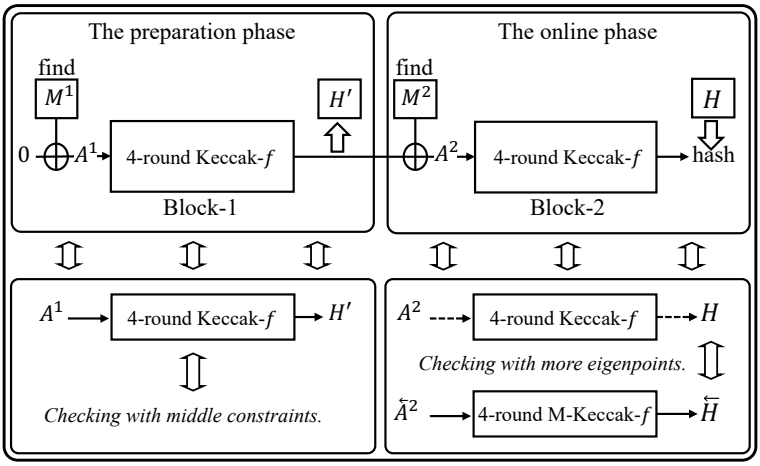


Fig. 4: The Allocating Rotational Cryptanalysis based Preimage Attack.

As shown in the Figure 4, for a given hash value H, we are aiming to find two preimage: A^1(the first messaged block) and A^2(the second messaged block) in their respective blocks instead of a 1-block one. Specifically, for the block-1, the output state of preimage A^1 must meets the constraints 1: $H'_{3,3,z} \oplus 1 = H'_{3,4,z}, H'_{4,3,z} \oplus 1 = H'_{4,4,z}$, $\bigoplus_{x,z} H'_{x,4,z} = 0$. We use the generic preimage attack to find a desired preimage A^1 with low complexity. For the block-2, owing to the last 7 lanes in the output state H' of block-1 and the second messaged state A^2 are identical, we can make a lot of candidate messaged states

A^2 by choosing the second message blocks M^2 carefully in order that all the candidate messaged states A^2 meet the conditions 2: $A^2_{x,1,z} = A^2_{x,3,z} = A^2_{x,4,z} \oplus 1$, $\bigoplus_{x,z} A^2_{x,4,z} = 0$. It is noting that these candidate messaged states A^2 form the specified search space for us to look for the correct second preimage A^2 via the rotational cryptanalysis, which is totally different from the [12] that solving the linear equations system to work out the proper second messaged block A^2. And we will show later that the propagation of the rotational relations between the each state of the specified search space and their rotational counterparts will leave more eigenpoints through the 4-round modified KECCAK.

3.2 Explore the new propagations of the rotational relations.

Before mounting a preimage attack on the KECCAK−224, we need to first analyze how and why these candidate messaged states A^2 with their rotational counterparts make a good propagation of rotational relations. In the work [12], the given values of constant bits $A^2_{x,1,z}$, $A^2_{x,3,z}$, and $A^2_{x,4,z}$, which satisfy the equqtions 2: $A^2_{x,1,z} = A^2_{x,3,z} = A^2_{x,4,z} \oplus 1$; $\bigoplus_{x,z} A^2_{x,4,z} = 0$, make sure the following equations have certain solutions for $\{S_{x,z}\}$,

$$A^2_{x,4,z} \oplus S_{x-1,z} \oplus S_{x+1,z-1} = 1 \tag{3}$$

where $0 \le x < 5$ and $0 \le z < 64$. That means the sum value of column $A_{x,z}$ can be work out as follows:

$$A^2_{x,0,z} \oplus A^2_{x,1,z} \oplus A^2_{x,2,z} \oplus A^2_{x,3,z} \oplus A^2_{x,4,z} = A^2_{x,0,z} \oplus A^2_{x,2,z} \oplus A^2_{x,4,z} = S_{x,z};$$

and

$$A^2_{x,0,z} \oplus A^2_{x,2,z} = A^2_{x,4,z} \oplus S_{x,z}.$$

Note that the $A^2_{x,4,z}$ and $S_{x,z}$ are constant bits, and therefore the unknown bits pair $(A^2_{x,0,z}, A^2_{x,2,z})$ has only two possible values to match with these constraints: $A^2_{x,4,z} \oplus S_{x-1,z} \oplus S_{x+1,z-1} = 1$. Without loss of generality, we suppose the $A^2_{x',4,z'} = 1$ and $S_{x',z'} = 0$, then the $(A^2_{x',0,z'}, A^2_{x',2,z'}) = (0,1)$ or $(1,0)$. Recall that we have $5 \times 64 = 320$ columns, and the number of the candidate messaged states A^2 that match all these constraints 3 is 2^{320}.

Then, let's talk about how the first operator θ of block-2 has been disturbed under these 2^{320} candidate messaged states A^2 for the better propagation of the rotational relations. According to the definition of operator θ, we have:

$$\begin{aligned} \Theta^1_{x,y,z} &= A^2_{x,y,z} \oplus S_{x-1,z} \oplus S_{x+1,z-1} \\ &= A^2_{x,y,z} \oplus A^2_{x,4,z} \oplus A^2_{x,4,z} \oplus S_{x-1,z} \oplus S_{x+1,z-1} \\ &= A^2_{x,y,z} \oplus A^2_{x,4,z} \oplus 1. \end{aligned}$$

Figure 5 intuitively reflects what happened on messaged states A^2, and the

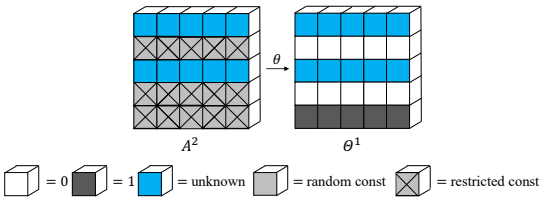


Fig. 5: What happened on the second messaged block A^2 after the first θ.

first map θ of the block-2 applied on these special messaged states A^2 is actually equivalent to the following:

$$\Theta^1_{x,y,z} = A^2_{x,y,z} \oplus A^2_{x,4,z} \oplus 1. \tag{4}$$

Note that we no longer consider the linearity of bits throughout our work, and any XOR operation between constant bits and unknown bits obviously results in unknown bits. To distinguish, we use a blue cube to indicate that this bit is unknown.

As shown in Figure 6, after the candidate messaged states A^2 throughing the maps θ, ρ and π of first round, the states Π^1 have a good property which the unknown bits are separated into discontinuities by the constant bits 0 or 1 in each row. So, according to the definition of the map $\chi : X_{x,y,z} =$

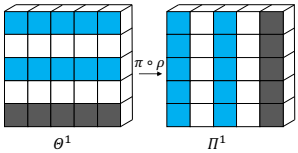


Fig. 6: The maps ρ and π only change the position of bits.

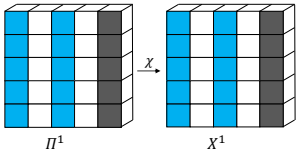


Fig. 7: The columns 1, 3 and 4 keep unchanged.

$\Pi_{x,y,z} \oplus (\Pi_{x+1,y,z} \oplus 1) \cdot \Pi_{x+2,y,z}$, the first operator χ of the block-2 works as follows:

$$\begin{aligned}
X^1_{0,y,z} &= \Pi^1_{0,y,z} \oplus (0 \oplus 1) \cdot \Pi^1_{2,y,z} = \Pi^1_{0,y,z} \oplus \Pi^1_{2,y,z};\\
X^1_{1,y,z} &= 0 \oplus (\Pi^1_{2,y,z} \oplus 1) \cdot 0 = 0;\\
X^1_{2,y,z} &= \Pi^1_{2,y,z} \oplus (0 \oplus 1) \cdot 1 = \Pi^1_{2,y,z} \oplus 1;\\
X^1_{3,y,z} &= 0 \oplus (1 \oplus 1) \cdot \Pi^1_{0,y,z} = 0;\\
X^1_{4,y,z} &= 1 \oplus (\Pi^1_{0,y,z} \oplus 1) \cdot 0 = 1.
\end{aligned}$$

In fact, as shown in Figure 7, at this point the first χ only change the first and third columns of each slice, and no operations AND are involved between the unknown bits. Therefore, the effect of χ corresponds to the following:

$$X^1_{0,j,z} = \Pi^1_{0,j,z} \oplus \Pi^1_{2,j,z};\quad X^1_{1,j,z} = \Pi^1_{1,j,z};\quad X^1_{2,j,z} = \Pi^1_{2,j,z} \oplus 1;$$

$$X^1_{3,j,z} = \Pi^1_{3,j,z}; \quad X^1_{4,j,z} = \Pi^1_{3,j,z}. \tag{5}$$

Now, we focus on the maps in the second round. We first select a 5×64 constant array C randomly, and use $C_{x,y}$ to denote the element of C, where $x \in \{0, 2\}$ and $0 \leq z < 64$. To weaken the second θ and χ in the block-2, we need to make sure that the first round output states I^1 of these candidate messaged states A^2 satisfy the following 128 constraints:

$$\bigoplus_{j=0}^{4} I^1_{0,j,z} = C_{0,z}; \quad \bigoplus_{j=0}^{4} I^1_{2,j,z} = C_{2,z}, \tag{6}$$

where $0 \leq z < 64$. Namely, for a output state I^1 we require the sums of bits in column 0 of slice z is equal to constant $C_{0,z}$. Therefore, the second θ of block-2 can be reduced as follows:

$$\begin{aligned}
\Theta^2_{0,y,z} &= I^1_{0,y,z} \oplus \bigoplus_{j=0}^{4} I^1_{4,j,z} \oplus \bigoplus_{j=0}^{4} I^1_{1,j,z-1} = I^1_{0,y,z} \oplus 1;\\
\Theta^2_{1,y,z} &= I^1_{1,y,z} \oplus \bigoplus_{j=0}^{4} I^1_{0,j,z} \oplus \bigoplus_{j=0}^{4} I^1_{2,j,z-1} = I^1_{1,y,z} \oplus C_{0,z} \oplus C_{2,z-1};\\
\Theta^2_{2,y,z} &= I^1_{2,y,z} \oplus \bigoplus_{j=0}^{4} I^1_{1,j,z} \oplus \bigoplus_{j=0}^{4} I^1_{3,j,z-1} = I^1_{2,y,z};\\
\Theta^2_{3,y,z} &= I^1_{3,y,z} \oplus \bigoplus_{j=0}^{4} I^1_{2,j,z} \oplus \bigoplus_{j=0}^{4} I^1_{4,j,z-1} = I^1_{3,y,z} \oplus C_{2,z} \oplus 1;\\
\Theta^2_{4,y,z} &= I^1_{4,y,z} \oplus \bigoplus_{j=0}^{4} I^1_{3,j,z} \oplus \bigoplus_{j=0}^{4} I^1_{0,j,z-1} = I^1_{4,y,z} \oplus C_{0,z-1}.
\end{aligned}$$

In other words, for this special case the equivalent of map θ can be described by the following formulas:

$$\begin{aligned}
&\Theta^2_{0,y,z} = I^1_{0,y,z} \oplus 1; \quad \Theta^2_{1,y,z} = I^1_{1,y,z} \oplus C_{0,z} \oplus C_{2,z-1}; \quad \Theta^2_{2,y,z} = I^1_{2,y,z}\\
&\Theta^2_{3,y,z} = I^1_{3,y,z} \oplus C_{2,z} \oplus 1; \quad \Theta^2_{4,y,z} = I^1_{4,y,z} \oplus C_{0,z-1},
\end{aligned} \tag{7}$$

and shown the form of states Θ^2 in the Figure 8. Finally, we need to focus on how the second χ changes the bits $\Pi^2_{x,y,z}$ indeed. As shown in the Figure 9, the obtained states Π^2 from states Θ^2 still keep the good properties that is the unknowns bits in each row are discontinuous. Therefore, like the first χ shown in the formulas 5, there sill are no operations AND involved between the unknown bits for the second χ. However, the constant bits of each row in states Π^2 are deeply bound to the constant array C we selected in formulas 6. For a set of given constants $C_{x,z}$, the specific forms of map χ are different in

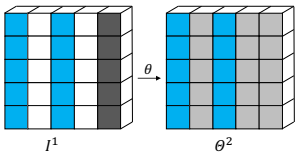


Fig. 8: All the bits of columns 1, 3 and 4 are constant.

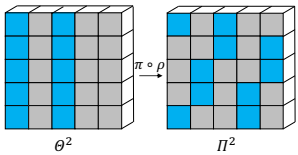


Fig. 9: The unknown bits of states Π^2 keep the discontinuity in each row.

each row. Between different sets of constants $C_{x,z}$, the specific forms of map χ in a same row are still different. Therefore, we cannot give a unified algebraic form of the second reduced χ here. For a given set of constants $C_{x,z}$, we will use the program to simulate the second χ of block-2 and calculate the states X^2.

