Differential Biases in Reduced-Round Keccak

Sourav Das and Willi Meier

Alcatel-Lucent India Ltd and FHNW, Windisch, Switzerland
Email: sourav10101976@gmail.com

Abstract. The Keccak hash function is the winner of the SHA-3 competition. In this paper, we examine differential propagation properties of Keccak constituent functions. We discover that low-weight differentials produce a number of biased and fixed difference bits in the state after two rounds and provide a theoretical explanation for the existence of such a bias. We also describe several other propagation properties of Keccak with respect to differential cryptanalysis. Combining our propagation analysis with results from the existing literature we find distinguishers on six rounds of the Keccak hash function with complexity 2^{52} for the first time in this paper.

Keywords: SHA-3, Propagation Analysis, Double-kernel, TDA

1 Introduction

Cryptographic hash functions are important tools in cryptography that can be used for multiple purposes including authentication, integrity check of executables, digital signatures, etc. These are deterministic functions, H, that given an input or message of an arbitrary length, M, return a short pseudo-random value of fixed length, n. A hash function should be easy to compute and is usually defined by an iterative construction and a compression function.

The classical security requirements of a hash function are:

1. Collision resistance: finding two message $M1$ and $M2$ so that $H(M1) = H(M2)$ must be "hard". The minimum complexity of such an attack is given by birthday bound which is $2^{n/2}$ calls to the compression function.
2. Second preimage resistance: Given a message $M1$ and its hash value $h = H(M1)$, finding another message $M2$ so that $H(M2) = h$ must be hard. The generic second preimage attack has a complexity of 2^n calls to the compression function.
3. Preimage resistance: Given a hash value h, finding a message $M1$ so that $H(M1) = h$ must be hard. The generic preimage attack requires 2^n calls to the compression function.

In a practical sense, finding a collision or a (second) preimage of the hash function should require not significantly less calls to the compression function than the generic attacks. Besides these properties, a hash function is expected to satisfy a few other conditions, e.g, the hash output should not be distinguishable from random output.

In the last decade, many of the standardized hash functions e.g. MD5 [19], SHA1 [16] have suffered from serious collision attacks [21], [20]. The confidence in the standard SHA-2 has then been put into question due to its resemblance with SHA-1. As a consequence, the American National Institute of Standards and Technology (NIST) has launched in 2008 a competition to find a new hash function standard, SHA-3. From the 64 initial submissions, two rounds and four years later, Keccak emerged as the winner in 2012.

Keccak is a sponge based hash function. One of the stated reasons for the selection of Keccak by NIST as the new SHA-3 hash standard was its exceptional resistance to cryptanalytic attacks [7]. Even though it was a prime target for several years and a lot of cryptanalytic effort went into it [1], [5], [3], [8], [12], [4], [10], [11], [9], [13], [17], there was limited progress so far in mounting an attack even in greatly simplified versions of its various flavors. In particular, the best known results so far have been successful only up to five rounds of Keccak. One of the main reasons for this lack of progress is that the probabilities of the standard differential characteristics of Keccak's internal permutation are quite small, as was rigorously shown in [8].

This paper investigates on the propagation properties of certain low-weight differentials in Keccak. It discovers that there exist biased differential bits in the output part of Keccak after two rounds. It traces back the existence of these biased bits to the quadratic non-linear function, χ, of Keccak. There also exist state bits with fixed difference with probability 1 after two rounds. The existence of fixed difference bits with probability 1 after two rounds was reported in [14], [15], without giving further explanation. As a consequence of the propagation analysis as described in this paper we not only provide a clear explanation for the existence of such bits, but also discover many bits with strongly biased differences. Further, this paper extends the propagation properties to four rounds of Keccak by using a *double-kernel*, a concept studied in [17], extending from the concept of a *kernel* of differences in [4]. We exhaustively search all possible double-kernels upto weight six in Keccak permutation and discover some interesting properties of double-kernels.

As a main application, we show that some of our four-round distinguishers enable an extension to six-round distinguishers for Keccak-224. The complexity of the resulting distinguishers amounts to the evaluation of the hash function for 2^{52} message pairs. This uses computation backwards for two rounds and is based on results published in [9] and [10], where a heuristic algorithm is designed (called Target Difference Algorithm - TDA), that allows to find message pairs which satisfy a given target difference after one Keccak permutation round.

In [17] a very efficient distinguisher on four rounds of the Keccak hash function is described. This uses the concept of free bits combined with well chosen messages, and is not available for backwards computation over two rounds, to provide a distinguisher over six rounds. Also, there are previous results known on distinguishers of the Keccak permutation, that enter more than six rounds: Zero-sum distinguishers [1], can distinguish the full 24-round internal permutation from a random permutation [5], [6], [12], and in [13], a differential distinguisher on eight rounds of the internal permutation is described. However these distinguishers are not easily amenable to provide distinguishers for the hash function. This paper provides a six round round distinguisher on Keccak hash function i.e. it considers the initial capacity bits set to zero. The Table 1 provides the best results of cryptanalysis of Keccak hash function.

Table 1. The best cryptanalytic results on Keccak on the *hash function* settings

Variant	No of Rounds	Complexity	Type of Attack	Reference
512, 384, 256, 224	4	Improvement by 2^3 - 2^8	preimage	[15]
224,256	4	practical	collision	[10]
224,256	5	practical	near-collision	[10]
256	5	2^{115}	collision	[11]
384, 512	3	practical	collision	[11]
224, 256	4	2^{24}	distinguisher	[17]
224	6	2^{52}	distinguisher	This Paper

As a final application of our propagation analysis, we provide an explanation why the near collision on five rounds of Keccak as reported in [9], [10], does not match with the expected near collision predicted by their differential trail.

