Rotational cryptanalysis of round-reduced Keccak

Paweł Morawiecki[1], Josef Pieprzyk[2], and Marian Srebrny[1,3]

[1] Section of Informatics, University of Commerce, Kielce, Poland
pawelm@wsh-kielce.edu.pl

[2] Department of Computing, Macquarie University, Australia
josef.pieprzyk@mq.edu.au

[3] Institute of Computer Science, Polish Academy of Sciences, Poland
marians@ipipan.waw.pl

Abstract. In this paper we attack round-reduced Keccak hash function with a technique called rotational cryptanalysis. We focus on Keccak variants proposed as SHA-3 candidates in the NIST's contest for a new standard of cryptographic hash function. Our main result is a preimage attack on 4-round Keccak and a 5-round distinguisher on Keccak-f[1600] permutation — the main building block of Keccak hash function.

Key words: preimage attack, Keccak, rotational cryptanalysis, SHA-3

1 Introduction

In 2007, the U.S. National Institute of Standards and Technology (NIST) announced a public contest aiming at the selection of a new standard for a cryptographic hash function. The main motivation behind starting the contest has been the security flaws identified in the SHA-1 standard in 2005. Similarities between SHA-1 and the most recent standard SHA-2 were worrisome and NIST decided that a new, stronger hash function would be needed. Overall, 51 functions were submitted to the first round of the contest. In July 2009 out of the submitted functions, 14 were selected to the second round. At the end of 2010, the five finalists were announced. The Keccak hash function [5] is one of them. In this paper we analyze Keccak using a technique called rotational cryptanalysis.

Rotational analysis is a relatively new type of attack. The technique was mentioned and applied in [2, 12, 14], and formally introduced in [10]. Unlike the differential analysis, where for a pair (x, y) the attacker follows the propagation of the difference $x \oplus y$ through the cryptographic system, in the rotational analysis, the adversary investigates the propagation of the rotational pair through the cryptographic transformations. Khovratovich and Nikolić in [10] analyze the primitives composed of only three operations: addition, rotation, xor (ARX). For these primitives, they prove that the probability that a rotational pair of inputs will produce a rotational pair on the output depends on the number

of additions only. In [15] a rotational distinguisher was designed for the keyed permutation of the Shabal hash function. Rotational cryptanalysis was combined with the rebound attack and applied to the compression function of the SHA-3 candidate Skein and its underlying cipher Threefish [11].

The known cryptanalytic results on KECCAK can be divided into two types. The first type is showing a non-random behaviour, weakness in the KECCAK's internal permutation, such as our rotational distinguishers. The second type is the attacks on the core security properties of the whole function (a preimage attack and a collision attack). The most successful result (in terms of a number of rounds) on the KECCAK's permutation is the zero-sum distinguisher proposed in [1] and later improved in [6, 7]. However, the complexities of these distinguishers is very high. For example, the zero-sum distinguisher for all 24 rounds has the complexity of 2^{1579}. A differential analysis of KECCAK's internal permutation, given in [8], leads to distinguishers up to 8 rounds with complexity of $2^{491.47}$. Among the attacks on the KECCAK hash function, the most rounds were broken by Bernstein in his 8-round preimage attack [3]. However, the complexity of the attack is only marginally lower than exhaustive search (around half a bit for the 8-round attack). Also with the aid of differential analysis, Naya-Plasencia et al. mounted the preimage and collision attacks on 2-round KECCAK [13]. In [9] the same result (2-round preimage and 2-round collision attacks) were obtained through the SAT-based attacks.

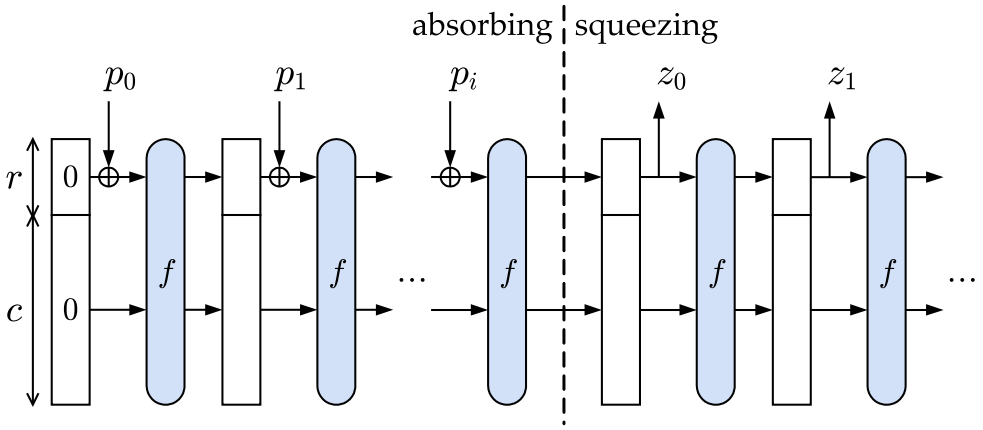
In this paper we focus our analysis on the KECCAK variants proposed as SHA-3 candidates. First we analyze the permutation KECCAK-f[1600]. We mount the 4-round rotational distinguisher and then enhance it with a correlation analysis which improves the result to 5 rounds. We implement the distinguishers and verify the experimental results against the theoretical model. A family of 4-round distinguishers is the base for our 4-round preimage attack with the complexity of 64 times lower than exhaustive search.

2 KECCAK

In this section we provide a description of KECCAK to the extent necessary for understanding the attack described in the paper. For a complete specification, we refer the interested reader to the original specification [5].