Summarily, when the candidate messaged states A^2 that have the form 2 and meet two sets of constraints 3, 6 are used as the input of 4-round Keccak, the maps θ in the first two rounds are equivalent to the formulas: 4 and 7, respectively, and both maps χ in the first two round do not involve the operation AND. Now, the last thing we need to do before tracking the propagation of rotational relations between the candidate messaged state A^2 and the corresponding rotational counterpart $\overleftarrow{A}^2$ is to make sure that the $\overleftarrow{A}^2$ is running on the same mappings θ and χ as A^2. To achieve this, there are only four possible choices for the constant array C mentioned above:

$$C^1 = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \end{pmatrix}; \; C^2 = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \end{pmatrix};$$

$$C^3 = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \end{pmatrix}; \; C^4 = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{pmatrix}.$$

Note that for any rotational number, the rotational counterparts corresponding to these canstant arrays are themselves. The following Theorem 3.1 ensures that the candidate messaged state A^2 and the corresponding rotational counterparts $\overleftarrow{A}^2$ share the same set of maps involved in the first two round when the constant array C is chosen from one of above arrays.

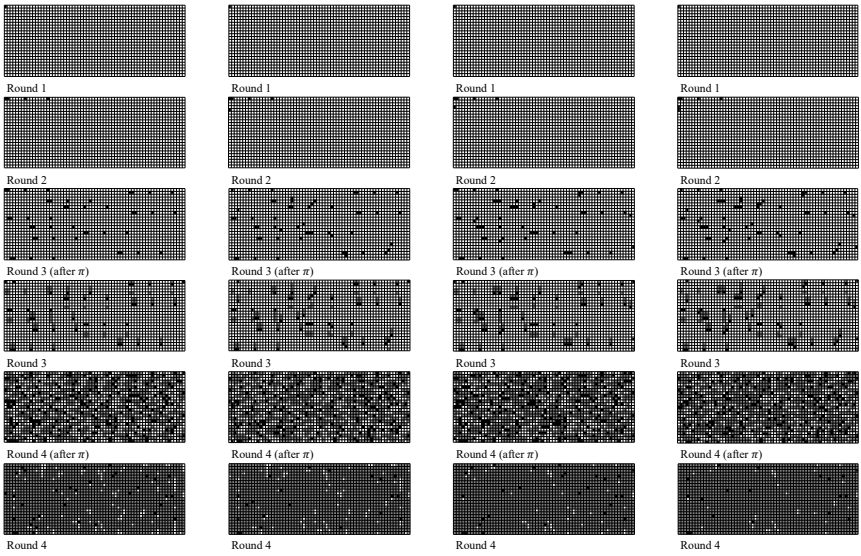
Theorem 3.1 *Let the constant array C be one of the four we found, then, for any rotational number n, the rotational counterparts $\overleftarrow{A}^2$ still satisfy the two set constraints 3 and 6, and therefore shares the same maps with the candidate messaged states A^2.*

Now, we can explore four new propagations of rotational relations, and we expect the eigenpoints to be better retained. In fact, the following Theorem 3.2 theoretically guarantees that all the eigenpoints in each newly explored propagation are fully preserved at the end of the 2.5-round Keccak. The

relevant proofs of Theorem 3.1 and Theorem 3.2 can be found in the Appendix A.

Theorem 3.2 $\forall\ (x, y, z) \in \{0, \cdots, 4\} \times \{0, \cdots, 4\} \times \{0, \cdots, 63\}$, *the rotational relation* $P_{x,y,z}$ *between any state array* A^2 *selected from the specified search space* Γ *and the corresponding rotational counterpart* $\overleftarrow{A}^2$ *can be used as an eigenpoint after 2.5 rounds Rnd.*

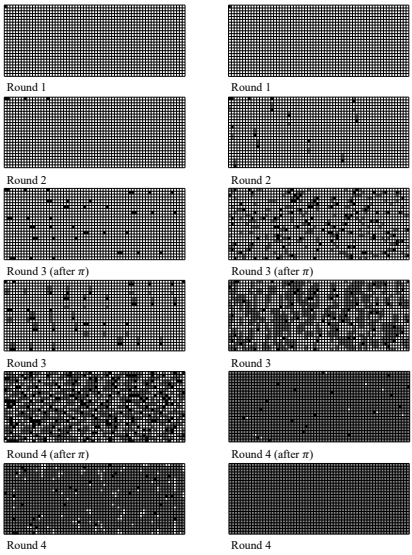
Here, the following Figure 10 shows all 4 improved propagations of rotational relations while the constant array C is chosen to be C^1, C^2, C^3 and C^4, respectively. Note that the white block denotes the eigenpoint for $P_{x,y,z} = 0$,



(a) For the C^1. (b) For the C^2. (c) For the C^3. (d) For the C^4.

Fig. 10: The improved propagations for different constant arrays.

the black block denotes the eigenpoint for $P_{x,y,z} = 1$ and the gray block denotes the rotational relation for $0 < P_{x,y,z} < 1$ which cannot provide a clear information between the bit value $A^2_{x,y,z}$ and the corresponding bit value $\overleftarrow{A}^2_{x,y,z}$. Fortunately, it's quite obvious that our experimental results strongly support our theoretical analysis. Specifically, at the end of the 2.5-round, only white blocks and black blocks as eigenpoints are involved, and at the end of 3.5-round, there are 125, 114, 107 and 99 eigenpoints in the first 3.5 lanes, respectively. As shown in the Figure 11, the number of eigenpoints remaining at the end of the 3.5-round shown on the left is approximately 40 times higher compared to the work of Morawiecki et al. shown on the right. Furthermore,



(a) The best one. (b) The original one.

Fig. 11: Compared to the original propagation introduced in [9]

we succeeded in keeping the eigenpoints to the end of 4-round KECCAK-224 with 32, 21, 16 and 12 respectively. Contrarily, the original propagation of Morawiecki et al no longer has any usable information at the same rounds.

Another interesting point we found is that the later and fewer eigenpoints for $P_{x,y,z} = 1$ appear will allow a better propagation of the rotational relations, resulting in keeping more eigenpoints at the end. For instance, the number of eigenpoints for $P_{x,y,z} = 1$ at the end round 2 in the Figure 10(c) is only one more than that in the Figure 10(a), yet after the propagation of round 3 and 4, the final number of eigenpoints in the Figure 10(c) is only half of that in the Figure 10(a). Therefore, we believe that the propagation of rotational relations is still consistent with the avalanche effect in cryptography.

3.3 Match the constant array C without running Keccak

Here, we focus on the preparation of the the specified search space Γ for the searching of second message block at a low cost. Recall that each element of the specified search space Γ has 640 unknown bits and must meet two sets of constraints. First, at the beginning of block-2, the 320 linear constraints 3 are satisfied. Second, all the output states I^1 of the first Rnd satisfy the 128 linear constraints 6. In the previous section, we have discussed a feasible method to quickly generate a large number of state arrays A^2 that meet the first set of constraints. However, one way to efficiently obtain the second-constraints-satisfied candidate messaged states A^2 is not obvious. A direct approach is taking all the 2^{320} state arrays A^2 as inputs of first Rnd and

calculating 2^{320} output states I^1. After checking every I^1 whether meets the constraints 6, we can obtain $2^{320-128} = 2^{192}$ candidate messaged states A^2 which meet all the two sets of constraints, and finally form the specified search space Γ. Unfortunately, this method needs to apply a large number of 1-round KECCAK, and the computational cost is unbearable. The work we show here puts forward a new idea: For any given constant array C, we successfully find a set of new linear constraints which are used to describe the relations between the unknown bits of candidate messaged states A^2 and the bit values $C_{x,y}$. This allows us to use these new constraints at the beginning of the block-2 to quickly generate a large number of candidate messaged states A^2, resulting in the corresponding outputs of 1-round KECCAK just meet the constraints 6.

According to the equivalent 5 of first χ, only the column 0 and column 3 of each slice are changed as follows:

$$X^1_{0,j,z} = \Pi^1_{0,j,z} \oplus \Pi^1_{2,j,z}; \quad X^1_{2,j,z} = \Pi^1_{2,j,z} \oplus 1;$$

Then we have:

$$C_{2,z} = \bigoplus_{j=0}^{4} X^1_{2,j,z} = \bigoplus_{j=0}^{4} \Pi^1_{2,j,z} \oplus 1,$$

$$\bigoplus_{j=0}^{4} \Pi^1_{2,j,z} = C_{2,z} \oplus 1.$$

So,

$$C_{0,z} = \bigoplus_{j=0}^{4} X^1_{2,j,z} = \bigoplus_{j=0}^{4} (\Pi^1_{0,j,z} \oplus \Pi^1_{2,j,z})$$

$$= \bigoplus_{j=0}^{4} \Pi^1_{0,j,z} \oplus \bigoplus_{j=0}^{4} \Pi^1_{2,j,z} = \bigoplus_{j=0}^{4} \Pi^1_{0,j,z} \oplus C_{2,z} \oplus 1,$$

and

$$\bigoplus_{j=0}^{4} \Pi^1_{0,j,z} = C_{0,z} \oplus C_{2,z} \oplus 1.$$

For the sake of uniformity, here we ignore the effect of first ι that only flips the bit value $X^1_{0,0,0}$, and the case of involved column $X^1_{0,y,0}$ can be handled similarly.

Then, we need to find out the relations between the inputs $P_{x,y,z}$ of π and these constants $C_{x,z}$. According to the definition of π, for all triples (x, y, z), the π works as follows $\Pi_{x,y,z} = P_{(x+3y) mod\ 5,x,z}$. Therefore, we have:

$$\bigoplus_{j=0}^{4} \Pi^1_{0,j,z} = \bigoplus_{j=0}^{4} P^1_{j,0,z} = C_{0,z} \oplus C_{2,z} \oplus 1;$$

$$\bigoplus_{j=0}^{4} \Pi^1_{2,j,z} = \bigoplus_{j=0}^{4} P^1_{j,2,z} = C_{2,z} \oplus 1.$$

Now, we focus on the relations between the inputs Θ^1 of ρ and these constants $C_{x,z}$. The following Table 2 lists the *offset* involved here, while $P_{x,y,z} = \Theta_{x,y,(z\ +offset)mod\ 64}$. Without loss of generality, we take the case of

Table 2: The offsets in the row 0 and row 2.

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$
$y=0$	0	1	190	28	91
$y=2$	3	10	171	153	231

$z = 2$ as an example shown as follows:

$$\bigoplus_{j=0}^{4} P^1_{j,0,2} = P^1_{0,0,2} \oplus P^1_{1,0,2} \oplus P^1_{2,0,2} \oplus P^1_{3,0,2} \oplus P^1_{4,0,2}$$

$$= \Theta^1_{0,0,2} \oplus \Theta^1_{1,0,1} \oplus \Theta^1_{2,0,4} \oplus \Theta^1_{3,0,38} \oplus \Theta^1_{4,0,39} = C_{0,2} \oplus C_{2,2} \oplus 1$$

Similarly, we have:

$$\Theta^1_{0,2,63} \oplus \Theta^1_{1,2,56} \oplus \Theta^1_{2,2,23} \oplus \Theta^1_{3,2,41} \oplus \Theta^1_{4,2,27} = C_{2,2} \oplus 1.$$

Recall that first θ works as 4, we can therefore finally find out the new linear constraints for the second messaged states A^2 as follows:

$$\begin{aligned} &A^2_{0,0,2} \oplus A^2_{1,0,1} \oplus A^2_{2,0,4} \oplus A^2_{3,0,38} \oplus A^2_{4,0,39} \oplus \\ &A^2_{0,4,2} \oplus A^2_{1,4,1} \oplus A^2_{2,4,4} \oplus A^2_{3,4,38} \oplus A^2_{4,4,39} \\ &= C_{0,2} \oplus C_{2,2}; \end{aligned}$$

$$\begin{aligned} &A^2_{0,2,63} \oplus A^2_{1,2,56} \oplus A^2_{2,2,23} \oplus A^2_{3,2,41} \oplus A^2_{4,2,27} \oplus \\ &A^2_{0,4,63} \oplus A^2_{1,4,56} \oplus A^2_{2,4,23} \oplus A^2_{3,4,41} \oplus A^2_{4,4,27} \\ &= C_{2,2}, \end{aligned}$$

and the complete constraints are shown in the Appendix B. It is worth remembering that all the bits located in the plane 4 of candidate messaged states A^2 and random bits $C_{x,z}$ are constants, and all the unknown bits are located in the plane 0 and plane 2 of candidate messaged states A^2.

In the block-2, recall that the last 7 lanes in the output state H' of latest block and the second messaged state A^2 are identical. By using two set of constraints, the linear constraints 3 and the new explored constraints shown in Appendix B, we can therefore efficiently obtain totall $2^{640-320-128} = 2^{192}$ candidate messaged states A^2 for a candidate preimage A^1. In order to be able to find the preimage A^2 of the hash value H, we need the specified search space Γ which consists of at least 2^{224} candidate messaged states A^2. That's why we need to execute the preparation phase $2^{224-192} = 2^{32}$ times to get 2^{32} different candidate preimages A^1.