This paper is organized as follows. Section 2 describes the Keccak hash function. The propagation properties of Keccak sponge constituent functions are given in Section 3. Section 4 shows how to get 2 rounds distinguishers using the propagation properties. Section 5 gives the application of the propagation properties to get biased difference bits upto six rounds of Keccak and thereby get a six round distinguisher. Finally, Section 6 shows another application of these propagation properties.

2 Description of Keccak

Keccak is a family of sponge hash functions. A sponge hash function absorbs a message block of r bits into its internal state and subsequently applies an internal permutation to the state. This step is repeated until all the blocks of the message to hash have been treated. Next, in the squeezing phase, r bits are generated from the state before each new permutation application, until the number of desired output bits has been generated. In the following we recall the recommended Keccak versions for SHA-3. All versions use the same internal permutation: $Keccakf[1600]$.

The $Keccakf[1600]$ state consists of 1600 bits, organized in 64 *slices* of 5×5 bits. The position of a bit in a slice can be given by its x and y value. The z coordinate gives the number of the slice $0 \leq z \leq 63$. Most of the steps in the round function of Keccak are invariant to a translation in z direction. The only part non-invariant is the round constant addition ι.

The permutation of the full Keccak hash function consists of 24 iterations of the round function. The round function itself is composed of five steps:

1. θ: Xor to each bit the XOR of two *columns* (column = same x value, y from 0 to 4). The first column is in the same slice as the bit and the second column is in the slice before the bit.
2. ρ: Translate a bit in z direction.
3. π: Permute the bits within a slice.
4. χ: Apply a 5×5 S-box on one *row* (row = same y value, x from 0 to 4).
5. ι: Addition of round constant.

Each of the versions (224, 256, 384 and 512 output bits) has a different block message size r. The capacity in a sponge construction is the size of the internal state minus the size of a message block. Consequently, they all have a different capacity c:

– For an output of 224 bits, r = 1152 and c = 448.
– For an output of 256 bits, r = 1088 and c = 512.
– For an output of 384 bits, r = 832 and c = 768.
– For an output of 512 bits, r = 576 and c = 1024.

In this paper, we have represented the state as a 5×5 matrix where each element of the matrix is a 64-bit *lane* (lane= same x and y value, z ranging from 0 to 63). In the lane, the LSB is at the right side i.e. it is in Little Endian notation.

3 Propagation properties of the Keccak constituent functions

In this section, we consider some propagation properties of the constituent functions θ, ρ, π, χ and ι. In propagation analysis, we perform a large number of experiments and observe the state output differences. If the state difference is always 1, then we call that state difference bit as *active*. If it is always 0, then we call that difference bit as *inactive*. If the state difference is 0 for half the times (and 1 for half the times), then we call that state difference bit as *balanced*.

3.1 Propagation properties of θ, ρ, π and ι

The θ transformation on each state bit depends linearly on 10 other state bits and the self bit.

Property 1. If an even number of bits of the input of θ are active (i.e. change their value with probability 1), and the remaining input bits are inactive (i.e. stay constant), then the output state bit is inactive.

Similarly, if an odd number of bits of the input of θ are active, and the remaining input bits are inactive, then the output state bit is active.

Property 2. If any bit of the input state of θ is balanced then the output state bit is also balanced.

Property 3. The functions ρ, π and ι, being simple bitwise permutations, do not change any propagation properties of the input.

3.2 Propagation properties of χ

The S-box of the χ layer is a 5-bit S-box, where every output bit y_i depends on only three input bits. Out of these, it is dependent linearly on one bit and non-linearly on another two bits (operations are in $GF(2)$):

$$y_i = x_i + (x_{i+1} + 1) \cdot x_{i+2} \tag{1}$$

Assume a set of input differences causes a set of output differences. We describe how the input set affects the output set for a particular bit.

Property 4. If x_i is active and there is no difference in any of x_{i+1} and x_{i+2}, then the output bit, y_i, is active.

Property 5. If there is a difference either in x_{i+1} or x_{i+2} or in both, then the output is balanced.

If there is a difference in one of these inputs, the output difference will depend on the value of the other non-linear bit. If the value of the other bit is zero, then there is no difference in the output, else if it is 1, then there is a difference. Assume a difference in both inputs. If the input values for $\sim (x_{i+1})$ and x_{i+2} are 00 (11), the other input values are 11 (00), hence the product difference is 1 for these two cases. If the input values of these two bits are 01 (10), then the other values are 10 (01), hence the product difference is 0.

Next we consider the properties when some input differences are balanced.

Property 6. If the difference in x_i is balanced, then the output difference is also balanced (independently of the propagation properties of the other input variables).

Now consider the following two cases with fixed difference in x_i.

Property 7. If either the difference in x_{i+1} or x_{i+2} is balanced, and the other difference is fixed, then the output difference is biased with probability 3/4.

Property 8. If differences in both x_{i+1} and x_{i+2} are balanced, then the output difference is biased with probability 5/8.

If differences in both x_{i+1} and x_{i+2} are balanced, the following are the possibilities about their input differences:

00 implies difference zero for 25 percent cases.

01, 10, 11 implies the output is balanced for $25 \times 3 = 75$ percent cases.

So the output difference is zero for (25+75/2)/100=125/200=5/8 cases.