KECCAK uses the sponge construction and hence is a member of the sponge function family [4]. Figure 1 shows the construction. It can be used as a hash function but also can be applied for generating infinite bit stream, making it suitable as a stream cipher or a pseudorandom bit generator. In this paper we focus on the sponge construction for cryptographic hashing. KECCAK has two main parameters r and c, which are called bitrate and capacity, respectively. The sum of those two makes the state size, which KECCAK operates on. For the SHA-3 proposal, the state size is 1600 bits. Different values for bitrate and capacity give the trade-off between speed and security. The higher bitrate gives the faster function that is less secure. KECCAK follows the sponge two-phase processing. In the first phase (also called the absorbing phase), r-bit input message blocks

Fig. 1. Sponge Construction [4]



are xored with the first r bits of the state, interleaved with applications of the function f (called Keccak-f in the specification). The absorbing phase is finished when all message blocks have been processed. In the second phase (also called the squeezing phase), the first r bits of the state are returned as part of the output bits, interleaved with applications of the function f. The squeezing phase is finished after the desired length of output digest has been produced.

The default values for Keccak are $r = 1024$, $c = 576$, which gives 1600-bit state. Keccak can also operate on smaller states but through the whole paper we always refer to the default variant with 1600-bit state. The state can be visualised as an array of 5×5 lanes, each lane is 64-bit long. The state size determines the number of rounds in Keccak-f function. For the default 1600-bit state there are 24 rounds. All rounds are the same except for constants which are different for each round.

Below there is a pseudo-code of the single round. In the latter part of the paper, we often refer to the algorithm steps (denoted by Greek letters) described in the following pseudo-code.

```
Round(A,RC) {

θ step
C[x] = A[x,0] xor A[x,1] xor A[x,2] xor
       A[x,3] xor A[x,4],                        forall x in (0...4)
D[x] = C[x-1] xor rot(C[x+1],1),                 forall x in (0...4)
A[x,y] = A[x,y] xor D[x],              forall (x,y) in (0...4,0...4)

ρ and π steps                          forall (x,y) in (0...4,0...4)
```

```
B[y,2*x+3*y] = rot(A[x,y], r[x,y]),

χ step                                    forall (x,y) in (0...4,0...4)
A[x,y] = B[x,y] xor ((not B[x+1,y]) and B[x+2,y]),

ι step
A[0,0] = A[0,0] xor RC

return A   }
```

All the operations on the indices shown in the pseudo-code are done modulo 5. `A` denotes the complete permutation state array and `A[x,y]` denotes a particular lane in that state. `B[x,y]`, `C[x]`, `D[x]` are intermediate variables. The constants `r[x,y]` are the rotation offsets, while `RC` are the round constants. `rot(W,m)` is the usual bitwise cyclic shift operation, moving bit at position i into position $i+m$ in lane `W` ($i+m$ are done modulo 64 – note that 64 is the lane size for the default variant of Keccak).

3 Rotational distinguishers for the Keccak-f[1600] permutation

A rotational distinguisher takes advantage of the fact that some transformations preserve the rotational relation, i.e. if a rotational pair is given as the input of the transformation, then the corresponding outputs are a rotation pair. Let us define a rotational pair in the context of the Keccak-f[1600] permutation.

Definition 1. *A rotational pair is a pair of 1600-bit states (A, $A^{\leftarrow}$), where each lane in the state $A^{\leftarrow}$ is created by bitwise cyclic shift operation of the corresponding lane in the state A. The operation moves the bit $A_{(x,y,z)}$ into the position $A^{\leftarrow}_{(x,y,z+n)}$ (z+n is done modulo 64). The coordinates x, y range from 0 to 4 specifying the lane in the state, the coordinate z ranges from 0 to 63 specifying the bit number in the given lane, n is called a rotational number and is the same for every lane.*

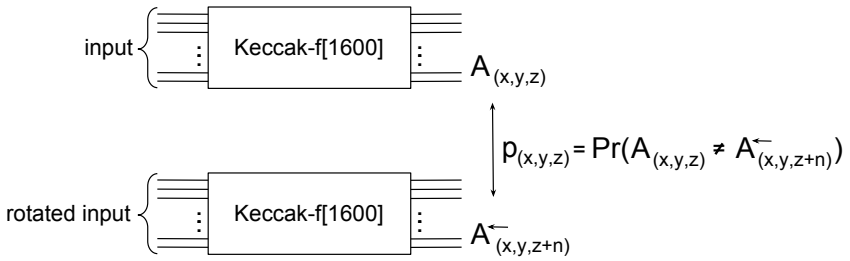
With the notation introduced in Definition 1, we can refer to a state by $A^{\leftarrow}$, to a lane by $A_{(3,2)}$, to a value of a single bit by $A^{\leftarrow}_{(1,4,6)}$, or to a position of a single bit by $(3, 1, 60)$.

Remark 1. A direct consequence of Definition 1 is that for a given state A, there are up to 64 possible rotational pairs including a pair, where A and $A^{\leftarrow}$ are the same (having $n = 0$). We will use this fact in the preimage attack described later in the paper.

When we trace a rotational pair through the Keccak-f[1600] algorithm, it turns out that after a number of steps there are very few bits, which preserve a rotational relation, that is $A_{(x,y,z)} = A^{\leftarrow}_{(x,y,z+n)}$. For our analysis, it is useful to define the probability that the two corresponding bits preserve the rotational relation.

Definition 2. *A probability $p_{(x,y,z)}$ is the probability that two corresponding bits $A_{(x,y,z)}$ and $A^{\leftarrow}_{(x,y,z+n)}$ have different values and is expressed as $p_{(x,y,z)} = 1/2 + \epsilon_{(x,y,z)}$. Therefore if $\epsilon_{(x,y,z)} = 1/2$, the corresponding bits have opposite values and if $\epsilon_{(x,y,z)} = -1/2$, the corresponding bits are equal. In case $\epsilon_{(x,y,z)} = 0$, the bits are independent.*

Fig. 2. Probabilistic relation between bits in a rotational pair



For a random permutation, the probability $p_{(x,y,z)}$ should be equal to 0.5 for all (x, y, z). Thus if we can show that for KECCAK-f[1600] for some (x, y, z), the probability $p_{(x,y,z)}$ is expected to deviate from 0.5, then we have a distinguisher. To calculate how the probabilities change through the successive steps of the algorithm, let us first analyze two basic bitwise operations used in KECCAK.