3.4 The preimage attack on 4-round Keccak-224

Now, we describe our preimage attack on the 4-round KECCAK-224 in detail and give a complete complexity analysis. It is worth mentioning here that we consider the cost of running once 4-round KECCAK-224 is equal to $\mathcal{O}(1)$ both in the classical setting and the later quantum setting. As shown in Figure 4, our preimage attack is divided into two phases. In the preparation phase, we first need to use the generic preimage attacks to find 2^{32} candidate preimages A^1 to ensure that all these corresponding middle states meet the constraints 1. For a random state array of A^1, the probability of satisfying all the $2 \times 64 + 1 = 129$ constraints is equal to 2^{-129}. Therefore, we can expect to obtain one desired preimage A^1 form searching 2^{129} random values of A^1, and the cost of the preparation phase is equal to $2^{129} \times 2^{32} = 2^{161}$.

With the help of all the 2^{32} candidate preimages A^1, we quickly generate the specified search space Γ which has 2^{224} candidate messaged states A^2 in total. According to the rotational cryptanalysis, given a 224 bits hash value, the probability that we guess one of the rotational counterpart $\overleftarrow{A}^2$ is $2^{-224} \times 64 = 2^{-218}$. Thus we need to select 2^{218} state arrays $\overleftarrow{A}^2$ which are the rotational counterparts for some candidate messaged states A^2 to search a correct rotational counterpart of the certain second preimage state A^2. Here, we still use the Algorithm 1 proposed by Morawiecki et al [9] to find the corresponding rotational counterpart $\overleftarrow{A}^2$:

Algorithm 1 Search for the second messaged block A^2

Input: given a hash value
Output: the second preimage A^2

1: Randomly select a state array $\overleftarrow{A}^2$ as the input;
2: Run 4-round modified KECCAK on the state array $\overleftarrow{A}^2$;
3: **for** $n = 0$ to $n < 64$ **do**
4: Set flag $= 1$;
5: **for all** 32 eigenponts (x, y, z) **do**
6: **if** Bit $I^4_{x,y,z+n}$ and bit $H_{x,y,z}$ satisfy the relation as the eigenpoint (x, y, z) **then**
7: flag $=$ (flag $+1$) mod2;
8: **end if**
9: **end for**
10: **if** flag $== 1$ **then**
11: Rotate back the randomly select state by n bits and run 4-round Keccak-224 on it to check whether the state is the the second preimage A^2 of a given hash B.
12: **end if**
13: **end for**

After repeating the Algorithm 1 2^{218} times, we expect to obtain the second messaged block A^2, and by using these bit values of the latest 7 lanes in A^2, we can easily find the first messaged block A^1 from the 2^{32} candidate preimages of A^1. Compared to Morawiecki et al.'s attack on the 4-round Keccak-224, our improvement is that: thanks to the 32 eigenpoints we obtained in the previous section, the probability that the corresponding output state $\overleftarrow{H}$ of a random state array $\overleftarrow{A}^2$ matches the hash value H at all these eigenpoints is greatly reduced from 2^{-3} to 2^{-32}, which pushes forward the Morawiecki et al.'s conclusions. Therefore, there will be only around $2^{224}/2^{32} = 2^{192}$ false state arrays A^2 that need to be checked in step 11 by running the 4-round Keccak-224, resulting in a lower computational cost. Summing up, the workload of the attack is 2^{161} (the preparation phase) + $(2^{218} + 2^{192})$ (the online phase), and the complexity of the attack is roughly 2^{218} times of 4-round Keccak-224 calls, which is about 64 times better than the generic preimage attack. It is worth mentioning that we will not discuss the case where the second messaged block A^2 is a cyclic pattern with less than 64 rotational counterparts, which has almost no effect on our preimage attack.

4 The new preimage attack in the quantum setting

In this section, we mainly show the detail implementation of quantum theoretical preimage attack driven by our allocating rotational cryptanalysis. We first design several unitary oracle operators as the classifiers to help us mark the desired preimage blocks A^1 and A^2 at their respective phase. Then, we show how the quantum search algorithms work with these classifiers to find the proper preimage blocks A^1 and A^2 for a hash value H of 4-round Keccak-224, and give the complexity analysis of our preimage attack.

4.1 Designing the oracle operators for rotational allocating approach

For searching the first preimage block A^1, we choose the generic quantum preimage attack. Recall that the probability of satisfying all the $2\times64+1 = 129$ constraints 1 is equal to 2^{-129}. Therefore, we need a unitary oracle operator to distinguish the proper preimage block A^1 from a superposition state composed of 2^{129} state arrays. The constraints 1 can obviously be used as the key to classification criteria, and we give a feasible definition of the oracle function $f(A^1) : \{0,1\}^{129} \to \{0,1\}$ for the unitary oracle operator $U_f : |x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle$ which is committed to marking the proper preimage block A^1. As shown in Algorithm 2, the pseudo-codes of the implementation of oracle operator U_f is given below:

Algorithm 2 Implementation of the oracle operator U_f

Input: $|A^1\rangle|y\rangle$
Output: $|A^1\rangle|y \oplus f(A^1)\rangle$
1: For a given state array A^1, set $flag = 0$;
2: Run 4-round KECCAK-224 on the state array A^1;
3: Check whether the output state satisfying all the constraints 1; If so, set the $flag = 1$; Do nothing otherwise;
4: Return 1 as the value of $f(A^1)$ if and only if $flag = 1$; Return 0 otherwise;
5: Uncompute Steps $2 - 3$.

It is worth mentioning that the ancillary qubit $|y\rangle$ is usually in the state: $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. After applying the oracle U_f, we have the state $|A^1\rangle\frac{|0\oplus f(A^1)\rangle-|1\oplus f(A^1)\rangle}{\sqrt{2}} = (-1)^{f(A^1)}|A^1\rangle|-\rangle$. Therefore, the oracle U_f can only be used to mark a state $|A^1\rangle$ which are acted as the proper preimage A^1. This efficient classification method is known as "Phase Kickback" in the quantum computing, and will be used frequently later. The detailed quantum circuit model of the oracle operator U_f is shown in the Figure 12. Here, we use the

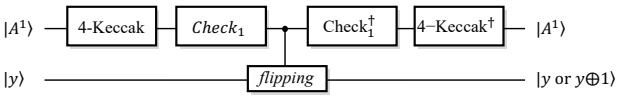


Fig. 12: The quantum circuit model of the unitary oracle operator U_f.

block "$Check_1$" to denote the step 3 of Algorithm 2, and the block "flipping" to represent the process of phase kickback. Note that the blocks with † are the uncomputing counterparts of those without the †, and this will be used again later.

Now let's focus on the marking of second preimage block A^2. We want to find one of the corresponding rotational counterparts $\overleftarrow{A}^2$ to obtain the second preimage block A^2 indirectly. To judge whether a state array $\overleftarrow{A}^2$ is a rotational counterpart of A^2, we must rotate back the state array $\overleftarrow{A}^2$ by using a proper rotational number n. Therefore, our first job is to find the proper rotational number n for a rotational counterpart by using the 32 eigenpoints found in the section 3.2. Concretely, for a rotational counterpart $\overleftarrow{A}^2$ acted as the input of 4-round modified KECCAK-224, the proper rotational number n ensures that the corresponding output state $\overleftarrow{H}$ and the hash value H satisfy all the relations determined by these eigenpoints. Here, we define an oracle function $N(n, \overleftarrow{H}) : \{0,1\}^{112} \to \{0,1\}$ to help the unitary oracle operator $U_N : |n\rangle|\overleftarrow{H}\rangle|y\rangle \to |n\rangle|\overleftarrow{H}\rangle|y \oplus N(n, \overleftarrow{H})\rangle$ mark the proper rotational number n

of the rotational counterpart $\overleftarrow{A}^2$, and the following Algorithm 3 spreads the implementation details of U_N.

Algorithm 3 Implementation of oracle operator U_N.

Input: $|n\rangle|\overleftarrow{H}\rangle|y\rangle$
Output: $|n\rangle|\overleftarrow{H}\rangle|y \oplus N(n, \overleftarrow{H})\rangle$

1: For a given rotational number n, set $flag = 0$;
2: **if** $0 \leq n < 64$ **then** Check if these matched values of $\overleftarrow{H}$ satisfy all the 32 relations with the corresponding values of H by accessing these eigenpoints stored in the qRAM. If so, set $flag = 1$. Do nothing otherwise;
3: **else** Do nothing;
4: **end if**
5: Return 1 as the value of oracle function $N\left(n, \overleftarrow{H}\right)$ if and only if $flag = 1$, and return 0 otherwise;
6: Uncompute the Steps $2 - 4$.

Note that throughout this process, the given state value $\overleftarrow{H}$ is a constant value and the rotational number n is the variable. The following Theorem 4.1 makes sure that the oracle U_N can flip the phase of the proper rotational number $|n\rangle$ for the rotational counterpart $\overleftarrow{A}^2$ by using the phase kickback with high efficiency.

Theorem 4.1 *Given a superposition state* $\sum_{n\in\{0,1\}^{112}} \alpha_n|n\rangle$*, the oracle* U_N *marks the state* $|n\rangle$ *only if* n *is the rotational number of rotational counterpart* $\overleftarrow{A}^2$ *corresponding to the preimage block* A^2*.*

The proofs about this theorem can be found in Appendix A. As what we have down for the oracle U_f, the detailed quantum circuit model of the oracle operator U_f is shown in the Figure below 13, where the block "$Check_2$" is used to denote the step 2 of Algorithm 3.

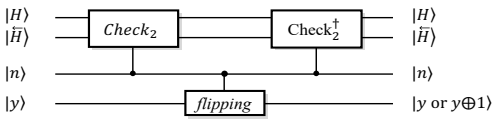


Fig. 13: The quantum circuit model of the unitary oracle operator U_N.

After having the ability to find the proper rotational number n, we now concentrate on marking the $\overleftarrow{A}^2$ of preimage block A^2. Recall that for a given state array there are up to 64 possible rotational counterparts. We have prepared the

specified search space Γ which has total 2^{224} state array to search the preimage A^2 for a given 224 bits hash value H. Furthermore, we only need to build a superposition state composed of $2^{224}/64 = 2^{218}$ state arrays $\overleftarrow{A}^2$ obtained from the elements of Γ. Similarly, a boolean function $F(\overleftarrow{A}^2) : \{0,1\}^{218} \to \{0,1\}$ for the unitary oracle operator $U_F : |\overleftarrow{A}^2\rangle|y\rangle \to |\overleftarrow{A}^2\rangle|y \oplus F(\overleftarrow{A}^2)\rangle$ is defined here to mark one of the rotational counterparts $\overleftarrow{A}^2$ for the second preimage A^2. The Algorithm 4 listed below gives the pseudo-codes of the implementation of oracle U_F.

Algorithm 4 Implementation of oracle operator U_F.

Input: $|\overleftarrow{A}^2\rangle|y\rangle$
Output: $|\overleftarrow{A}^2\rangle|y \oplus F(\overleftarrow{A}^2)\rangle$

1: For the given state array $\overleftarrow{A}^2$, set $flag_1 = flag_2 = 0$;
2: Run 4-round modified KECCAK-224 on the state array $\overleftarrow{A}^2$ to obtain the output state $\overleftarrow{H}$;
3: Run the amplitude amplification on the oracle U_N to find the proper rotational number n; If $0 \leq n < 64$, set $flag_1 = 1$, else, set $flag_1 = 0$;
4: Rotate back the state array $\overleftarrow{A}^2$ by n bits to obtain the state array A^2, and verify if all the 112 equations are satisfied; If so, set $flag_2 = 1$, else, set $flag_2 = 0$;
5: Return 1 as the value of oracle function $F(\overleftarrow{A}^2)$ if and only if $flag_1 = flag_2 = 1$, and return 0 otherwise;
6: Uncompute the Steps $2 - 4$.

It is worth mentioning that in the step 3 of Algorithm 4, we embed the amplitude amplification running on the oracle U_N into the implementation of the unitary oracle operator U_F. Similar oracle design schemes also appeared in [20, 22, 26]. The following Theorem 4.2 shows that this method requires less computational cost than the way adopted by Wang et al. in [26].

Theorem 4.2 *The oracle operator U_F can work both on the rotational counterpart $\overleftarrow{A}^2$ of the second preimage A^2 and other random string with only twice running the 4-round modified* KECCAK-224.

For a more detailed discussion, see the proofs of Theorem 4.2 in Appendix A. The final detailed quantum circuit model for the oracle operator U_F is shown in the Figure 14, where the block "$Check_3$" denotes the step 4 of Algorithm 4. Note that the operation of the state array $\overleftarrow{A}^2$ is still involved in the step 4 of Algorithm 4, which makes it impossible for us to directly run the 4-round modified KECCAK-224 on state array $\overleftarrow{A}^2$ in the step 2 of Algorithm 4. That's

why we need to allocate auxiliary qubits $|0\rangle^{\otimes 1600}$ to run the 4-round modified KECCAK-224 with state array $\overleftarrow{A}^2$ as input.