Next, we consider some properties from the difference distribution table (DDT) of the S-box of Keccak. The DDT of the Keccac S-box can be found in Appendix B, Table 7 of [13]. We can see an important property as below.

Property 9. In the DDT of the Keccac S-box, every bit is either fixed or balanced.

This follows immediately from the fact that χ is a quadratic function. The difference function for each output bit is linear for any given input difference in χ and is thus either fixed or balanced. For example, when the input difference is $0x01$ (which is used maximally in our analysis), the possible output differences are $0x01$, $0x09$, $0x11$ and $0x19$ each having occurrences of exactly eight times. One can observe that the output difference is always 1 in the 0^{th} bit; it is always 0 on 1^{st} and 2^{nd} bits; and, always balanced on the 3^{rd} and the 4^{th} bits. This property can be easily proven for every case using the boolean equation of the S-box i.e. Equation 1.

4 Application of Propagation Properties: Two Rounds Distinguishers

In this section, we show the first application of the propagation properties as described in the previous section. First we present how to calculate the distinguishers using the active, inactive and the biased bits. The balanced bits seem of no assistance, as they don't remain exactly balanced bits after few rounds.

4.1 Method for Finding Distinguishers

The above properties suggest that there will be fixed and balanced difference bits after a round when we start with a low-weight differential (refer Properties 4, 5, 6). After another round, the balanced difference bits and the fixed difference bits will give rise to biased difference bits (refer Properties 7 and 8). It is evident from all the properties of χ functions that the there are five possibilities of the differential state bits after performing a large number of experiments, namely, always active (m), always inactive (n), biased towards zero or one in 75 percent trials (q), biased towards zero or one in 62.5 percent trials (r) and unbiased (s). We show below how we construct a favorable event for a distinguisher. A suitable favorable event for our distinguisher might be as follows:

1. We have $m+n$ deterministic bits. We require these to have the right value.
2. We have a number q of bits with probability about 0.75 to be either 0 or 1. Then the expected value of bits which take the value as predicted by the bias is $q \cdot p = 0.75 \cdot q$.
3. We have r bits to be either 0 or 1 with prob. about 0.625. Then the expected value of correct bits is $0.625 \cdot r$.

Thus our test first asks if the $(m+n)$ deterministic bits are correct. If no, ignore the rest. If yes, count the numbers of correct bits under the previous conditions 2. and 3. If one or both of these numbers are at least $k_1 = 0.75 \cdot q$ or $k_2 = 0.625 \cdot r$, respectively, we have a desirable event.

According to Binomial distribution, the probability for at least k successes in n trials having a probability p is:

$$P(X \geq k) = \Sigma_{i=k}^{n} C_i^n p^i (1-p)^{n-i} \tag{2}$$

This can be approximated by a normal distribution. For this we make use of the following formulae for n trials with probability p:
Mean=$\mu = np$, Standard deviation=$\sigma = \sqrt{np(1-p)}$.

Convert the binomial variable (X) to normal variable by: $Z = (X - \mu)/\sigma$.

Finally, calculate the probability of the event, $P((np - \mu)/\sigma < Z \leq (n - \mu)/\sigma) = P(0 < Z \leq (n - \mu)/\sigma)$ (since, $np = \mu$), from the normal distribution table.

Clearly, if we require all the bits are correct, the probability becomes p^n. We look for a favourable event that is able to distinguish later the hash output from random with a good tradeoff between distinguishing property and number of samples. An example of such a favourable event is given below.

Example 1. Assume that in a large number of experiments, we have got 16 deterministic difference bits (i.e. totally active or inactive bits), 35 bits with a differential bias 0.75 and 31 bits with a differential bias 0.625. Consider now an event to be favorable if all 16 deterministic bits are correct, and at the same time all 35 bits with bias 0.75 are correct. Moreover require that for the bits with bias 0.625, at least $\mu = np = 31 \times 0.625 = 19.375$ bits are correct.

Then, for the 16 deterministic bits and the 35 bits with the bias 0.75, the probability of success for the random case is, $2^{-16-35} = 2^{-51}$, whereas for the biased case is $1 \times 0.75^{35} \approx 2^{-15}$.

According to normal approximation, the probability that at least $\mu = 19.375$ of the 31 bits with a differential bias of 0.625 are correct is 1/2. If the bits were random, then $\mu = 31 \times .5 = 15.5$ and $\sigma = 2.78$. In that case, the probability of the same number of bits to be correct were: $P(((19.375 - 15.5)/2.78) < Z \leq ((31 - 15.5)/2.78)) = P(1.39 < Z \leq 5.57)$. From a table, $P = 1 - 0.91774 = 0.08226 = 2^{-3.6}$.

Hence, the total probability for the biased case is $2^{-15-1} = 2^{-16}$ whereas for the random case it is $2^{-51-3.6} = 2^{-54.6}$. So, the total probability for the event in the biased case is almost 2^{39} times higher than the random case. This gives a strong distinguishing property for this event.

□

4.2 Two Rounds Distinguishers for Low Weight Differentials

In this section, it will be shown that low-weight differentials can provide distinguishers for two rounds. Let us consider some low weight differentials of weight w on the message part of the hash function. Now let us see how the differences propagate through different constituent functions of Keccak.

θ **Function of Round 1:** On the θ function of the first round, the differences of weight w, will spread the differences to roughly $11w$ bits if the differences are sparse enough. Hence, on a large number of experiments, the state difference remains fixed after the θ function of the first round.