Lemma 1 (AND). *Given the AND operation, its input bits a, b and the output bit out. Then the probability*

$$p_{out} = \frac{1}{2}((1 - p_a)p_b + (1 - p_b)p_a + p_a p_b),$$

where the probabilities p_a and p_b are defined according to Definition 2.

Lemma 2 (XOR). *Given the XOR operation, its input bits a, b and the output bit out. Then the probability*

$$p_{out} = (1 - p_a)p_b + (1 - p_b)p_a,$$

where the probabilities p_a and p_b are defined according to Definition 2.

Proofs of the lemmas are given in Appendix.

There is also the bitwise NOT operation in the algorithm but it does not affect the probabilities. NOT flips the values of the corresponding bits $A_{(x,y,z)}$ and $A^{\leftarrow}_{(x,y,z+n)}$ but their relation (or precisely speaking the probability of relation

$p_{(x,y,z)}$) remains unchanged. Also the bitwise cyclic shift operation (denoted in the pseudo-code as `rot(W,n)`) does not change the values of probabilities. It rotates the bits in the lane so their positions (coordinates z in $p_{(x,y,z)}$) change while their probabilities $p_{(x,y,z)}$ are not changed.

Having explained how the basic bitwise operations change the rotation probabilities, the analysis of the KECCAK-f[1600] steps remains mostly straightforward. In the transformation θ, there is the XOR operation only, so Lemma 2 is sufficient to calculate the rotation probabilities. For the transformations ρ and π, nothing needs to be calculated as only the position of bits change. In the transformation χ, the two Lemma 1 and Lemma 2 are applied. The last step is the transformation ι, where a lane is xored with a constant. Xoring with '0' does not change anything. However, if there is '1' at position n in the constant, then xoring with a constant change the probabilities as follows

$$p_{(x,y,z)} := 1 - p_{(x,y,z)} \text{ and}$$
$$p_{(x,y,z-n)} := 1 - p_{(x,y,z-n)}$$

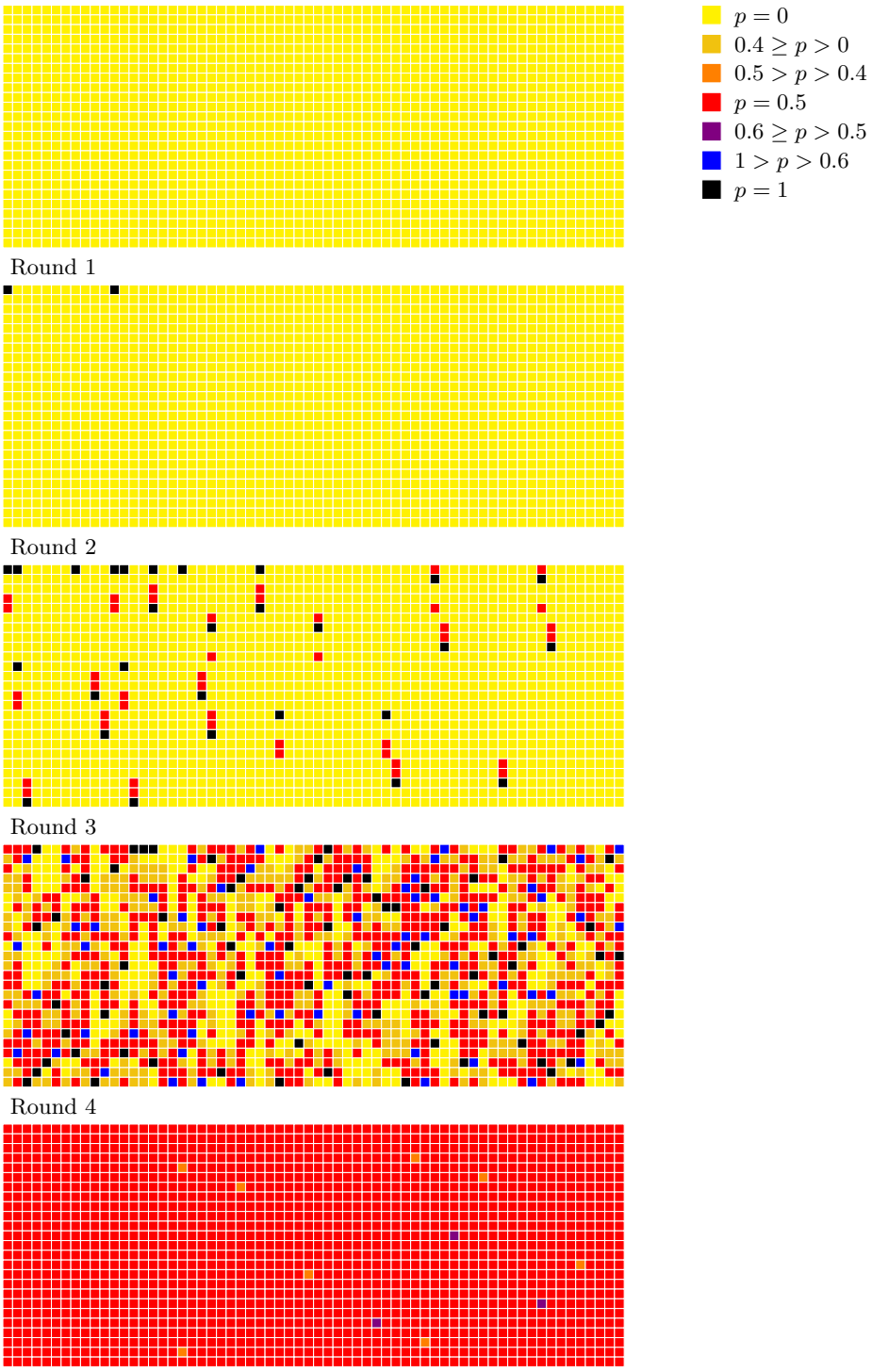
Example 1. Let us consider two 8-bit lanes $A_{(0,0)}$ and $A^{\leftarrow}_{(0,0)}$ with the following binary values: $A_{(0,0)} = 00000010$ and $A^{\leftarrow}_{(0,0)} = 00010000$.
Note that $A^{\leftarrow}_{(0,0)}$ is rotated by three positions thus $n = 3$. Because $\forall_z\ A_{(0,0,z)} = A^{\leftarrow}_{(0,0,z+3)}$, then $\forall_z\ p_{(x,y,z)} = 0$ (according to Definition 2). Now if both lanes are xored with 8-bit constant $C = 00000001$, new values of lanes are $A_{(0,0)} = 00000011$ and $A^{\leftarrow}_{(0,0)} = 00010001$. Rotational relation has been spoilt at two positions (0 and 5), therefore the probabilities $p_{(0,0,0)}$ and $p_{(0,0,5)}$ is now equal to 1. In KECCAK-f[1600] the constants are 64-bit long but the reasoning shown above is still valid.