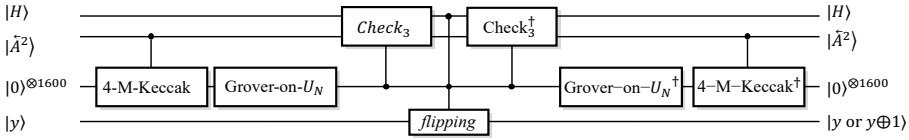


Fig. 14: The quantum circuit model of the unitary oracle operator U_F.

4.2 Quantum Preimage Attack on 4-round Keccak-224

In the quantum setting, we use the generic quantum preimage attacks to find 2^{32} candidate preimages A^1 for the preparation phase. Specifically, prepare the initial 130-qubit state $|0\rangle$ first. After applying Hadamard $H^{\oplus 130}$, an equal superposition state $|\psi\rangle = \sum_{A^1\in\{0,1\}^{130}} |A^1\rangle$ build on these qubits, where we omit the amplitudes 2^{-65} for ease of exposition. Now, by using the amplitude amplification on the oracle U_f to this equal superposition state $|\psi\rangle$, we can obtain a superposition state $\sum_{A^1\in 0,1^{129}}(-1)^{f(A^1)}|A^1\rangle$. In each time of Grove iteration, the correct candidate preimage A^1 is marked out and its amplitude is slightly increased. After about $\mathcal{O}(2^{65})$ Grover iterations, we can expect to obtain a correct candidate preimage A^1 by measuring the final superposition state. Since the single call to the oracle U_f contains two executions of 4-round KECCAK-224, we believe that the cost of the preparation phase is around $2 \times 2^{65} \times 2^{32} = \mathcal{O}(2^{98})$.

Concretely in the online phase, we require about $224 + 218 + 1600$ qubits for the other half attack. After simple processing, our attack will start from the equal superposition state $|\psi\rangle = \sum_{\overleftarrow{A}^2\in\{0,1\}^{218}} |H\rangle \otimes |\overleftarrow{A}^2\rangle \otimes |0\rangle^{\otimes 1600}$. While applying the oracle U_F, the middle 218 qubits of $|\overleftarrow{A}^2\rangle$ acted as the input value to help us write the output value $|\overleftarrow{H}\rangle$ of 4-round modified KECCAK-224 to the last 1600 qubits of $|0\rangle^{\otimes 1600}$, then the $|\overleftarrow{H}\rangle$ and the hash value $|H\rangle$ are matched to find a possible rotational number n for the $|\overleftarrow{A}^2\rangle$, and finally the oracle U_F can mark the correct second preimage block $\overleftarrow{A}^2$. Similar to the previous preparation phase, after about $2^{218/2} = \mathcal{O}(2^{109})$ Grover iterations of the oracle U_F, the desired $\overleftarrow{A}^2$ can be obtained by measuring the middle 218 qubits. Thanks to the oracle U_F still contains only two executions of 4-round modified KECCAK-224, which is about half the cost of Wang et al. in [26], the cost of the online phase is estimated to be around $2 \times 2^{109} = \mathcal{O}(2^{110})$. It is easy to get the second preimage block A^2 from a correct rotational counterpart $\overleftarrow{A}^2$, and we can use the method in section 3.4 to finally determine the first preimage block A^1 from the 2^{32} possible values.

The SKW quantum algorithm can also be used for the preparation phase and the online phase to greatly reduce the difficulty of building quantum circuits for our preimage attack. In the work of Wang et al. in [26], the oracle operator only has less than 1/8 chance of successfully marking the rotational counterpart of preimage, as a result of few eigenpoints they found. Since the oracle operator U_F introduced in our attack works with more than 30 newly explored eigenpoints, the possibility of marking the correct rotational counterpart $\overleftarrow{A}^2$ is almost 1, and therefore the SKW based preimage attack still costs(the preparation phase: $\mathcal{O}(2^{99})$; the online phase: $\mathcal{O}(2^{111})$) less than the generic quantum preimage attack.

5 Conclusion

In this paper we have presented an allocating rotational cryptanalysis based preimage attack on the 4-round KECCAK-224, which has a good performance in quantum setting. By using more than 30 eigenpoints supported by the better propagation of rotational relations, an improvement of about 10 orders of magnitude on the accuracy of determining the rotational number of rotational counterpart has been achieved in the quantum setting. At the same time, we have proposed a method to quickly expand the state arrays that follow the new propagation into a specified search space in which the second preimage state can be found indirectly. As an outcome, an efficient oracle operator that requires only twice calling of the 4-round modified KECCAK-f has been designed at the end of this paper to mark the rotational counterpart of the second preimage state, which leads to the best amplitude amplification based dedicated quantum preimage attack to date against the 4-round KECCAK-224. Even the SKW algorithm, easy to achieve but low probability of success, has been given the ability to work with the efficient oracle operator with good complexity.

Our work shown here will promote the future research on the KECCAK from at least three aspects: First, the newly explored propagation of rotational relations can be easily extended to a good distinguisher for the KECCAK-f, which is not developed in detail in this work given the length of the paper. Second, this work essentially establishes an important connection between the linearization of KECCAK-f and the rotational cryptanalysis for the first time. A better linearization of KECCAK-f in the future will make it possible to keep enough eigenpoints even at the end of 4.5 rounds KECCAK-f, which will improve the preimage attack against KECCAK-224 up to 5 rounds. Finally, the introduction of the Harrow-Hassidim-Lloyd(HHL) quantum algorithm in the quantum counterpart of the algebraic attack, which provides an exponential speed-up for solving linear systems, will make it interesting to further improve the preimage recovery against the KECCAK variants in the field of quantum cryptanalysis.

Acknowledgement

This work is supported in part by the Key Research and Development Program of Shaanxi (No. 2021ZDLGY06-04), Guangxi Key Laboratory of Cryptography and Information Security (No. GCIS201802).

Appendix A

Proof of Theorem 3.1.

To illustrate the candidate messaged state A^2 and the corresponding rotational counterparts $\overleftarrow{A}^2$ share the same maps, here we only need to prove that the rotational counterparts $\overleftarrow{A}^2$ still keep two sets of constraints 3 and 6 mentioned in section 3.2. For a given candidate messaged state A^2, we have:

$$A^2_{x,4,z} \oplus S_{x-1,z} \oplus S_{x+1,z-1} = 1,$$

where $0 \leq x < 5$ and $0 \leq z < 64$. Consider the rotational counterpart $\overleftarrow{A}^2$ of this element A^2 with the rotational number n, we have:

$$\overleftarrow{A}^2_{x,4,z} \oplus \overleftarrow{S}_{x-1,z} \oplus \overleftarrow{S}_{x+1,z-1}$$

$$= A^2_{x,4,z-n} \oplus S_{x-1,z-n} \oplus S_{x+1,z-1-n} = 1,$$

where $0 \leq x < 5$ and $0 \leq z < 64$, which is exactly the same as the previous set of constraints 3. For the second set of constraints 6, we have the state I^1 corresponded to the element A^2 keeps the following constraints:

$$\bigoplus_{j=0}^{4} I^1_{0,j,z} = C_{0,z}; \quad \bigoplus_{j=0}^{4} I^1_{2,j,z} = C_{2,z},$$

where $0 \leq z < 64$. For the rotational counterpart $\overleftarrow{A}^2$ that meets the constraints 6, we have:

$$\bigoplus_{j=0}^{4} \overleftarrow{I}^1_{0,j,z} = \bigoplus_{j=0}^{4} I^1_{0,j,z-n} = C_{0,z};$$

$$\bigoplus_{j=0}^{4} \overleftarrow{I}^1_{2,j,z} = \bigoplus_{j=0}^{4} I^1_{2,j,z-n} = C_{2,z};$$

where $0 \leq z < 64$. Owing to each rotational counterpart of the constant array C used in the section 3.2 is still the constant array C, the rotational counterpart $\overleftarrow{A}^2$ obviously also meets the above constraints.

Proof of Theorem 3.2.

Recall that the XOR operation changes the rotational relations as: $P_{out} = P_a + P_b - 2P_aP_b$. Let the intputs a and b be 0 or 1, which means the rotational relations of the inputs are eigenpoints, there are only 4 possible combinations $(0,0),(0,1),(1,0)$ and $(1,1)$ for inputs pair (a,b). The corresponding outputs are listed below

$$\begin{aligned}
P_{out}[(a,b)=(0,0)] &= 0+0-2\times 0\times 0 = 0;\\
P_{out}[(a,b)=(0,1)] &= 0+1-2\times 0\times 1 = 1;\\
P_{out}[(a,b)=(1,0)] &= 1+0-2\times 1\times 0 = 1;\\
P_{out}[(a,b)=(1,1)] &= 1+1-2\times 1\times 1 = 0.
\end{aligned}$$

Therefore, the output of the XOR operation between the bits corresponding to the eigenpoints is still an eigenpoint. Fot the AND operation, we have $P_{out} = \frac{1}{2}(P_a + P_b - P_aP_b)$, and do the same as above.

$$\begin{aligned}
P_{out}[(a,b)=(0,0)] &= \frac{1}{2}(0+0-0\times 0) = 0;\\
P_{out}[(a,b)=(0,1)] &= \frac{1}{2}(0+1-0\times 1) = \frac{1}{2};\\
P_{out}[(a,b)=(1,0)] &= \frac{1}{2}(1+0-1\times 0) = \frac{1}{2};\\
P_{out}[(a,b)=(1,1)] &= \frac{1}{2}(1+1-1\times 1) = \frac{1}{2}.
\end{aligned}$$

We find that the AND operation stinctively makes the rotational relations acted as eigenpoints tend to be independent$(P_{x,y,z}[A^2_{x,y,z} \neq \overleftarrow{A}^2_{x,y,z+n}] \to \frac{1}{2})$. For other bitwise operations and XORed-with-constant operations involved in KECCAK-f, the eigenpoints are obviously not be affected.

According to the definition of rotational counterpart, all the rotational relations between the state array A^2 and its rotational counterpart $\overleftarrow{A}^2$ at the beginning are essentially eigenpoints. Specifically,

$$\forall n \in \{0,\cdots,63\}, P_{x,y,z}[A^2_{x,y,z} \neq \overleftarrow{A}^2_{x,y,z+n}] = 0,$$

where $0 \le x,y < 5$ and $0 \le z < 64$. On the one hand, it is noting that the diffusion of maps θ 4 and 7 are greatly reduced. That why there are fewer eigenpoints $P_{x,y,z} = 1$ after 2.5 rounds Rnd, which will greatly improve the retention of eigenpoints in later Rnd. On the other hand, the special maps χ mentioned in section 3.2 does not involve the operation AND, which further leads to the fact that the first 2.5 rounds do not involve the AND operation. Therefore, all the eigenpoints can pass through the 2.5 rounds Rnd without any reduction.

Proof of Theorem 4.1.

Suppose $\overleftarrow{A}^2$ is a rotational counterpart of $\overleftarrow{H}$'s preimage A^2, and the corresponding rotational number is $n = i$. According to the Figure 10(a), we have obtained 32 eigenpoints from the first 3.5 lanes. Therefore, for a random rotational number n excluded the right one i will be marked with probability of $2^{-32} \to 0$, and the right one i will be marked with probability of 1. That means for the right $\overleftarrow{H}$, there are $1 + 63 \times 2^{-32} \to 1$ rotational number candidate can be marked in every iteration, which further means that only the right rotational number can be marked. After 2^{58} iterations, we can find out the correct rotational number for the right rotational counterpart $\overleftarrow{A}^2$ with probability $p = 1$. Suppose $\overleftarrow{A}^2$ is not a rotational counterpart of $\overleftarrow{H}$'s preimage A^2 and therefore there is not a meaningful rotational number n for the random value $\overleftarrow{H}$. So, there are $64 \times 2^{-32} \to 0$ can be marked in any iteration. After 2^{58} iterations, the output $n < 64$ with probability $p = 2^{-112}$.

Proof of Theorem 4.2.

Let the state array $\overleftarrow{A}^2$ be one of the rotational counterparts corresponding to the second preimage $\overleftarrow{A}^2$. According to the definition of the oracle operator U_N mentioned in the section 4.1, we can use the amplitude amplification to find the rotational number is $n = i$ of rotational counterpart $\overleftarrow{A}^2$ in the step 2. Owing to a valid rotational number n is less than 64, in the step 3 the first flag $flag_1$ will be set to 1 with probability of 100%. After rotating back the rotational counterpart $\overleftarrow{A}^2$ by $n = i$ bits, we are expected to obtain the second preimage A^2. Therefore, all the 112 equations describing the relations between the bits of second preimage A^2 and any 112 bits of hash value will pass the verification in the step 4, and the second flag $flag_2$ will also be set to 1 with probability of 100%. By using the phase kickback technique, the oracle operator U_F can certainly flip the phase of the rotational counterpart $\overleftarrow{A}^2$ of second preimage A^2.