ρ**,** π **and** ι **Functions of Round 1:** These functions keep the weight of the differences.

χ **Function of Round 1:** The χ function will keep the difference fixed (either 0 or 1) for some bits, and make it balanced for some other bits, as follows from Properties 4, 5, 6 and 9.

θ **Function of Round 2:** The balanced bit differences produced in the χ layer in the first round will make 11 bits per balanced bits of the state as balanced. On the other hand, the fixed differences (of value 1) will make 11 bits per state bit (excluding the balanced bits) as fixed differences (of value 1). If all the 11 bits involved in the θ transformation of a particular state bit have no differences (i.e. fixed difference 0) then, that particular state bit difference will be always 0. Since, at the end of round 1, the number of balanced state bits is still small, there are still a lot of fixed difference bits at the end of θ transformation of Round 2.

ρ**,** π **and** ι **Functions of Round 2:** These functions keep the weight of the differences.

χ **Function of Round 2:** At this point, the χ function has three types of input state differences. The first one is always 0, the second one is always 1 and the third one is balanced in a large number of experiments. By virtue of the Properties 4 to 9, it produces fixed output differences, balanced output differences and biased output differences in a large number of experiments.

Thus, at the end of Round 2, we still get fixed output differences and biased output differences in the state. If we get some of these conditions in the output part of the hash function, then we will have distinguishers. Now there are special techniques already developed in literature to have low-weight differences up to a certain number of rounds. Using those techniques combined with the propagation analysis shown so far, we will show how to get distinguishers for 3, 4, 5 and 6 rounds in the rest of the paper.

Note that we can expect biased bits starting with a low weight differential only after two rounds but not after one round. Further as we shall see, we cannot expect biased bits after three rounds.

Now, we are able to get biased difference bits after two rounds of Keccak, provided the input difference has relatively small weight. Let us try to quantify how much small i.e. what is the maximum weight that can give rise to the fixed/biased bits after the second round. The χ layer is the one that creates the balanced bits at round 1 and θ layer is the one which spreads the balanced bits at round 2. Let w_{max} be the maximum weight that will produce fixed bits and/or biased bits after round 2. After θ layer, this weight will spread to $11w_{max}$ state bits. In χ layer, every fixed difference bit makes two more bits balanced (by extension of Property 5). Hence after the first χ layer, $22w_{max}$ bits become balanced. After the second θ layer, the total number of bits that becomes balanced are $11 \times 22w_{max} = 242w_{max}$. Again in χ layer, if an input difference bit is balanced, the corresponding output difference bit is also balanced (by property 6). In order that all the 1600 state bits do not become balanced after θ layer of round 2, the maximum value of w_{max} can be only $1600/242 \approx 6$.

In our case, we have always started with a differential of weight 6, and indeed we have got some fixed or biased bits in the output after two rounds. The above heuristics assume that the weight is evenly distributed in the state. However, this is not the case always and some difference bits are clustered (θ and χ, tend to cluster but ρ and π attempt to distribute them evenly). This clustering gives us some more fixed/biased difference bits after round 2. But, most of the bits after round 2 are balanced, hence the θ operation and the χ operation in round 3 spreads the balanced bits to all the state bits. Thus we don't get any more fixed or biased bits after round 3.

4.3 The Effect of ι in Differential Bias Propagation

The round constant, ι, of the last round will not have any effect on the differentials. This is because the round constant is simply XORed at the end and the differences will cancel it out. However, in our biased differential propagation analysis, we have passed through one extra round by value before calculating the final differential for our propagation analysis. Hence, in our method of calculating differential bias propagation, the ι constant will matter for the last but one round. It will not matter for the last round. However, the round constants are different for different rounds and our differential bias propagation analysis are performed in the last two rounds. So, when we consider the differential bias propagation analysis for Keccak permutation for two, three and four rounds, we should consider the ι constants for first, second and third round, respectively. But, when we consider the distinguishers for Keccak hash functions later, we will prepend two more rounds. In that case, we have to consider the right ι constant accordingly (i.e. ι of round 5, for a six round distinguisher) in the subsequent sections.

5 Distinguishers of Six Rounds of Keccak

In this section, we proceed step by step to extend the distinguishers of two rounds using the propagation properties, to three, four and six rounds distinguishers. We extend the two rounds distinguishers for three rounds using the concept of *kernels*. A *kernel* is, as defined by Keccak authors, a state difference with even number of differences in each column. When the state difference is in a kernel, the θ transformation on that state differences will not change it. We further extend this for four rounds by using a *double-kernel*. A *double kernel*, as considered in [17], is a differential path where the state differences of two successive rounds are in a kernel. As a main application, a few of the distinguishers as found for four rounds will be extended to the hash function when reduced to six rounds by using *target-difference-algorithm (TDA)*. *TDA*, as defined in [10], allows to find message pairs which satisfy a given target difference after one Keccak permutation round.

5.1 Extension to 3 Rounds - Starting with a Kernel

We can easily extend the above distinguishers to 3 rounds by starting with a Kernel. If we start with a low difference kernel with the differences in the message part of the hash function, the θ, ρ, π will maintain the same difference; then it is already well-known that the χ layer will maintain this sparse difference with a probability 2^{-2} per difference bit [17]. If there are r difference bits, then the total probability of maintaining the same difference is 2^{-2r}. Thus, at the end of round 1, we get a low-weight difference with a probability 2^{-2r}. This can be now the starting point for the distinguisher of two rounds using our method. Hence, we can easily get three rounds distinguishers using this method with an additional complexity of 2^{-2r} ($r \leq 6$).