3.1 4-round distinguishers

We build a 4-round rotational distinguisher and show that after 4 rounds, there are some coordinates (x, y, z) for which $p_{(x,y,z)}$ deviates from 0.5. Figure 3 illustrate an evolution of rotation probabilities. A single square represents a value (or a range of values) of the probability $p_{(x,y,z)}$. Usually in this paper, we refer to a lane by its two coordinates (x, y). However here for the sake of diagram readability instead of 5x5 matrix of lanes there are 25 rows, each representing a single lane. For example, a value of $p_{(0,1,0)}$ is represented by the leftmost square in the sixth row and $p_{(4,4,63)}$ is represented by the rightmost square in the last (25th) row.

At the beginning, all corresponding bits from a rotational pair are equal so $\forall_{x,y,z}\ p_{(x,y,z)} = 0$. After the first application of ι, some probabilities $p_{(x,y,z)}$ change and in the subsequent steps these changes propagate and influence other bits. For most rotational numbers n, there are some probabilities $p_{(x,y,z)}$ deviating from 0.5 until the end of the 4th round. After θ in the 5th round, all $p_{(x,y,z)} = 0.5$, thus the round-reduced KECCAK-f[1600] cannot be distinguished from a random permutation.

Fig. 3. Evolution of probabilities $p_{(x,y,z)}$ through 4 rounds of KECCAK-f[1600]. A rotational number n set to 11.



We carried out the experiments to verify whether the 4-round distinguisher gives results as our model predicts. For example, according to our calculations, at the end of 4th round for $n = 10$, the probability $p_{(4,4,14)} = 0.5625$. We chose randomly 10000 rotational pairs and ran them on the 4-round KECCAK-f[1600]. Results of the experiment confirm the value of $p_{(4,4,14)}$. For around 5625 rotational pairs considered, bits had different values. Note that for a random permutation, the count should be very close to 5000. We experimented also with other values of (x, y, z) and rotational numbers n, each time obtaining the results consistent with the theory.

3.2 Extension to 5-round distinguisher

To extend the distinguisher to 5 rounds, we show that for some bits a probability of correlation between them deviates from 0.5. Let us first give an observation which helps to mount the 5-round distinguisher.

Observation 1 Consider two bits $(A_{(x,y,z)}, A_{(x,y',z)})$ from state A which are in the same column and let us assume that we know correlation (or in general, probability of correlation) between them (e.g. probability that $A_{(x,y,z)} \neq A_{(x,y',z)}$). Our point is that θ does not change this correlation (or probability of correlation). It is because θ treats each bit within a column in the same way: either it flips all 5 of them or it leaves them unchanged.

We can use this observation in our rotational analysis. The only difference is that instead of a probability of correlation between bits $A_{(x,y,z)}$ and $A_{(x,y',z)}$, we speak about a probability of correlation between one pair $(A_{(x,y,z)}, A^{\leftarrow}_{(x,y,z+n)})$ and the second pair $(A_{(x,y',z)}, A^{\leftarrow}_{(x,y',z+n)})$. Each of these two pairs has the relation between its bits (that is bits have either the same or opposite values). We are interested whether the relations are the same in both pairs, specifically the probability that relations are the same in both pairs. Clearly, for a random permutation the answer should be equal to 0.5. Thus if we can show that for KECCAK-f[1600] for some bits the value deviates from 0.5, then we have a distinguisher.

First we determine a rotational number n for which $p_{(x,y,z)}$ and $p_{(x,y',z)}$ have the highest deviation from 0.5 at the end of the 4th round. It turns out that for $n = 1$, $p_{(2,1,37)}$ and $p_{(2,2,37)}$ is the best pair ($p_{(2,1,37)} = 0.5625$ and $p_{(2,2,37)} = 0.49219$).
Now let p_c denotes a probability that in the first pair $(A_{(x,y,z)}, A^{\leftarrow}_{(x,y,z+n)})$ and in the second pair $(A_{(x,y',z)}, A^{\leftarrow}_{(x,y',z+n)})$ is the same relation. That is the probability:

$$p_c = p_{(x,y,z)} \cdot p_{(x,y',z)} + (1 - p_{(x,y,z)})(1 - p_{(x,y',z)})$$

We can calculate p_c for the chosen pair $p_{(2,1,37)}$ and $p_{(2,2,37)}$.