In most cases, input state array $\overleftarrow{A}^2$ is a random string that has nothing to do with the second preimage A^2. Therefore, the value n we obtained in the step 3 is a random value uniformly distributed from 0 to $2^{112} - 1$, and the first flag $flag_1$ will be set to 1 with probability of $64/2^{112} = 2^{-104}$. Later in the step 4, the obtained value A^2 from rotating back the random string $\overleftarrow{A}^2$ by n bits still has nothing to do with the second preimage A^2, and passes through the verification of 112 equations with probability of 2^{-112}. This means that the probability of oracle operator U_F flipping the phase of random string $\overleftarrow{A}^2$ is only $2^{-112} \times 2^{-104} = 2^{-216} \to 0$.

What is worth mentioning is that the method of using 112 equations to verify the correctness of second preimage A^2, both the scale of quantum circuit implementation and the consumption of calculation are far less than the method of calling 4-round modified Keccak-224. Therefore, we only use

the running of 4-round modified KECCAK-224 twice(include the uncomputing counterpart) throughout the entire process, which is better than that of Wang et al. in [26].

Appendix B

$$A_{0,0,0} \oplus A_{1,0,63} \oplus A_{2,0,2} \oplus A_{3,0,36} \oplus A_{4,0,37} \oplus A_{0,4,0} \oplus A_{1,4,63} \oplus A_{2,4,2} \oplus A_{3,4,36} \oplus A_{4,4,37} = C_{0,0} \oplus C_{2,0};$$

$$A_{0,2,61} \oplus A_{1,2,54} \oplus A_{2,2,21} \oplus A_{3,2,39} \oplus A_{4,2,25} \oplus A_{0,4,61} \oplus A_{1,4,54} \oplus A_{2,4,21} \oplus A_{3,4,39} \oplus A_{4,4,25} = C_{2,0}$$

$$A_{0,0,1} \oplus A_{1,0,0} \oplus A_{2,0,3} \oplus A_{3,0,37} \oplus A_{4,0,38} \oplus A_{0,4,1} \oplus A_{1,4,0} \oplus A_{2,4,3} \oplus A_{3,4,37} \oplus A_{4,4,38} = C_{0,1} \oplus C_{2,1};$$

$$A_{0,2,62} \oplus A_{1,2,55} \oplus A_{2,2,22} \oplus A_{3,2,40} \oplus A_{4,2,26} \oplus A_{0,4,62} \oplus A_{1,4,55} \oplus A_{2,4,22} \oplus A_{3,4,40} \oplus A_{4,4,26} = C_{2,1}$$

$$A_{0,0,2} \oplus A_{1,0,1} \oplus A_{2,0,4} \oplus A_{3,0,38} \oplus A_{4,0,39} \oplus A_{0,4,2} \oplus A_{1,4,1} \oplus A_{2,4,4} \oplus A_{3,4,38} \oplus A_{4,4,39} = C_{0,2} \oplus C_{2,2};$$

$$A_{0,2,63} \oplus A_{1,2,56} \oplus A_{2,2,23} \oplus A_{3,2,41} \oplus A_{4,2,27} \oplus A_{0,4,63} \oplus A_{1,4,56} \oplus A_{2,4,23} \oplus A_{3,4,41} \oplus A_{4,4,27} = C_{2,2}$$

$$A_{0,0,3} \oplus A_{1,0,2} \oplus A_{2,0,5} \oplus A_{3,0,39} \oplus A_{4,0,40} \oplus A_{0,4,3} \oplus A_{1,4,2} \oplus A_{2,4,5} \oplus A_{3,4,39} \oplus A_{4,4,40} = C_{0,3} \oplus C_{2,3};$$

$$A_{0,2,0} \oplus A_{1,2,57} \oplus A_{2,2,24} \oplus A_{3,2,42} \oplus A_{4,2,28} \oplus A_{0,4,0} \oplus A_{1,4,57} \oplus A_{2,4,24} \oplus A_{3,4,42} \oplus A_{4,4,28} = C_{2,3}$$

$$A_{0,0,4} \oplus A_{1,0,3} \oplus A_{2,0,6} \oplus A_{3,0,40} \oplus A_{4,0,41} \oplus A_{0,4,4} \oplus A_{1,4,3} \oplus A_{2,4,6} \oplus A_{3,4,40} \oplus A_{4,4,41} = C_{0,4} \oplus C_{2,4};$$

$$A_{0,2,1} \oplus A_{1,2,58} \oplus A_{2,2,25} \oplus A_{3,2,43} \oplus A_{4,2,29} \oplus A_{0,4,1} \oplus A_{1,4,58} \oplus A_{2,4,25} \oplus A_{3,4,43} \oplus A_{4,4,29} = C_{2,4}$$

$$A_{0,0,5} \oplus A_{1,0,4} \oplus A_{2,0,7} \oplus A_{3,0,41} \oplus A_{4,0,42} \oplus A_{0,4,5} \oplus A_{1,4,4} \oplus A_{2,4,7} \oplus A_{3,4,41} \oplus A_{4,4,42} = C_{0,5} \oplus C_{2,5};$$

$$A_{0,2,2} \oplus A_{1,2,59} \oplus A_{2,2,26} \oplus A_{3,2,44} \oplus A_{4,2,30} \oplus A_{0,4,2} \oplus A_{1,4,59} \oplus A_{2,4,26} \oplus A_{3,4,44} \oplus A_{4,4,30} = C_{2,5}$$

$$A_{0,0,6} \oplus A_{1,0,5} \oplus A_{2,0,8} \oplus A_{3,0,42} \oplus A_{4,0,43} \oplus A_{0,4,6} \oplus A_{1,4,5} \oplus A_{2,4,8} \oplus A_{3,4,42} \oplus A_{4,4,43} = C_{0,6} \oplus C_{2,6};$$

$$A_{0,2,3} \oplus A_{1,2,60} \oplus A_{2,2,27} \oplus A_{3,2,45} \oplus A_{4,2,31} \oplus A_{0,4,3} \oplus A_{1,4,60} \oplus A_{2,4,27} \oplus A_{3,4,45} \oplus A_{4,4,31} = C_{2,6}$$

$$A_{0,0,7} \oplus A_{1,0,6} \oplus A_{2,0,9} \oplus A_{3,0,43} \oplus A_{4,0,44} \oplus A_{0,4,7} \oplus A_{1,4,6} \oplus A_{2,4,9} \oplus A_{3,4,43} \oplus A_{4,4,44} = C_{0,7} \oplus C_{2,7};$$

$$A_{0,2,4} \oplus A_{1,2,61} \oplus A_{2,2,28} \oplus A_{3,2,46} \oplus A_{4,2,32} \oplus A_{0,4,4} \oplus A_{1,4,61} \oplus A_{2,4,28} \oplus A_{3,4,46} \oplus A_{4,4,32} = C_{2,7}$$

$$A_{0,0,8} \oplus A_{1,0,7} \oplus A_{2,0,10} \oplus A_{3,0,44} \oplus A_{4,0,45} \oplus A_{0,4,8} \oplus A_{1,4,7} \oplus A_{2,4,10} \oplus A_{3,4,44} \oplus A_{4,4,45} = C_{0,8} \oplus C_{2,8};$$

$$A_{0,2,5} \oplus A_{1,2,62} \oplus A_{2,2,29} \oplus A_{3,2,47} \oplus A_{4,2,33} \oplus A_{0,4,5} \oplus A_{1,4,62} \oplus A_{2,4,29} \oplus A_{3,4,47} \oplus A_{4,4,33} = C_{2,8}$$

$$A_{0,0,9} \oplus A_{1,0,8} \oplus A_{2,0,11} \oplus A_{3,0,45} \oplus A_{4,0,46} \oplus A_{0,4,9} \oplus A_{1,4,8} \oplus A_{2,4,11} \oplus A_{3,4,45} \oplus A_{4,4,46} = C_{0,9} \oplus C_{2,9};$$

$$A_{0,2,6} \oplus A_{1,2,63} \oplus A_{2,2,30} \oplus A_{3,2,48} \oplus A_{4,2,34} \oplus A_{0,4,6} \oplus A_{1,4,63} \oplus A_{2,4,30} \oplus A_{3,4,48} \oplus A_{4,4,34} = C_{2,9}$$

$$A_{0,0,10} \oplus A_{1,0,9} \oplus A_{2,0,12} \oplus A_{3,0,46} \oplus A_{4,0,47} \oplus A_{0,4,10} \oplus A_{1,4,9} \oplus A_{2,4,12} \oplus A_{3,4,46} \oplus A_{4,4,47} = C_{0,10} \oplus C_{2,10};$$

$$A_{0,2,7} \oplus A_{1,2,0} \oplus A_{2,2,31} \oplus A_{3,2,49} \oplus A_{4,2,35} \oplus A_{0,4,7} \oplus A_{1,4,0} \oplus A_{2,4,31} \oplus A_{3,4,49} \oplus A_{4,4,35} = C_{2,10}$$

$$A_{0,0,11} \oplus A_{1,0,10} \oplus A_{2,0,13} \oplus A_{3,0,47} \oplus A_{4,0,48} \oplus A_{0,4,11} \oplus A_{1,4,10} \oplus A_{2,4,13} \oplus A_{3,4,47} \oplus A_{4,4,48} = C_{0,11} \oplus C_{2,11};$$

$$A_{0,2,8} \oplus A_{1,2,1} \oplus A_{2,2,32} \oplus A_{3,2,50} \oplus A_{4,2,36} \oplus A_{0,4,8} \oplus A_{1,4,1} \oplus A_{2,4,32} \oplus A_{3,4,50} \oplus A_{4,4,36} = C_{2,11}$$

$$A_{0,0,12} \oplus A_{1,0,11} \oplus A_{2,0,14} \oplus A_{3,0,48} \oplus A_{4,0,49} \oplus A_{0,4,12} \oplus A_{1,4,11} \oplus A_{2,4,14} \oplus A_{3,4,48} \oplus A_{4,4,49} = C_{0,12} \oplus C_{2,12};$$

$$A_{0,2,9} \oplus A_{1,2,2} \oplus A_{2,2,33} \oplus A_{3,2,51} \oplus A_{4,2,37} \oplus A_{0,4,9} \oplus A_{1,4,2} \oplus A_{2,4,33} \oplus A_{3,4,51} \oplus A_{4,4,37} = C_{2,12}$$

$$A_{0,0,13} \oplus A_{1,0,12} \oplus A_{2,0,15} \oplus A_{3,0,49} \oplus A_{4,0,50} \oplus A_{0,4,13} \oplus A_{1,4,12} \oplus A_{2,4,15} \oplus A_{3,4,49} \oplus A_{4,4,50} = C_{0,13} \oplus C_{2,13};$$

$$A_{0,2,10} \oplus A_{1,2,3} \oplus A_{2,2,34} \oplus A_{3,2,52} \oplus A_{4,2,38} \oplus A_{0,4,10} \oplus A_{1,4,3} \oplus A_{2,4,34} \oplus A_{3,4,52} \oplus A_{4,4,38} = C_{2,13}$$

$$A_{0,0,14} \oplus A_{1,0,13} \oplus A_{2,0,16} \oplus A_{3,0,50} \oplus A_{4,0,51} \oplus A_{0,4,14} \oplus A_{1,4,13} \oplus A_{2,4,16} \oplus A_{3,4,50} \oplus A_{4,4,51} = C_{0,14} \oplus C_{2,14};$$

$$A_{0,2,11} \oplus A_{1,2,4} \oplus A_{2,2,35} \oplus A_{3,2,53} \oplus A_{4,2,39} \oplus A_{0,4,11} \oplus A_{1,4,4} \oplus A_{2,4,35} \oplus A_{3,4,53} \oplus A_{4,4,39} = C_{2,14}$$

$$A_{0,0,15} \oplus A_{1,0,14} \oplus A_{2,0,17} \oplus A_{3,0,51} \oplus A_{4,0,52} \oplus A_{0,4,15} \oplus A_{1,4,14} \oplus A_{2,4,17} \oplus A_{3,4,51} \oplus A_{4,4,52} = C_{0,15} \oplus C_{2,15};$$

$$A_{0,2,12} \oplus A_{1,2,5} \oplus A_{2,2,36} \oplus A_{3,2,54} \oplus A_{4,2,40} \oplus A_{0,4,12} \oplus A_{1,4,5} \oplus A_{2,4,36} \oplus A_{3,4,54} \oplus A_{4,4,40} = C_{2,15}$$

$$A_{0,0,16} \oplus A_{1,0,15} \oplus A_{2,0,18} \oplus A_{3,0,52} \oplus A_{4,0,53} \oplus A_{0,4,16} \oplus A_{1,4,15} \oplus A_{2,4,18} \oplus A_{3,4,52} \oplus A_{4,4,53} = C_{0,16} \oplus C_{2,16};$$