5.2 Extension to 4 Rounds - Starting with a Double Kernel

Most of the successful differential paths on various rounds have made use of the double kernels to increase the number of rounds. We also immediately increase the number of rounds for the double kernel. We have used some of the existing two round trails to find distinguishers up to four rounds.

We start with the double kernel that was found in [17]. The double kernel defined by them is shown in Table 2. Starting with this differential, we have performed 2^{20} experiments[1] with two more rounds and checked the output part after four rounds for Keccak-224, Keccak-256, Keccak-384 and Keccak-512 to see if there are biased bits. We have found many bits which are fixed (i.e. always active or always inactive) and many bits which are biased. The results are shown in Table 3 where the entries with 0.75 means that for those many bits, the number of zeros (or ones) occurred 75 or 25 percent times and the entries with 0.625 means that for those many bits, the number of zeros (or ones) occurred 62.5 or 37.5 percent times. These cases represent the biased bits. We can easily employ the methods given in Section 4 (see Example 1) to determine the distinguishers. Note that in [17] already a very efficient distinguisher on the Keccak hash function reduced to four rounds is described. This is based however on a trick using free bits and selecting appropriate messages that is not available when we later prepend two more rounds.

Table 2. The differential path of [17]

	---------------1	8---------------	----------------	----------------	----------------
	---------------1	----------------	--------4-------	----------------	----------------
δ_0	----------------	8---------------	--------4-------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
	---------------1	----------------	--------------2-	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
δ_1	---------------1	------1---------	----------------	----------------	----------------
	----------------	------1---------	--------------2-	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
	---------------1	----------------	----------------	----------------	----------------
	----------------	----------------	---------------8	-----------2----	----------------
δ_2	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----4-----------	----------1-----	----------------
	--------------8-	----------------	----------------	----------------	----------------

[1] We started our analysis with 2^{20} experiments; but later on we found that we can clearly distinguish the biased and fixed bits with 10000 or even 1000 experiments. As computational time difference between 10000 and 1000 experiments was not significant enough, in the subsequent sections we have performed experimentation with 10000 samples.

Table 3. The biased difference bits after 4-rounds using the differential path of [17]

Variant	Active	Inactive	0.75	0.625
224	7	9	35	31
256	7	11	45	34
384	7	17	77	59
512	10	20	100	73

5.3 Differential Distinguishers with all Possible Double Kernels up to Weight Six

After finding deterministic and biased bits, we investigated towards finding the best possible results by looking into all possible double kernels upto weight six. As shown in the previous section, we get two rounds distinguishers using the propagation properties only when we start with a differential of weight upto six. Since we start with a kernel with three non-zero columns, each with weight 2, the computational complexity to find all the double kernels is reasonable. The computational complexity to find all the double kernels is: *(all possible combinations of two bits in three different columns)* $\times$ *(all possible combinations of three columns in 320 columns)* $= (C^5_2)^3 \times (C^{320}_3) = 5.4 \times 10^9$ ρ and π operations. We found that there exists an "equivalent class" of double kernels. Note that a kernel becomes a double kernel only when a specific permutation of ρ and π rearranges them to another kernel. Now if the difference bits in the original kernel are shifted along the z-direction equally, then ρ will shift those difference bits equally. The lane rearrangement done by the π function will keep the state again in a kernel. Hence, the following property of double kernel can easily be observed.

Property 10. For a low-weight kernel which results into a double-kernel, if the difference bits are equally shifted along the z-axis, then the new kernel also results in a double kernel.

Since there are 64 bits in a lane, the following property follows:

Property 11. The total number of double kernels must be divisible by 64.

Finally, each of these shifted double kernels will provide almost same but shifted propagation properties for the state bits. That means there will be similar number of active, inactive and biased bits in the state after four rounds for each of these double kernels. However, it is not exactly same because of the effect of ι in the last but one round (see Section 4.3). Hence, we give the following propagation property of a double kernel:

Property 12. There exists an equivalent class, with respect to the propagation properties of two additional rounds, of double kernels consisting of 64 members which will give rise to the almost same number of biased differential bits (active, inactive, biased) in the state after four rounds.

We have done an exhaustive search of double kernels and found a total of 512 double kernels in Keccak. Hence, there are a total of eight equivalent classes of double kernels. The double kernels for all the eight cases are given in the Table 4. For each of the double kernels of Table 3, the number of active bits, inactive