$$p_c = 0.5625 \cdot 0.49219 + (1 - 0.5625)(1 - 0.49219) = 0.49902375$$

This is the p_c value at the beginning of the 5th round. Then we have to examine how the steps in the algorithm change this probability. As explained in Observation 1, θ does not change this value. Subsequent algorithm steps ρ and π also do not change p_c value, they only change a position of p_c which now refers to different pairs of bits $(A_{(1,2,43)}, A^{\leftarrow}_{(1,2,44)})$ and $(A_{(2,0,16)}, A^{\leftarrow}_{(2,0,17)})$. After that there is χ which preserves the relation between the first pair and the second with a probability equals 0.53125. (Detailed calculations are given in Appendix.) Finally, ι does not affect our analysis here. Therefore, to have our pairs with the same relation at the end of the 5th round, there are two ways this event may occur. Either the pairs enter into the 5th round with the same relation and χ does not spoil it or they enter into the 5th round with the opposite relation and χ 'fixes' it. Then the total probability p_c for the chosen pair at the end of the 5th round is:

$$p_c = 0.53125 \cdot 0.49902375 + (1 - 0.53125) \cdot (1 - 0.49902375) = 0.499938984$$

For a random permutation p_c should be equal to 0.5; so, the bias is very small. To experimentally verify and observe the bias we need to check many rotational pairs. A sufficient number of rotational pairs m is calculated from Chernoff bound and can be expressed as the following inequality:

$$m \geq \frac{1}{(p_c - 0.5)^2} \ln \frac{1}{\sqrt{\epsilon}},$$

where ϵ is the probability of an error of the bound (typically set to 0.05). From the inequality we have $m \geq 402\,332\,890$ and in the experiment we checked $403\,000\,000$ rotational pairs. The distinguisher can be described in a few short steps:

1. Generate randomly $403\,000\,000$ rotational pairs
2. **For each** pair
   (a) Run 5-round Keccak-f[1600] on the state A and the state $A^{\leftarrow}$;
   (b) **if** $(A_{(1,2,43)} = A^{\leftarrow}_{(1,2,44)})$ **and** $(A_{(2,0,16)} = A^{\leftarrow}_{(2,0,17)})$ **or** $(A_{(1,2,43)} \neq A^{\leftarrow}_{(1,2,44)})$ **and** $(A_{(2,0,16)} \neq A^{\leftarrow}_{(2,0,17)})$ **then** $counter := counter + 1$;

The result from the experiment was consistent with the predicted bias. In fact the observed bias was even slightly higher ($201\,450\,503$ pairs had the same relation whereas the predicted value is $p_c \cdot 403\,000\,000 = 201\,475\,410$.) Note that for a random permutation, the counter should be very close to $201\,500\,000$.

4 Preimage attacks on round-reduced Keccak

First, we describe the preimage attack on 3-round Keccak which is based on the rotational distinguisher given in the previous section. Then we show how to extend the attack to 4 rounds. To have the attack working on Keccak hash function, we have to consider padding and Keccak parameters. Let us consider Keccak candidate for SHA3-512, that is Keccak with $r = 576$, $c = 1024$ and

a hash length set to 512 bits. For the preimage attack we propose the following structure of the message. A message length is 574 bits, where first 8 lanes (512 bits) are unknown (to determine by the attacker). Last 62 bits of the message are set to 1. The message is padded with two 1s giving a block of 576 bits. This way we fulfill a condition that all lanes (except first 8 lanes) have all 0s or 1s. Similarly we would mount the constraints on a message when attacking KECCAK with different parameters (including all KECCAK variants proposed as SHA-3 candidates).

4.1 3-round preimage attack

The goal of our attack is to find a preimage for a given 512-bit hash h. In the structure of the message described above we have 512 unknown bits, then we can expect that among 2^{512} possible messages there is at least one with a given hash. The main idea of our attack is to find a rotational counterpart of the preimage and show that the workload for this task is below exhaustively trying all 2^{512} values. Once we have a rotational counterpart of the preimage, we simply rotate it back and get the preimage.

As stated in Remark 1, for a given state A there are up to 64 possible rotational pairs (including the identity function). In the state A there are 512 unknown bits, then the probability that we guess one of the rotational pairs (an unknown A and its rotational counterpart $A^{\leftarrow}$) is $64 \cdot 2^{-512} = 2^{-506}$. Thus we need 2^{506} guesses. There is a subtlety here which should be mentioned. There are some messages which have fewer than 64 rotations. These 'special' messages have a cyclic pattern. For example a message starting with four 0s then four 1s, then four 0s and so on. However, the number of 'special' messages is relatively small in comparison to 2^{512}. It can be shown there are 2^{256} such messages for our case. (See Appendix for detailed analysis.) For simplicity, we can start our attack with checking 2^{256} these special messages. Then there are still almost 2^{512} possibilities left, but at least we are sure that in this poll each state can give 64 rotational pairs.

Before launching the main loop of the attack, it is useful to generate 64 diagrams (the same as shown in Figure 3), each with a different rotational number n. It allows to determine the coordinates (x, y, z) for which $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$ at the end of the 3rd round. Because the attacker knows only 512 bits of a hash, we have to consider only (x, y, z) such that $64x + 320y + z < 512$.

Here is the main loop of the attack given in the following pseudo-code:

1. guess first 8 lanes (512 bits) of the state $A^{\leftarrow}$, the other bits are fixed according to the structure of the message given above.
2. run 3-round KECCAK on the state $A^{\leftarrow}$.
3. **for** $n := 0$ **to** $n < 64$ **do**
   (a) *candidate* := **true**;
   (b) **for all** (x, y, z) such that $64x + 320y + z < 512$ **do**
   if $(p_{(x,y,z)} = 0)$ **and** $(A_{(x,y,z)} \neq A^{\leftarrow}_{(x,y,z+n)})$ **then** *candidate* := **false**;

if $(p_{(x,y,z)} = 1)$ **and** $(A_{(x,y,z)} = A^{\leftarrow}_{(x,y,z+n)})$ **then** *candidate* := **false**;

(c) **if** (candidate=**true**) **then** rotate back the guessed state by n bits and run 3-round KECCAK on it to check whether the state is the preimage of a given hash.