$$A_{0,2,13} \oplus A_{1,2,6} \oplus A_{2,2,37} \oplus A_{3,2,55} \oplus A_{4,2,41} \oplus A_{0,4,13} \oplus A_{1,4,6} \oplus A_{2,4,37} \oplus A_{3,4,55} \oplus A_{4,4,41} = C_{2,16}$$

$$A_{0,0,17} \oplus A_{1,0,16} \oplus A_{2,0,19} \oplus A_{3,0,53} \oplus A_{4,0,54} \oplus A_{0,4,17} \oplus A_{1,4,16} \oplus A_{2,4,19} \oplus A_{3,4,53} \oplus A_{4,4,54} = C_{0,17} \oplus C_{2,17};$$

$$A_{0,2,14} \oplus A_{1,2,7} \oplus A_{2,2,38} \oplus A_{3,2,56} \oplus A_{4,2,42} \oplus A_{0,4,14} \oplus A_{1,4,7} \oplus A_{2,4,38} \oplus A_{3,4,56} \oplus A_{4,4,42} = C_{2,17}$$

$$A_{0,0,18} \oplus A_{1,0,17} \oplus A_{2,0,20} \oplus A_{3,0,54} \oplus A_{4,0,55} \oplus A_{0,4,18} \oplus A_{1,4,17} \oplus A_{2,4,20} \oplus A_{3,4,54} \oplus A_{4,4,55} = C_{0,18} \oplus C_{2,18};$$

$$A_{0,2,15} \oplus A_{1,2,8} \oplus A_{2,2,39} \oplus A_{3,2,57} \oplus A_{4,2,43} \oplus A_{0,4,15} \oplus A_{1,4,8} \oplus A_{2,4,39} \oplus A_{3,4,57} \oplus A_{4,4,43} = C_{2,18}$$

$$A_{0,0,19} \oplus A_{1,0,18} \oplus A_{2,0,21} \oplus A_{3,0,55} \oplus A_{4,0,56} \oplus A_{0,4,19} \oplus A_{1,4,18} \oplus A_{2,4,21} \oplus A_{3,4,55} \oplus A_{4,4,56} = C_{0,19} \oplus C_{2,19};$$

$$A_{0,2,16} \oplus A_{1,2,9} \oplus A_{2,2,40} \oplus A_{3,2,58} \oplus A_{4,2,44} \oplus A_{0,4,16} \oplus A_{1,4,9} \oplus A_{2,4,40} \oplus A_{3,4,58} \oplus A_{4,4,44} = C_{2,19}$$

$$A_{0,0,20} \oplus A_{1,0,19} \oplus A_{2,0,22} \oplus A_{3,0,56} \oplus A_{4,0,57} \oplus A_{0,4,20} \oplus A_{1,4,19} \oplus A_{2,4,22} \oplus A_{3,4,56} \oplus A_{4,4,57} = C_{0,20} \oplus C_{2,20};$$

$$A_{0,2,17} \oplus A_{1,2,10} \oplus A_{2,2,41} \oplus A_{3,2,59} \oplus A_{4,2,45} \oplus A_{0,4,17} \oplus A_{1,4,10} \oplus A_{2,4,41} \oplus A_{3,4,59} \oplus A_{4,4,45} = C_{2,20}$$

$$A_{0,0,21} \oplus A_{1,0,20} \oplus A_{2,0,23} \oplus A_{3,0,57} \oplus A_{4,0,58} \oplus A_{0,4,21} \oplus A_{1,4,20} \oplus A_{2,4,23} \oplus A_{3,4,57} \oplus A_{4,4,58} = C_{0,21} \oplus C_{2,21};$$

$$A_{0,2,18} \oplus A_{1,2,11} \oplus A_{2,2,42} \oplus A_{3,2,60} \oplus A_{4,2,46} \oplus A_{0,4,18} \oplus A_{1,4,11} \oplus A_{2,4,42} \oplus A_{3,4,60} \oplus A_{4,4,46} = C_{2,21}$$

$$A_{0,0,22} \oplus A_{1,0,21} \oplus A_{2,0,24} \oplus A_{3,0,58} \oplus A_{4,0,59} \oplus A_{0,4,22} \oplus A_{1,4,21} \oplus A_{2,4,24} \oplus A_{3,4,58} \oplus A_{4,4,59} = C_{0,22} \oplus C_{2,22};$$

$$A_{0,2,19} \oplus A_{1,2,12} \oplus A_{2,2,43} \oplus A_{3,2,61} \oplus A_{4,2,47} \oplus A_{0,4,19} \oplus A_{1,4,12} \oplus A_{2,4,43} \oplus A_{3,4,61} \oplus A_{4,4,47} = C_{2,22}$$

$$A_{0,0,23} \oplus A_{1,0,22} \oplus A_{2,0,25} \oplus A_{3,0,59} \oplus A_{4,0,60} \oplus A_{0,4,23} \oplus A_{1,4,22} \oplus A_{2,4,25} \oplus A_{3,4,59} \oplus A_{4,4,60} = C_{0,23} \oplus C_{2,23};$$

$$A_{0,2,20} \oplus A_{1,2,13} \oplus A_{2,2,44} \oplus A_{3,2,62} \oplus A_{4,2,48} \oplus A_{0,4,20} \oplus A_{1,4,13} \oplus A_{2,4,44} \oplus A_{3,4,62} \oplus A_{4,4,48} = C_{2,23}$$

$$A_{0,0,24} \oplus A_{1,0,23} \oplus A_{2,0,26} \oplus A_{3,0,60} \oplus A_{4,0,61} \oplus A_{0,4,24} \oplus A_{1,4,23} \oplus A_{2,4,26} \oplus A_{3,4,60} \oplus A_{4,4,61} = C_{0,24} \oplus C_{2,24};$$

$$A_{0,2,21} \oplus A_{1,2,14} \oplus A_{2,2,45} \oplus A_{3,2,63} \oplus A_{4,2,49} \oplus A_{0,4,21} \oplus A_{1,4,14} \oplus A_{2,4,45} \oplus A_{3,4,63} \oplus A_{4,4,49} = C_{2,24}$$

$$A_{0,0,25} \oplus A_{1,0,24} \oplus A_{2,0,27} \oplus A_{3,0,61} \oplus A_{4,0,62} \oplus A_{0,4,25} \oplus A_{1,4,24} \oplus A_{2,4,27} \oplus A_{3,4,61} \oplus A_{4,4,62} = C_{0,25} \oplus C_{2,25};$$

$$A_{0,2,22} \oplus A_{1,2,15} \oplus A_{2,2,46} \oplus A_{3,2,0} \oplus A_{4,2,50} \oplus A_{0,4,22} \oplus A_{1,4,15} \oplus A_{2,4,46} \oplus A_{3,4,0} \oplus A_{4,4,50} = C_{2,25}$$

$$A_{0,0,26} \oplus A_{1,0,25} \oplus A_{2,0,28} \oplus A_{3,0,62} \oplus A_{4,0,63} \oplus A_{0,4,26} \oplus A_{1,4,25} \oplus A_{2,4,28} \oplus A_{3,4,62} \oplus A_{4,4,63} = C_{0,26} \oplus C_{2,26};$$

$$A_{0,2,23} \oplus A_{1,2,16} \oplus A_{2,2,47} \oplus A_{3,2,1} \oplus A_{4,2,51} \oplus A_{0,4,23} \oplus A_{1,4,16} \oplus A_{2,4,47} \oplus A_{3,4,1} \oplus A_{4,4,51} = C_{2,26}$$

$$A_{0,0,27} \oplus A_{1,0,26} \oplus A_{2,0,29} \oplus A_{3,0,63} \oplus A_{4,0,0} \oplus A_{0,4,27} \oplus A_{1,4,26} \oplus A_{2,4,29} \oplus A_{3,4,63} \oplus A_{4,4,0} = C_{0,27} \oplus C_{2,27};$$

$$A_{0,2,24} \oplus A_{1,2,17} \oplus A_{2,2,48} \oplus A_{3,2,2} \oplus A_{4,2,52} \oplus A_{0,4,24} \oplus A_{1,4,17} \oplus A_{2,4,48} \oplus A_{3,4,2} \oplus A_{4,4,52} = C_{2,27}$$

$$A_{0,0,28} \oplus A_{1,0,27} \oplus A_{2,0,30} \oplus A_{3,0,0} \oplus A_{4,0,1} \oplus A_{0,4,28} \oplus A_{1,4,27} \oplus A_{2,4,30} \oplus A_{3,4,0} \oplus A_{4,4,1} = C_{0,28} \oplus C_{2,28};$$

$$A_{0,2,25} \oplus A_{1,2,18} \oplus A_{2,2,49} \oplus A_{3,2,3} \oplus A_{4,2,53} \oplus A_{0,4,25} \oplus A_{1,4,18} \oplus A_{2,4,49} \oplus A_{3,4,3} \oplus A_{4,4,53} = C_{2,28}$$

$$A_{0,0,29} \oplus A_{1,0,28} \oplus A_{2,0,31} \oplus A_{3,0,1} \oplus A_{4,0,2} \oplus A_{0,4,29} \oplus A_{1,4,28} \oplus A_{2,4,31} \oplus A_{3,4,1} \oplus A_{4,4,2} = C_{0,29} \oplus C_{2,29};$$

$$A_{0,2,26} \oplus A_{1,2,19} \oplus A_{2,2,50} \oplus A_{3,2,4} \oplus A_{4,2,54} \oplus A_{0,4,26} \oplus A_{1,4,19} \oplus A_{2,4,50} \oplus A_{3,4,4} \oplus A_{4,4,54} = C_{2,29}$$

$$A_{0,0,30} \oplus A_{1,0,29} \oplus A_{2,0,32} \oplus A_{3,0,2} \oplus A_{4,0,3} \oplus A_{0,4,30} \oplus A_{1,4,29} \oplus A_{2,4,32} \oplus A_{3,4,2} \oplus A_{4,4,3} = C_{0,30} \oplus C_{2,30};$$

$$A_{0,2,27} \oplus A_{1,2,20} \oplus A_{2,2,51} \oplus A_{3,2,5} \oplus A_{4,2,55} \oplus A_{0,4,27} \oplus A_{1,4,20} \oplus A_{2,4,51} \oplus A_{3,4,5} \oplus A_{4,4,55} = C_{2,30}$$

$$A_{0,0,31} \oplus A_{1,0,30} \oplus A_{2,0,33} \oplus A_{3,0,3} \oplus A_{4,0,4} \oplus A_{0,4,31} \oplus A_{1,4,30} \oplus A_{2,4,33} \oplus A_{3,4,3} \oplus A_{4,4,4} = C_{0,31} \oplus C_{2,31};$$

$$A_{0,2,28} \oplus A_{1,2,21} \oplus A_{2,2,52} \oplus A_{3,2,6} \oplus A_{4,2,56} \oplus A_{0,4,28} \oplus A_{1,4,21} \oplus A_{2,4,52} \oplus A_{3,4,6} \oplus A_{4,4,56} = C_{2,31}$$

$$A_{0,0,32} \oplus A_{1,0,31} \oplus A_{2,0,34} \oplus A_{3,0,4} \oplus A_{4,0,5} \oplus A_{0,4,32} \oplus A_{1,4,31} \oplus A_{2,4,34} \oplus A_{3,4,4} \oplus A_{4,4,5} = C_{0,32} \oplus C_{2,32};$$

$$A_{0,2,29} \oplus A_{1,2,22} \oplus A_{2,2,53} \oplus A_{3,2,7} \oplus A_{4,2,57} \oplus A_{0,4,29} \oplus A_{1,4,22} \oplus A_{2,4,53} \oplus A_{3,4,7} \oplus A_{4,4,57} = C_{2,32}$$

$$A_{0,0,33} \oplus A_{1,0,32} \oplus A_{2,0,35} \oplus A_{3,0,5} \oplus A_{4,0,6} \oplus A_{0,4,33} \oplus A_{1,4,32} \oplus A_{2,4,35} \oplus A_{3,4,5} \oplus A_{4,4,6} = C_{0,33} \oplus C_{2,33};$$

$$A_{0,2,30} \oplus A_{1,2,23} \oplus A_{2,2,54} \oplus A_{3,2,8} \oplus A_{4,2,58} \oplus A_{0,4,30} \oplus A_{1,4,23} \oplus A_{2,4,54} \oplus A_{3,4,8} \oplus A_{4,4,58} = C_{2,33}$$

$$A_{0,0,34} \oplus A_{1,0,33} \oplus A_{2,0,36} \oplus A_{3,0,6} \oplus A_{4,0,7} \oplus A_{0,4,34} \oplus A_{1,4,33} \oplus A_{2,4,36} \oplus A_{3,4,6} \oplus A_{4,4,7} = C_{0,34} \oplus C_{2,34};$$