Table 4. The equivalent classes of the double-kernels

Sl No	δ_i	Differentials				
1	δ_0	- - - - - - - - - - - - - - - 1 - - - - - - - - - - - - - - - 1 -	8 - 8 -	- 4 - - - - - - - - - - - - - - - 4 -	- -	- -
1	δ_1	- - - - - - - - - - - - - - - 1 - 1 -	- 1 - - - - - - - - - - - - - - - 1 -	- - - - - - - - - - - - - 2 - 2 - - - - - - - - - - - - - - - - - -	- -	- -
2	δ_0	- - - - - - - - - - - - - - - 1 - - - - - - - - - - - - - - - 1 -	- -	- 4 - 4 -	- -	- - - - - - 2 - 2 -
2	δ_1	- - - - - - - - - - - - - - - 1 - 1 - - - - - - - - - - - - - - - -	- 1 - - - - - - - - - - - - - - - 1 -	- -	- 2 - - - - - - - - - - - - - - - 2 -	- -
3	δ_0	- - - - - - - - - - - - - - - 1 - 1 - - - - - - - - - - - - - - - -	- -	- - - - - - - - - - - - - - - 4 - 4 -	- 1 - - - - - - - - - - - - - - - 1 -	- -
3	δ_1	- - - - - - - - - - - - - - - 1 - 1	- -	- - - - 2 - 2 -	- - - - - 2 - 2 - - - - - - - - - -	- -
4	δ_0	- - - - - - - - - - - - - - - 1 - 1	- -	- 2 - 2 - - - - -	- - - - - - 1 - - - - - - - - - - - - - - - 1 -	- -
4	δ_1	- - - - - - - - - - - - - - - 1 - - - - - - - - - - - - - - - 1 -	- 8 - 8 - - - - - -	- -	- -	- 4 - - - - - - - - - - - - - - - 4 -
5	δ_0	- 1 - 1 - - - - - - - - - - - - - - - -	- 1 - - - - - - - - - - - - - - - 1 -	- -	- 2 - - - - - - - - - - - - - - - 2 -	- -
5	δ_1	- -	- 1 - - - - - - - - - - - - - - - 1 - - - - - - - - -	- 4 - - - - - - - - - - - - - - - 4 - - - - - - - - - - - - - - - - -	- 2 - 2 - - - - - - - - - -	- -
6	δ_0	- 1 - 1	- -	- 2 - - - - - - - - - - - - - - - 2 - - - - -	- -	- 1 - - - - - - - - - - - - - - - 1 -
6	δ_1	- -	- -	- 8 - 8	- 1 - - - - - - - - - - - - - - - 1 -	- 4 - - - - - - - - - - - - - - - 4 -
7	δ_0	- -	- - - - - - - - - - - - - - - 1 - 1 -	- 8 - - - - - - - - - - - - - - - 8 -	- -	- - - - - - 4 - 4 -
7	δ_1	- 2 - - - - - - - - - - - - - - - 2 - - - - - - - - - - - - - - - -	- -	- - - - - - - - - - - - - 4 - 4 - - - - - - - - - - - - - - - - - -	- 4 - - - - - - - - - - - - - - - 4 -	- -
8	δ_0	- -	- -	- - - - - - - - - - - - - - - 1 - 1	- - - - - - - 4 - 4 -	- 8 - - - - - - - - - - - - - - - 8 - - - - - - - - - - -
8	δ_1	- - - - - - - - - - - - - - - - 4 - 4 - - - - - - - - - - - - - - -	- -	- -	- - 8 - 8 -	2 - - - - - - - - - - - - - - - 2 -

Table 5. The differential biased bits after 4-rounds for all double kernels

Sl.	Variant	Active	Inactive	0.75	$\mu_{.75}$	$\sigma_{.75}$	$Z_{.75}$	.25	$\mu_{.25}$	$\sigma_{.25}$	$Z_{.25}$	0.625	$\mu_{.625}$	$\sigma_{.625}$	$Z_{.625}$	.375	$\mu_{.375}$	$\sigma_{.375}$	$Z_{.375}$
1	224	7	9	9	6.75	1.30	1.73	26	6.50	2.21	8.83	7	4.38	1.28	2.05	24	9.00	2.37	6.32
1	256	7	11	10	7.50	1.37	1.83	35	2.50	2.56	12.69	7	4.38	1.28	2.05	27	10.13	2.52	6.71
1	384	7	17	19	14.25	1.89	2.52	58	4.75	3.30	16.15	13	8.13	1.75	2.79	46	17.25	3.28	8.76
1	512	10	20	20	15.00	1.94	2.58	80	5.00	3.87	19.36	16	10.00	1.94	3.10	57	21.38	3.66	9.75
2	224	4	11	11	8.25	1.44	1.91	24	6.00	2.12	8.49	11	6.88	1.61	2.57	22	8.25	2.27	6.06
2	256	4	12	11	8.25	1.44	1.91	28	2.75	2.29	11.02	13	8.13	1.75	2.79	25	9.38	2.42	6.45
2	384	4	16	15	11.25	1.68	2.24	45	3.75	2.90	14.20	20	12.50	2.17	3.46	39	14.63	3.02	8.06
2	512	4	19	19	14.25	1.89	2.52	58	4.75	3.30	16.15	22	13.75	2.27	3.63	55	20.63	3.59	9.57
3	224	1	7	8	6.00	1.22	1.63	22	5.50	2.03	8.12	7	4.38	1.28	2.05	22	8.25	2.27	6.06
3	256	1	7	9	6.75	1.30	1.73	28	2.25	2.29	11.24	9	5.63	1.45	2.32	25	9.38	2.42	6.45
3	384	4	13	15	11.25	1.68	2.24	41	3.75	2.77	13.43	21	13.13	2.22	3.55	34	12.75	2.82	7.53
3	512	6	15	24	18.00	2.12	2.83	57	6.00	3.27	15.60	24	15.00	2.37	3.79	51	19.13	3.46	9.22
4	224	0	4	11	8.25	1.44	1.91	20	5.00	1.94	7.75	12	7.50	1.68	2.68	24	9.00	2.37	6.32
4	256	0	5	13	9.75	1.56	2.08	24	3.25	2.12	9.78	14	8.75	1.81	2.90	26	9.75	2.47	6.58
4	384	4	8	19	14.25	1.89	2.52	36	4.75	2.60	12.03	19	11.88	2.11	3.38	44	16.50	3.21	8.56
4	512	5	13	25	18.75	2.17	2.89	49	6.25	3.03	14.10	22	13.75	2.27	3.63	59	22.13	3.72	9.92
5	224	4	4	8	6.00	1.22	1.63	32	8.00	2.45	9.80	5	3.13	1.08	1.73	29	10.88	2.61	6.95
5	256	4	4	11	8.25	1.44	1.91	36	2.75	2.60	12.80	6	3.75	1.19	1.90	33	12.38	2.78	7.42
5	384	8	8	19	14.25	1.89	2.52	53	4.75	3.15	15.31	12	7.50	1.68	2.68	45	16.88	3.25	8.66
5	512	9	13	23	17.25	2.08	2.77	69	5.75	3.60	17.58	22	13.75	2.27	3.63	58	21.75	3.69	9.83
6	224	0	2	12	9.00	1.50	2.00	18	4.50	1.84	7.35	14	8.75	1.81	2.90	27	10.13	2.52	6.71
6	256	0	2	14	10.50	1.62	2.16	20	3.50	1.94	8.52	16	10.00	1.94	3.10	29	10.88	2.61	6.95
6	384	2	4	18	13.50	1.84	2.45	34	4.50	2.52	11.68	27	16.88	2.52	4.02	40	15.00	3.06	8.16
6	512	3	4	23	17.25	2.08	2.77	44	5.75	2.87	13.32	34	21.25	2.82	4.52	54	20.25	3.56	9.49
7	224	3	11	11	8.25	1.44	1.91	26	6.50	2.21	8.83	8	5.00	1.37	2.19	28	10.50	2.56	6.83
7	256	3	11	13	9.75	1.56	2.08	31	3.25	2.41	11.51	10	6.25	1.53	2.45	28	10.50	2.56	6.83
7	384	6	16	21	15.75	1.98	2.65	49	5.25	3.03	14.43	16	10.00	1.94	3.10	39	14.63	3.02	8.06
7	512	8	24	27	20.25	2.25	3.00	66	6.75	3.52	16.84	20	12.50	2.17	3.46	51	19.13	3.46	9.22
8	224	4	1	8	6.00	1.22	1.63	18	4.50	1.84	7.35	10	6.25	1.53	2.45	20	7.50	2.17	5.77
8	256	5	1	9	6.75	1.30	1.73	19	2.25	1.89	8.87	13	8.13	1.75	2.79	24	9.00	2.37	6.32
8	384	7	4	12	9.00	1.50	2.00	27	3.00	2.25	10.67	20	12.50	2.17	3.46	36	13.50	2.90	7.75
8	512	8	4	21	15.75	1.98	2.65	37	5.25	2.63	12.05	24	15.00	2.37	3.79	50	18.75	3.42	9.13