The attacker compares the probabilities $p_{(x,y,z)}$ from the distinguisher with the actual values of A (the given hash) and $A^{\leftarrow}$ state (a result of 3-round KECCAK on a guessed state). So, for example, if $p_{(2,3,1)} = 0$, then the bits $A_{(2,3,1)}$ and $A^{\leftarrow}_{(2,3,1+n)}$ have to be the same. If the bits are different, then the candidate is rejected as a potential rotational counterpart of the preimage. (It is the point in the pseudo-code where a variable *candidate* becomes false.)

As said earlier, running the main loop 2^{506} times, we should get one rotational counterpart of the preimage. It could be the case that our guess (candidate) of a rotational counterpart is not rejected, but in fact it is not a rotational counterpart. Let us call it a false positive candidate. There will be many such false positive candidates and the number of them is calculated as follows. For each distinguisher (each with a different rotational number n), one can calculate the number of triples (x, y, z) for which $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$ at the end of the 3rd round. For example, for the distinguisher with $n = 11$ (as shown in Figure 3), there are 149 triples (x, y, z) for which $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$ at the end of the 3rd round. (We remind that we consider only (x, y, z) such that $64x + 320y + z < 512$.) The best case is for the distinguisher with $n = 1$, for which the number of triples is 183 and the worst case is for $n = 32$ with the number of triples equals 124. A probability that each of 124 $p_{(x,y,z)}$ has a correct value (as the rotational distinguisher predicts) is 2^{-124} and hence there will be around $2^{512}/2^{124} = 2^{388}$ false positive candidates to check. (2^{388} is the upper bound of the number of false positive candidates as we take the worst case for the calculation.)

Summing up, the workload of the attack is 2^{256} (checking special messages) + 2^{506} (main loop) + 2^{388} (checking false positive candidates). Thus an asymptotic complexity of the attack is $O(2^{506})$, 64 times better than the exhaustive search.

4.2 Extension to 4-round preimage attack

A direct extension of the attack to 4 rounds is not possible since there are not any $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$ in the 4th round of the rotational distinguisher. To make it possible we take advantage of the following observation.

Observation 2 In Example 1, Section 3 it was shown that application of ι to A and $A^{\leftarrow}$ states flips the value $p_{(x,y,z)}$ for some triples (x, y, z). Precisely, the number of flips is equal to Hamming weight of a round constant and this value is multiplied by two. So if there are three 1's, there will be six flips. Our point is that if we do not apply ι to an $A^{\leftarrow}$ state, there will be half as many flips. Let us see a simple example similar to Example 1.

Example 2. Let us consider two 8-bit lanes $A_{(0,0)}$ and $A^{\leftarrow}_{(0,0)}$ with the following binary values: $A_{(0,0)} = 00000010$ and $A^{\leftarrow}_{(0,0)} = 00010000$.
Note that $A^{\leftarrow}_{(0,0)}$ is rotated by three positions thus $n = 3$. Now $A_{(0,0)}$ is xored with 8-bit constant $C = 00000001$ and $A^{\leftarrow}_{(0,0)}$ is left without changes. Then we have $A_{(0,0)} = 00000011$ and the unchanged $A^{\leftarrow}_{(0,0)} = 00010000$. Therefore a rotational relation has been spoilt at only one position so now $p_{(0,0,0)}$ is equal to 1. In KECCAK-f[1600] the constants are 64-bit long but the reasoning shown here stays the same.

What is important from this simple observation is that fewer flips lead to fewer $p_{(x,y,z)}$ with undesirable 0.5 value. In consequence, now in the 4rd round there are 9 triples (x, y, z) for which $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$. These triples fulfil the condition $64x + 320y + z < 512$ as the attacker is given only 512 bits of a hash. In fact $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$ are not at the end of the 4rd round but before χ in the 4rd round. (Step χ destroys these desirable probabilities.) Fortunately, we can invert ι and χ from the given hash as χ operates on the rows independently and can be inverted on a row-by-row basis. In Appendix we give a diagram showing how the probabilities $p_{(x,y,z)}$ evolve and propagate in the modified version of KECCAK without ι.

The preimage attack on the 4-round KECCAK is very similar to the 3-round variant, they differ only in a few places. First, we need to invert ι and χ of the 4rd round from the given hash. Then, in Step 2 of the pseudo-code instead of running a normal 4-round KECCAK, we run a modified version without ι (in all 4 rounds) and without ι and χ in the 4rd round. Finally, there will be more false positive candidates as there are only 9 triples (x, y, z) for which $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$. The complexity of the attack stays the same as in the 3-round attack. That is 2^{256} (checking special messages) + 2^{506} (main loop) + 2^{503} (checking false positive candidates). It gives $O(2^{506})$.

Our preimage attack works also on KECCAK candidates for SHA3-224, SHA3-256, and SHA3-384. The complexity of the attack remains the same, the only difference is a higher number of false positive candidates to check. It is because an attacker knows fewer bits of a hash (a hash is shorter in these variants) and consequently there are fewer triples (x, y, z) for which $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$. Please note that if we try to attack KECCAK variant with higher bitrate r (e.g. a variant with $r = 600$ and $c = 1000$), the claimed security for this variant is $2^{c/2} = 2^{500}$. In such a case our attack is not actually an attack as its complexity is higher than the claimed security provided by designers.

We could not extend the attack to 5 or more rounds because in the 5th round all $p_{(x,y,z)} = 0.5$, as expected from a random permutation.

5 Conclusion

In this paper, we have presented the rotational distinguisher for KECCAK-f[1600] permutation — the main building block of the KECCAK hash function. The distinguisher has been enhanced with the correlation analysis, allowing us to reach

5 rounds with the complexity of 2^{29}. We have implemented and verified the distinguisher and experimental results have been consistent with the theoretical model. A family of 4-round distinguishers help us to mount the 4-round preimage attack with the complexity of 2^{506}. All the presented attacks are valid for all the KECCAK variants submitted as SHA-3 candidates. As future work, it would be interesting to investigate whether the differential rebound attack could improve the rotational distinguishers. These two types of analysis (rebound and rotational) were combined in the attacks on Skein hash function [11].