$$A_{0,2,31} \oplus A_{1,2,24} \oplus A_{2,2,55} \oplus A_{3,2,9} \oplus A_{4,2,59} \oplus A_{0,4,31} \oplus A_{1,4,24} \oplus A_{2,4,55} \oplus A_{3,4,9} \oplus A_{4,4,59} = C_{2,34}$$

$$A_{0,0,35} \oplus A_{1,0,34} \oplus A_{2,0,37} \oplus A_{3,0,7} \oplus A_{4,0,8} \oplus A_{0,4,35} \oplus A_{1,4,34} \oplus A_{2,4,37} \oplus A_{3,4,7} \oplus A_{4,4,8} = C_{0,35} \oplus C_{2,35};$$

$$A_{0,2,32} \oplus A_{1,2,25} \oplus A_{2,2,56} \oplus A_{3,2,10} \oplus A_{4,2,60} \oplus A_{0,4,32} \oplus A_{1,4,25} \oplus A_{2,4,56} \oplus A_{3,4,10} \oplus A_{4,4,60} = C_{2,35}$$

$$A_{0,0,36} \oplus A_{1,0,35} \oplus A_{2,0,38} \oplus A_{3,0,8} \oplus A_{4,0,9} \oplus A_{0,4,36} \oplus A_{1,4,35} \oplus A_{2,4,38} \oplus A_{3,4,8} \oplus A_{4,4,9} = C_{0,36} \oplus C_{2,36};$$

$$A_{0,2,33} \oplus A_{1,2,26} \oplus A_{2,2,57} \oplus A_{3,2,11} \oplus A_{4,2,61} \oplus A_{0,4,33} \oplus A_{1,4,26} \oplus A_{2,4,57} \oplus A_{3,4,11} \oplus A_{4,4,61} = C_{2,36}$$

$$A_{0,0,37} \oplus A_{1,0,36} \oplus A_{2,0,39} \oplus A_{3,0,9} \oplus A_{4,0,10} \oplus A_{0,4,37} \oplus A_{1,4,36} \oplus A_{2,4,39} \oplus A_{3,4,9} \oplus A_{4,4,10} = C_{0,37} \oplus C_{2,37};$$

$$A_{0,2,34} \oplus A_{1,2,27} \oplus A_{2,2,58} \oplus A_{3,2,12} \oplus A_{4,2,62} \oplus A_{0,4,34} \oplus A_{1,4,27} \oplus A_{2,4,58} \oplus A_{3,4,12} \oplus A_{4,4,62} = C_{2,37}$$

$$A_{0,0,38} \oplus A_{1,0,37} \oplus A_{2,0,40} \oplus A_{3,0,10} \oplus A_{4,0,11} \oplus A_{0,4,38} \oplus A_{1,4,37} \oplus A_{2,4,40} \oplus A_{3,4,10} \oplus A_{4,4,11} = C_{0,38} \oplus C_{2,38};$$

$$A_{0,2,35} \oplus A_{1,2,28} \oplus A_{2,2,59} \oplus A_{3,2,13} \oplus A_{4,2,63} \oplus A_{0,4,35} \oplus A_{1,4,28} \oplus A_{2,4,59} \oplus A_{3,4,13} \oplus A_{4,4,63} = C_{2,38}$$

$$A_{0,0,39} \oplus A_{1,0,38} \oplus A_{2,0,41} \oplus A_{3,0,11} \oplus A_{4,0,12} \oplus A_{0,4,39} \oplus A_{1,4,38} \oplus A_{2,4,41} \oplus A_{3,4,11} \oplus A_{4,4,12} = C_{0,39} \oplus C_{2,39};$$

$$A_{0,2,36} \oplus A_{1,2,29} \oplus A_{2,2,60} \oplus A_{3,2,14} \oplus A_{4,2,0} \oplus A_{0,4,36} \oplus A_{1,4,29} \oplus A_{2,4,60} \oplus A_{3,4,14} \oplus A_{4,4,0} = C_{2,39}$$

$$A_{0,0,40} \oplus A_{1,0,39} \oplus A_{2,0,42} \oplus A_{3,0,12} \oplus A_{4,0,13} \oplus A_{0,4,40} \oplus A_{1,4,39} \oplus A_{2,4,42} \oplus A_{3,4,12} \oplus A_{4,4,13} = C_{0,40} \oplus C_{2,40};$$

$$A_{0,2,37} \oplus A_{1,2,30} \oplus A_{2,2,61} \oplus A_{3,2,15} \oplus A_{4,2,1} \oplus A_{0,4,37} \oplus A_{1,4,30} \oplus A_{2,4,61} \oplus A_{3,4,15} \oplus A_{4,4,1} = C_{2,40}$$

$$A_{0,0,41} \oplus A_{1,0,40} \oplus A_{2,0,43} \oplus A_{3,0,13} \oplus A_{4,0,14} \oplus A_{0,4,41} \oplus A_{1,4,40} \oplus A_{2,4,43} \oplus A_{3,4,13} \oplus A_{4,4,14} = C_{0,41} \oplus C_{2,41};$$

$$A_{0,2,38} \oplus A_{1,2,31} \oplus A_{2,2,62} \oplus A_{3,2,16} \oplus A_{4,2,2} \oplus A_{0,4,38} \oplus A_{1,4,31} \oplus A_{2,4,62} \oplus A_{3,4,16} \oplus A_{4,4,2} = C_{2,41}$$

$$A_{0,0,42} \oplus A_{1,0,41} \oplus A_{2,0,44} \oplus A_{3,0,14} \oplus A_{4,0,15} \oplus A_{0,4,42} \oplus A_{1,4,41} \oplus A_{2,4,44} \oplus A_{3,4,14} \oplus A_{4,4,15} = C_{0,42} \oplus C_{2,42};$$

$$A_{0,2,39} \oplus A_{1,2,32} \oplus A_{2,2,63} \oplus A_{3,2,17} \oplus A_{4,2,3} \oplus A_{0,4,39} \oplus A_{1,4,32} \oplus A_{2,4,63} \oplus A_{3,4,17} \oplus A_{4,4,3} = C_{2,42}$$

$$A_{0,0,43} \oplus A_{1,0,42} \oplus A_{2,0,45} \oplus A_{3,0,15} \oplus A_{4,0,16} \oplus A_{0,4,43} \oplus A_{1,4,42} \oplus A_{2,4,45} \oplus A_{3,4,15} \oplus A_{4,4,16} = C_{0,43} \oplus C_{2,43};$$

$$A_{0,2,40} \oplus A_{1,2,33} \oplus A_{2,2,0} \oplus A_{3,2,18} \oplus A_{4,2,4} \oplus A_{0,4,40} \oplus A_{1,4,33} \oplus A_{2,4,0} \oplus A_{3,4,18} \oplus A_{4,4,4} = C_{2,43}$$

$$A_{0,0,44} \oplus A_{1,0,43} \oplus A_{2,0,46} \oplus A_{3,0,16} \oplus A_{4,0,17} \oplus A_{0,4,44} \oplus A_{1,4,43} \oplus A_{2,4,46} \oplus A_{3,4,16} \oplus A_{4,4,17} = C_{0,44} \oplus C_{2,44};$$

$$A_{0,2,41} \oplus A_{1,2,34} \oplus A_{2,2,1} \oplus A_{3,2,19} \oplus A_{4,2,5} \oplus A_{0,4,41} \oplus A_{1,4,34} \oplus A_{2,4,1} \oplus A_{3,4,19} \oplus A_{4,4,5} = C_{2,44}$$

$$A_{0,0,45} \oplus A_{1,0,44} \oplus A_{2,0,47} \oplus A_{3,0,17} \oplus A_{4,0,18} \oplus A_{0,4,45} \oplus A_{1,4,44} \oplus A_{2,4,47} \oplus A_{3,4,17} \oplus A_{4,4,18} = C_{0,45} \oplus C_{2,45};$$

$$A_{0,2,42} \oplus A_{1,2,35} \oplus A_{2,2,2} \oplus A_{3,2,20} \oplus A_{4,2,6} \oplus A_{0,4,42} \oplus A_{1,4,35} \oplus A_{2,4,2} \oplus A_{3,4,20} \oplus A_{4,4,6} = C_{2,45}$$

$$A_{0,0,46} \oplus A_{1,0,45} \oplus A_{2,0,48} \oplus A_{3,0,18} \oplus A_{4,0,19} \oplus A_{0,4,46} \oplus A_{1,4,45} \oplus A_{2,4,48} \oplus A_{3,4,18} \oplus A_{4,4,19} = C_{0,46} \oplus C_{2,46};$$

$$A_{0,2,43} \oplus A_{1,2,36} \oplus A_{2,2,3} \oplus A_{3,2,21} \oplus A_{4,2,7} \oplus A_{0,4,43} \oplus A_{1,4,36} \oplus A_{2,4,3} \oplus A_{3,4,21} \oplus A_{4,4,7} = C_{2,46}$$

$$A_{0,0,47} \oplus A_{1,0,46} \oplus A_{2,0,49} \oplus A_{3,0,19} \oplus A_{4,0,20} \oplus A_{0,4,47} \oplus A_{1,4,46} \oplus A_{2,4,49} \oplus A_{3,4,19} \oplus A_{4,4,20} = C_{0,47} \oplus C_{2,47};$$

$$A_{0,2,44} \oplus A_{1,2,37} \oplus A_{2,2,4} \oplus A_{3,2,22} \oplus A_{4,2,8} \oplus A_{0,4,44} \oplus A_{1,4,37} \oplus A_{2,4,4} \oplus A_{3,4,22} \oplus A_{4,4,8} = C_{2,47}$$

$$A_{0,0,48} \oplus A_{1,0,47} \oplus A_{2,0,50} \oplus A_{3,0,20} \oplus A_{4,0,21} \oplus A_{0,4,48} \oplus A_{1,4,47} \oplus A_{2,4,50} \oplus A_{3,4,20} \oplus A_{4,4,21} = C_{0,48} \oplus C_{2,48};$$

$$A_{0,2,45} \oplus A_{1,2,38} \oplus A_{2,2,5} \oplus A_{3,2,23} \oplus A_{4,2,9} \oplus A_{0,4,45} \oplus A_{1,4,38} \oplus A_{2,4,5} \oplus A_{3,4,23} \oplus A_{4,4,9} = C_{2,48}$$

$$A_{0,0,49} \oplus A_{1,0,48} \oplus A_{2,0,51} \oplus A_{3,0,21} \oplus A_{4,0,22} \oplus A_{0,4,49} \oplus A_{1,4,48} \oplus A_{2,4,51} \oplus A_{3,4,21} \oplus A_{4,4,22} = C_{0,49} \oplus C_{2,49};$$

$$A_{0,2,46} \oplus A_{1,2,39} \oplus A_{2,2,6} \oplus A_{3,2,24} \oplus A_{4,2,10} \oplus A_{0,4,46} \oplus A_{1,4,39} \oplus A_{2,4,6} \oplus A_{3,4,24} \oplus A_{4,4,10} = C_{2,49}$$

$$A_{0,0,50} \oplus A_{1,0,49} \oplus A_{2,0,52} \oplus A_{3,0,22} \oplus A_{4,0,23} \oplus A_{0,4,50} \oplus A_{1,4,49} \oplus A_{2,4,52} \oplus A_{3,4,22} \oplus A_{4,4,23} = C_{0,50} \oplus C_{2,50};$$

$$A_{0,2,47} \oplus A_{1,2,40} \oplus A_{2,2,7} \oplus A_{3,2,25} \oplus A_{4,2,11} \oplus A_{0,4,47} \oplus A_{1,4,40} \oplus A_{2,4,7} \oplus A_{3,4,25} \oplus A_{4,4,11} = C_{2,50}$$

$$A_{0,0,51} \oplus A_{1,0,50} \oplus A_{2,0,53} \oplus A_{3,0,23} \oplus A_{4,0,24} \oplus A_{0,4,51} \oplus A_{1,4,50} \oplus A_{2,4,53} \oplus A_{3,4,23} \oplus A_{4,4,24} = C_{0,51} \oplus C_{2,51};$$

$$A_{0,2,48} \oplus A_{1,2,41} \oplus A_{2,2,8} \oplus A_{3,2,26} \oplus A_{4,2,12} \oplus A_{0,4,48} \oplus A_{1,4,41} \oplus A_{2,4,8} \oplus A_{3,4,26} \oplus A_{4,4,12} = C_{2,51}$$

$$A_{0,0,52} \oplus A_{1,0,51} \oplus A_{2,0,54} \oplus A_{3,0,24} \oplus A_{4,0,25} \oplus A_{0,4,52} \oplus A_{1,4,51} \oplus A_{2,4,54} \oplus A_{3,4,24} \oplus A_{4,4,25} = C_{0,52} \oplus C_{2,52};$$

$$A_{0,2,49} \oplus A_{1,2,42} \oplus A_{2,2,9} \oplus A_{3,2,27} \oplus A_{4,2,13} \oplus A_{0,4,49} \oplus A_{1,4,42} \oplus A_{2,4,9} \oplus A_{3,4,27} \oplus A_{4,4,13} = C_{2,52}$$