bits, bits with bias 0.25 (towards 0), 0.75, 0.625, 0.375 and their corresponding μ, σ and Z (refer to Section 4) are given in Table 5. Using Table 4 and the normal distribution table, the distinguishers can be easily calculated as outlined in Section 4.

5.4 A Concrete Six Round Distinguisher of Keccak-224

In this section, we extend our work in the previous section to a six round distinguisher by using the results of [9] and [10]. It can be noted that it is easy to go one round backwards in Keccak *permutation* starting from a low-weight differential. The inverse χ layer maps 1-bit difference to a 1-bit difference with a probability 2^{-2}. Hence, going backwards by one round has a complexity of 2^{-12} for a differential with weight 6, as in our case. But, going back one round in the *hash function* is a challenging task. The main challenge is to keep the capacity bits to zero at the beginning of the hash function. The $\theta - inverse$ function of Keccak propagates too many bits even when we start with a low-weight difference. This problem was solved for the first time (heuristically) in [10] and [9]. They called this algorithm to solve this problem as Target Difference Algorithm (TDA). Prepending one more round, on page 14, Section 5.2 of [9], they say that "*For Keccak-224, the algorithm typically returned an affine subspace of message pairs with dimension of about 100 within one minute*" when they go backwards from a differential of double kernel i.e. with weight six. As done by them, we go back one more round directly with a probability of 2^{-12} and then appeal to their findings on TDA to get the hash inputs (and gain one more round). The two rounds given by the double kernels (which occurs with a probability of 2^{-24}) and additional two rounds to get biased differential bits enable us to get a six round distinguisher. This six rounds distinguisher is the result of a combination of TDA [9], double-kernel concept of [17] and the differential bias analysis of this paper.

Now let us calculate the complexity of these distinguishers. The double-kernel has a probability of 2^{-24}. Going backwards by one round will have a complexity of 2^{-12}. Hence, the total complexity of the differential path for this distinguisher is 2^{-36}. To have a distinguisher of six rounds using our propagation analysis, the total probability of a suitably chosen favorable event should be larger than the probability of the event if it were purely random. For the double kernel of [17] which corresponds to the entry with Sl. No. 1 in Table 3 and 4, such a favorable event is shown in Example 1. We found that the probability of the event using our propagation analysis is 2^{-16}. Multiplying by the probability of the characteristic, we get the total probability of this event as $2^{-16-36} = 2^{-52}$, whereas in the random case, the probability is $2^{-54.6}$. Since, the probability of the event is at least four times higher than the random case, we have a distinguisher for this event. The complexity of the distinguisher is about 2^{52}.

Clearly, we can have stronger distinguishers with an additional complexity by repeating the experiment or by considering an event with more bits with probability 0.625 to be correct.