References

1. Aumasson, J.P., Meier, W.: Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. Tech. rep., NIST mailing list (2009)
2. Bernstein, D.J.: Salsa20. Tech. rep., eSTREAM, ECRYPT Stream Cipher Project (2005), http://cr.yp.to/snuffle.html
3. Bernstein, D.J.: Second preimages for 6 (7? (8??)) rounds of Keccak? NIST mailing list (2010), http://ehash.iaik.tugraz.at/uploads/6/65/NIST-mailing-list_Bernstein-Daemen.txt
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponges, http://sponge.noekeon.org
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak sponge function family main document, http://keccak.noekeon.org/Keccak-main-2.1.pdf
6. Boura, C., Canteaut, A.: Zero-Sum Distinguishers for Iterated Permutations and Application to Keccak-f and Hamsi-256. In: Biryukov, A., Gong, G., Stinson, D. (eds.) Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 6544, pp. 1–17. Springer Berlin Heidelberg (2011)
7. Duan, M., Lai, X.: Improved zero-sum distinguisher for full round Keccak-f permutation. Chinese Science Bulletin 57, 694–697 (2012)
8. Duc, A., Guo, J., Peyrin, T., Wei, L.: Unaligned Rebound Attack - Application to Keccak. Cryptology ePrint Archive, Report 2011/420 (2011)
9. Homsirikamol, E., Morawiecki, P., Rogawski, M., Srebrny, M.: Security margin evaluation of SHA-3 contest finalists through SAT-based attacks. In: 11th Int. Conf. on Information Systems and Industrial Management. LNCS, vol. 7564. Springer Berlin Heidelberg (2012)
10. Khovratovich, D., Nikolić, I.: Rotational cryptanalysis of ARX. In: Proceedings of the 17th international conference on Fast software encryption. pp. 333–346. LNCS, Springer-Verlag (2010)
11. Khovratovich, D., Nikolic, I., Rechberger, C.: Rotational Rebound Attacks on Reduced Skein. In: ASIACRYPT'10. LNCS, vol. 6477, pp. 1–19 (2010)
12. Knudsen, L.R., Matusiewicz, K., Thomsen, S.S.: Observations on the Shabal keyed permutation. Available online (2009), http://www.mat.dtu.dk/people/S.Thomsen/shabal/shabal.pdf
13. Naya-Plasencia, M., Rck, A., Meier, W.: Practical analysis of reduced-round keccak. In: Bernstein, D., Chatterjee, S. (eds.) Progress in Cryptology INDOCRYPT 2011, Lecture Notes in Computer Science, vol. 7107, pp. 236–254. Springer Berlin Heidelberg (2011)
14. Standaert, F.X., Piret, G., Gershenfeld, N., Quisquater., J.J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: CARDIS'06. LNCS, vol. 3928, pp. 222–236 (2006)

15. Van Assche, G.: A rotational distinguisher on Shabals keyed permutation and its impact on the security proofs. Available online, `http://gva.noekeon.org/papers/ShabalRotation.pdf`

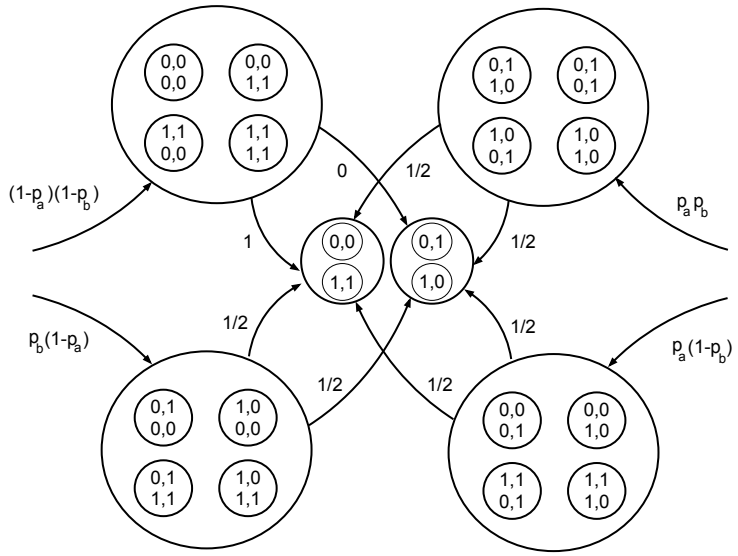
Appendix

Proof of Lemma 1

Analyzing the AND operation we consider two pairs of input bits. A pair from an A state and its counterpart from an $A^{\leftarrow}$ state. There are 16 possible combinations of pairs and we group them in fours. It is shown in Figure 4. Probabilities of getting the given group are also shown. The most inner circles represents pair of output bits (one bit from an A state and its counterpart from an $A^{\leftarrow}$ state). It is clear from Figure 4 that four paths lead to a circle with output bits having opposite values (pairs (0,1) and (1,0)). Actually, one path has probability 0 thus a calculation of p_{out} (a probability that output bits have opposite values) comes down to adding probabilities of the three paths. We have:

$$p_{out} = p_a p_b \cdot \frac{1}{2} + (1-p_a)p_b \cdot \frac{1}{2} + (1-p_b)p_a \cdot \frac{1}{2} = \frac{1}{2}((1-p_a)p_b + (1p_b)p_a + p_a p_b)$$

Fig. 4. All possible 'paths' for the bitwise AND operation for rotational pairs of bits.