$$A_{0,0,53} \oplus A_{1,0,52} \oplus A_{2,0,55} \oplus A_{3,0,25} \oplus A_{4,0,26} \oplus A_{0,4,53} \oplus A_{1,4,52} \oplus A_{2,4,55} \oplus A_{3,4,25} \oplus A_{4,4,26} = C_{0,53} \oplus C_{2,53};$$

$$A_{0,2,50} \oplus A_{1,2,43} \oplus A_{2,2,10} \oplus A_{3,2,28} \oplus A_{4,2,14} \oplus A_{0,4,50} \oplus A_{1,4,43} \oplus A_{2,4,10} \oplus A_{3,4,28} \oplus A_{4,4,14} = C_{2,53}$$

$$A_{0,0,54} \oplus A_{1,0,53} \oplus A_{2,0,56} \oplus A_{3,0,26} \oplus A_{4,0,27} \oplus A_{0,4,54} \oplus A_{1,4,53} \oplus A_{2,4,56} \oplus A_{3,4,26} \oplus A_{4,4,27} = C_{0,54} \oplus C_{2,54};$$

$$A_{0,2,51} \oplus A_{1,2,44} \oplus A_{2,2,11} \oplus A_{3,2,29} \oplus A_{4,2,15} \oplus A_{0,4,51} \oplus A_{1,4,44} \oplus A_{2,4,11} \oplus A_{3,4,29} \oplus A_{4,4,15} = C_{2,54}$$

$$A_{0,0,55} \oplus A_{1,0,54} \oplus A_{2,0,57} \oplus A_{3,0,27} \oplus A_{4,0,28} \oplus A_{0,4,55} \oplus A_{1,4,54} \oplus A_{2,4,57} \oplus A_{3,4,27} \oplus A_{4,4,28} = C_{0,55} \oplus C_{2,55};$$

$$A_{0,2,52} \oplus A_{1,2,45} \oplus A_{2,2,12} \oplus A_{3,2,30} \oplus A_{4,2,16} \oplus A_{0,4,52} \oplus A_{1,4,45} \oplus A_{2,4,12} \oplus A_{3,4,30} \oplus A_{4,4,16} = C_{2,55}$$

$$A_{0,0,56} \oplus A_{1,0,55} \oplus A_{2,0,58} \oplus A_{3,0,28} \oplus A_{4,0,29} \oplus A_{0,4,56} \oplus A_{1,4,55} \oplus A_{2,4,58} \oplus A_{3,4,28} \oplus A_{4,4,29} = C_{0,56} \oplus C_{2,56};$$

$$A_{0,2,53} \oplus A_{1,2,46} \oplus A_{2,2,13} \oplus A_{3,2,31} \oplus A_{4,2,17} \oplus A_{0,4,53} \oplus A_{1,4,46} \oplus A_{2,4,13} \oplus A_{3,4,31} \oplus A_{4,4,17} = C_{2,56}$$

$$A_{0,0,57} \oplus A_{1,0,56} \oplus A_{2,0,59} \oplus A_{3,0,29} \oplus A_{4,0,30} \oplus A_{0,4,57} \oplus A_{1,4,56} \oplus A_{2,4,59} \oplus A_{3,4,29} \oplus A_{4,4,30} = C_{0,57} \oplus C_{2,57};$$

$$A_{0,2,54} \oplus A_{1,2,47} \oplus A_{2,2,14} \oplus A_{3,2,32} \oplus A_{4,2,18} \oplus A_{0,4,54} \oplus A_{1,4,47} \oplus A_{2,4,14} \oplus A_{3,4,32} \oplus A_{4,4,18} = C_{2,57}$$

$$A_{0,0,58} \oplus A_{1,0,57} \oplus A_{2,0,60} \oplus A_{3,0,30} \oplus A_{4,0,31} \oplus A_{0,4,58} \oplus A_{1,4,57} \oplus A_{2,4,60} \oplus A_{3,4,30} \oplus A_{4,4,31} = C_{0,58} \oplus C_{2,58};$$

$$A_{0,2,55} \oplus A_{1,2,48} \oplus A_{2,2,15} \oplus A_{3,2,33} \oplus A_{4,2,19} \oplus A_{0,4,55} \oplus A_{1,4,48} \oplus A_{2,4,15} \oplus A_{3,4,33} \oplus A_{4,4,19} = C_{2,58}$$

$$A_{0,0,59} \oplus A_{1,0,58} \oplus A_{2,0,61} \oplus A_{3,0,31} \oplus A_{4,0,32} \oplus A_{0,4,59} \oplus A_{1,4,58} \oplus A_{2,4,61} \oplus A_{3,4,31} \oplus A_{4,4,32} = C_{0,59} \oplus C_{2,59};$$

$$A_{0,2,56} \oplus A_{1,2,49} \oplus A_{2,2,16} \oplus A_{3,2,34} \oplus A_{4,2,20} \oplus A_{0,4,56} \oplus A_{1,4,49} \oplus A_{2,4,16} \oplus A_{3,4,34} \oplus A_{4,4,20} = C_{2,59}$$

$$A_{0,0,60} \oplus A_{1,0,59} \oplus A_{2,0,62} \oplus A_{3,0,32} \oplus A_{4,0,33} \oplus A_{0,4,60} \oplus A_{1,4,59} \oplus A_{2,4,62} \oplus A_{3,4,32} \oplus A_{4,4,33} = C_{0,60} \oplus C_{2,60};$$

$$A_{0,2,57} \oplus A_{1,2,50} \oplus A_{2,2,17} \oplus A_{3,2,35} \oplus A_{4,2,21} \oplus A_{0,4,57} \oplus A_{1,4,50} \oplus A_{2,4,17} \oplus A_{3,4,35} \oplus A_{4,4,21} = C_{2,60}$$

$$A_{0,0,61} \oplus A_{1,0,60} \oplus A_{2,0,63} \oplus A_{3,0,33} \oplus A_{4,0,34} \oplus A_{0,4,61} \oplus A_{1,4,60} \oplus A_{2,4,63} \oplus A_{3,4,33} \oplus A_{4,4,34} = C_{0,61} \oplus C_{2,61};$$

$$A_{0,2,58} \oplus A_{1,2,51} \oplus A_{2,2,18} \oplus A_{3,2,36} \oplus A_{4,2,22} \oplus A_{0,4,58} \oplus A_{1,4,51} \oplus A_{2,4,18} \oplus A_{3,4,36} \oplus A_{4,4,22} = C_{2,61}$$

$$A_{0,0,62} \oplus A_{1,0,61} \oplus A_{2,0,0} \oplus A_{3,0,34} \oplus A_{4,0,35} \oplus A_{0,4,62} \oplus A_{1,4,61} \oplus A_{2,4,0} \oplus A_{3,4,34} \oplus A_{4,4,35} = C_{0,62} \oplus C_{2,62};$$

$$A_{0,2,59} \oplus A_{1,2,52} \oplus A_{2,2,19} \oplus A_{3,2,37} \oplus A_{4,2,23} \oplus A_{0,4,59} \oplus A_{1,4,52} \oplus A_{2,4,19} \oplus A_{3,4,37} \oplus A_{4,4,23} = C_{2,62}$$

$$A_{0,0,63} \oplus A_{1,0,62} \oplus A_{2,0,1} \oplus A_{3,0,35} \oplus A_{4,0,36} \oplus A_{0,4,63} \oplus A_{1,4,62} \oplus A_{2,4,1} \oplus A_{3,4,35} \oplus A_{4,4,36} = C_{0,63} \oplus C_{2,63};$$

$$A_{0,2,60} \oplus A_{1,2,53} \oplus A_{2,2,20} \oplus A_{3,2,38} \oplus A_{4,2,24} \oplus A_{0,4,60} \oplus A_{1,4,53} \oplus A_{2,4,20} \oplus A_{3,4,38} \oplus A_{4,4,24} = C_{2,63}$$

Here, we list the set of new linear constraints that act on the candidate messaged state A^2.

References

[1] Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions md4 and ripemd. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 1–18 (2005). Springer

[2] Wang, X., Yu, H.: How to break md5 and other hash functions. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 19–35 (2005). Springer

[3] Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full sha-1. In: Annual International Cryptology Conference, pp. 17–36 (2005). Springer

[4] Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on sha-0. In: Annual International Cryptology Conference, pp. 1–16 (2005). Springer

[5] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponges. online] http://sponge. noekeon. org (2011)

[6] Sha, N.: competition, 2007-2012. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html. November 6, 2021

[7] Dworkin, M.: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD (2015). https://doi.org/10.6028/NIST.FIPS.202

[8] Bernstein, D.J.: Second preimages for 6 (7?(8??)) rounds of keccak. NIST mailing list (2010)

[9] Morawiecki, P., Pieprzyk, J., Srebrny, M.: Rotational cryptanalysis of round-reduced keccak. In: International Workshop on Fast Software Encryption, pp. 241–262 (2013). Springer

[10] Guo, J., Liu, M., Song, L.: Linear structures: Applications to cryptanalysis of round-reduced keccak. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 249–274 (2016). Springer

[11] Li, T., Sun, Y., Liao, M., Wang, D.: Preimage attacks on the round-reduced keccak with cross-linear structures. IACR Transactions on Symmetric Cryptology, 39–57 (2017)

[12] Li, T., Sun, Y.: Preimage attacks on round-reduced keccak-224/256 via an allocating approach. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 556–584 (2019). Springer

[13] Liu, F., Isobe, T., Meier, W., Yang, Z.: Algebraic attacks on round-reduced keccak/xoodoo. Cryptology ePrint Archive (2020)

[14] Dinur, I.: Cryptanalytic applications of the polynomial method for solving multivariate equation systems over gf (2). In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 374–403 (2021). Springer

[15] Lokshtanov, D., Paturi, R., Tamaki, S., Williams, R., Yu, H.: Beating brute force for systems of polynomial equations over finite fields. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2190–2202 (2017). SIAM

[16] Wei, C., Wu, C., Fu, X., Dong, X., He, K., Hong, J., Wang, X.: Preimage Attacks on 4-round Keccak by Solving Multivariate Quadratic Systems. Cryptology ePrint Archive, Report 2021/732. https://ia.cr/2021/732 (2021)

[17] Hosoyamada, A., Sasaki, Y.: Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In: Cryptographers' Track at the RSA Conference, pp. 198–218 (2018). Springer

[18] Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. arXiv preprint arXiv:1510.05836 (2015)

[19] Gagliardoni, T.: Quantum security of cryptographic primitives. arXiv preprint arXiv:1705.02417 (2017)

[20] Leander, G., May, A.: Grover meets simon–quantumly attacking the fx-construction. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 161–178 (2017). Springer

[21] Bonnetain, X., Hosoyamada, A., Naya-Plasencia, M., Sasaki, Y., Schrottenloher, A.: Quantum attacks without superposition queries: the offline simon's algorithm. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 552–583 (2019). Springer

[22] Hosoyamada, A., Sasaki, Y.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday

bound. Advances in Cryptology–EUROCRYPT 2020 **12106**, 249 (2020)

[23] Dong, X., Sun, S., Shi, D., Gao, F., Wang, X., Hu, L.: Quantum collision attacks on aes-like hashing with low quantum random access memories. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 727–757 (2020). Springer

[24] Hosoyamada, A., Sasaki, Y.: Quantum collision attacks on reduced sha-256 and sha-512. IACR Cryptol. ePrint Arch. **2021**, 292 (2021)

[25] Amy, M., Di Matteo, O., Gheorghiu, V., Mosca, M., Parent, A., Schanck, J.: Estimating the cost of generic quantum pre-image attacks on sha-2 and sha-3. In: International Conference on Selected Areas in Cryptography, pp. 317–337 (2016). Springer

[26] Wang, R., Li, X., Gao, J., Li, H., Wang, B.: Quantum rotational cryptanalysis for preimage recovery of round-reduced keccak. Cryptology ePrint Archive (2022)

[27] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, pp. 212–219 (1996)

[28] Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics **305**, 53–74 (2002)

[29] Shenvi, N., Kempe, J., Whaley, K.B.: Quantum random-walk search algorithm. Physical Review A **67**(5), 052307 (2003)

[30] Pati, A.K., Braunstein, S.L.: Impossibility of deleting an unknown quantum state. arXiv preprint quant-ph/9911090 (1999)

[31] Dervovic, D., Herbster, M., Mountney, P., Severini, S., Usher, N., Wossnig, L.: Quantum linear systems algorithms: a primer. arXiv preprint arXiv:1802.08227 (2018)

[32] Giovannetti, V., Lloyd, S., Maccone, L.: Quantum random access memory. Physical review letters **100**(16), 160501 (2008)

[33] Park, D.K., Petruccione, F., Rhee, J.-K.K.: Circuit-based quantum random access memory for classical data. Scientific reports **9**(1), 1–8 (2019)