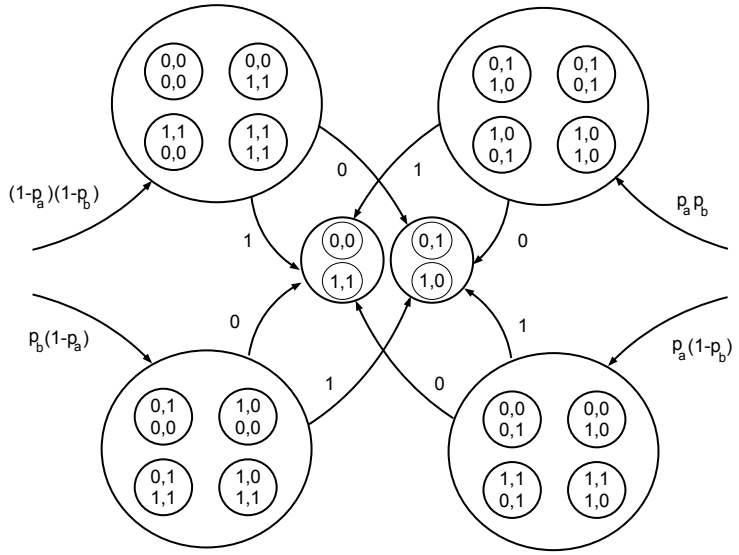


Proof of Lemma 2

Proof of Lemma 2 is the same as for Lemma 1. The only difference is that now there are only two paths leading to a circle with output bits having opposite values. It is shown in Figure 5. We have:

$$p_{out} = (1 - p_a)p_b \cdot 1 + (1 - p_b)p_a \cdot 1 = (1 - p_a)p_b + (1 - p_b)p_a$$

Fig. 5. All possible 'paths' for the bitwise XOR operation for rotational pairs of bits.



Probability of relation preservation by χ

We are given two pairs of bits $(A_{(1,2,43)}, A^{\leftarrow}_{(1,2,44)})$ and $(A_{(2,0,16)}, A^{\leftarrow}_{(2,0,17)})$. Each of these two pairs has the relation between its bits (that is bits have either the same or opposite values). We can also look at relation between pairs and there are two possibilities: either the same relation in both pairs or different relation in each pair. For example a pair (0,1) and a pair (1,0) means that relations in both pairs is the same (bits are different in pairs). We are interested in a probability that χ preserves the relation between pairs. χ changes the values of bits in the following way.

$$\left.\begin{array}{l} A_{(1,2,43)} := A_{(1,2,43)} \text{ XOR } (A_{(2,2,43)} \text{ AND } A_{(3,2,43)}) \\ A^{\leftarrow}_{(1,2,44)} := A^{\leftarrow}_{(1,2,44)} \text{ XOR } (A^{\leftarrow}_{(2,2,44)} \text{ AND } A^{\leftarrow}_{(3,2,44)}) \end{array}\right\} \quad \text{first rotational pair}$$

$$\left.\begin{array}{l} A_{(2,0,16)} := A_{(2,0,16)} \text{ XOR } (A_{(3,0,16)} \text{ AND } A_{(4,0,16)}) \\ A^{\leftarrow}_{(2,0,17)} := A^{\leftarrow}_{(2,0,17)} \text{ XOR } (A^{\leftarrow}_{(3,0,17)} \text{ AND } A^{\leftarrow}_{(4,0,17)}) \end{array}\right\} \text{ second rotational pair}$$

To keep the relation between bits $A_{(1,2,43)}$ and $A^{\leftarrow}_{(1,2,44)}$ from the first pair, the result of the AND operation has to be the same in both equations from the first pair. The probability of such event can be calculated from Figure 4. We add probabilities (paths) leading to the left, inner circle. (This circle represents the output bits with the same values.) Then a probability is:

$$p = (1-p_a)(1-p_b)\cdot 1 + p_a p_b \cdot \frac{1}{2} + (1-p_a)p_b \cdot \frac{1}{2} + (1-p_b)p_a \cdot \frac{1}{2}$$

In the 5th round, after θ all $p_{(x,y,z)} = \frac{1}{2}$, then a numerical value of p is:

$$p = (1-\frac{1}{2})(1-\frac{1}{2})1 + \frac{1}{2}\frac{1}{2}\frac{1}{2} + (1-\frac{1}{2})\frac{1}{2}\frac{1}{2} + (1-\frac{1}{2})\frac{1}{2}\frac{1}{2} = \frac{5}{8}$$

For the second rotational pair calculations are exactly the same with the result of $\frac{5}{8}$. The event that χ preserves the relation between the first and second pair can happen either when the relation in each pair is preserved or the relation in each pair is spoilt. Thus the probability of this event is equal to:

$$p_{event} = \frac{5}{8}\cdot\frac{5}{8} + (1-\frac{5}{8})(1-\frac{5}{8}) = \frac{34}{64} = 0.53125$$

Calculation of a number of special messages

According to Definition 1 and Remark 1, for a given state A there are up to 64 possible rotational pairs (including an identity function). There are some messages which have fewer than 64 rotations. These special messages must have a cyclic pattern (e.g. alternating four 1's and four 0's) in all lanes. All 0's or all 1's in the given lane are also considered cyclic here. Please note that if at least one lane in a state A is not cyclic then there are exactly 64 possible rotational pairs (A, $A^{\leftarrow}$). It is because this non-cyclic lane is distinct for each rotational number n and consequently the whole $A^{\leftarrow}$ will be distinct.

For a 64-bit lane there are 2^{32} cyclic patterns. In our preimage attack there are 8 unknown lanes in the A state (rest lanes are fixed and cyclic), so the number of combinations of cyclic patterns in these 8 lanes is: $\underbrace{2^{32}\cdot 2^{32}\ldots\cdot 2^{32}}_{8 \text{ factors}} = 2^{256}$. And hence the number of special messages is 2^{256}.

Evolution of probabilities $p_{(x,y,z)}$ in the modified KECCAK variant

Figure 6 shows how probabilities $p_{(x,y,z)}$ change in the modified KECCAK variant (without ι). The variant was used in 4-round preimage attack. Please note that in the 4th round, after θ, there are still $p_{(x,y,z)} = 0$ or $p_{(x,y,z)} = 1$ which is the key observation in the 4-round preimage attack.

Fig. 6. Evolution of probabilities $p_{(x,y,z)}$ in the modified KECCAK variant.