Since the differential path for the six round distinguisher itself has a complexity of 2^{36}, we cannot get any favorable event for some cases in Table 4 and Table 5. But for 3 out of 8 equivalent cases, we can get a favorable event leading to a distinguisher. The cases for which we have a distinguisher are with Sl. No. 1, 2 and 7 in Table 4 and 5.

This distinguisher does not carry over to a six round distinguisher of Keccak-256 without any further refinements (e.g., message modification), as the space of message pairs delivered by TDA as reported in [9] in this case may not always be large enough.

The methods as developed for ordinary differentials carry over to generalized internal differentials as brought up in [11]. However regarding distinguishers they appear to lead to no better results. This can also be extended for special differential characteristics as in [8]. Here again it doesn't appear to lead to better results regarding distinguishers.

6 A Second Application of Propagation Properties

As a final note, please observe that the difference bits in the near collision found in [10],[9] do not follow the difference bits predicted by the trail (see Appendix B of [10],[9]). Table 5 gives the 4-round characteristic that has been used to get near collisions. The near collision they have got are (the difference nibbles are shown in $\underline{\text{underline}}$):
For Keccak-256:

Output1=

407D4466 FEA8B231 EC9$\underline{6}$8181 $\underline{D}$F902165 23C$\underline{2}$19FF $\underline{54}$571D7$\underline{0}$ 2800F5$\underline{0}$6 E81$\underline{8}$6$\underline{44}$B

Output2=

407D4466 FEA8B231 EC9$\underline{2}$8181 $\underline{F}$F902165 23C$\underline{0}$19FF $\underline{1C}$571D7$\underline{4}$ 2800F5$\underline{1}$6 E81$\underline{0}$6$\underline{56}$B

For Keccak-224:

Output1=

85373497 97D871C2 FBD0A823 $\underline{5}$84C0ED4 $\underline{C}$1B$\underline{3}$BF$\underline{4}$F BC4087$\underline{6}$6 0584B08D

Output2=

85373497 97D871C2 FBD0A823 $\underline{7}$84C0ED4 $\underline{E}$1B$\underline{1}$BF$\underline{5}$F BC4087$\underline{7}$6 0584B08D

Note that the output strings given here are with 32 bits whereas the differentials given in Table 5 are 64 bits; both with little-endian format. For Keccak-256, the bits that are different in the near-collision are (starting the bit numbering from 0), 82(0x52), 125(0x7D), 145(0x91), 162(0xA2), 187(0xBB), 190(0xBE), 196(0xC4), 229(0xE5), 232(0xE8), 243(0xF3). For Keccak-224, the positions are, 125 (0x7D), 132(0x84), 145(0x91), 157(0x9D), 164(0xA4). Clearly, the near collision found does not coincide with the difference bits predicted by the characteristic (as given in δ_4 of Table 5). The authors did not give an explanation for that. Here, we give an explanation.

We started with δ_3 of Table 6 and checked the number of difference bits in the output part of Keccak-256 (automatically implying Keccak-224 as well). Table 7 gives the number of times the bits were different in the first 320 bits of the state which includes the output part i.e. the first 256 bits of the state (the rows indicate the first nibble and the columns indicate the second nibble of the output state bit position). Note that the first 320 bits are the first row of the state along the 64-bit lanes. We can observe that even with the differential biases, we can't explain the near collision completely. For example, the state bit at position 0x52 is always zero, but this bit shows up in the near collision of [9] as a difference bit. Now, take a look into the DDT of the S-box as given in [13]. Notice that for a single bit difference, there are only four possible outputs; each happens with the probability 2^{-2}. We found that the actual path taken in δ_3 of Table 6 is not a single bit difference mapping to a single bit difference in all the cases. The difference bit in row 3, column 3 (starting count with 0) of δ_3 of Table 6 actually mapped to a two bit difference in the output. The corresponding δ_3 is shown in Table 7. With this δ_3, we have performed experimentation with 10000 samples with one more round and Table 9 gives the number of output bits that were different in the first 320 state bits. Now we see that state bit position 0x52 is always active. This is the case for all the state bits mentioned above as the difference bits in near collision except the bit positions 0xC4 and 0xF3 where bits were active for 50 percent times. Recall that in Property 5, we have stated that if x_{i+1} is always different, then x_i is balanced in the χ layer. If we look for the next bit along the row for these two bit positions, we can see that both the bit positions (0xC4+0x40=)0x104 and (0xF3+0x40=)0x133 are always active; hence bit positions 0xC4 and 0xF3 are balanced. Hence we can explain the near collision found in [9].

Table 6. The 4-Round Characteristic Leading to Near Collision in [9]

δ_0	BD135E2FA6BD1346 BD135E2FA6BD1346 BD135E2FA6BD1346 BD135E2FA6BD1346 BD135E2FA6BD1346	12D789A92F12D78F 12D789A82F12D78F 12D789AB2F12D78F 12D789A92F12D78F 12D789A92F12D78F	D7E26BC344D7E224 D7E26BC344D7E264 D7E26BC344D7E224 D7E26BC344D7E224 D7E26BC344D7E224	E69AF134B5E69AD5 E69AF134B5E69AD5 E69AF134B5E29AD5 E69AF134B5E69AD5 E29AF134B5E69AD5	98BC4D6BF898BC58 98BC4D6BF898BC58 98BC4D6BF898BC58 98BC4D6BF898BC58 98BC4D6BF898BC58
δ_1	- -	- - - - - - - - - - - - 1 - 1 -	- 8 - - - - - - - - - - - - - - - 8 -	- -	- - - 4 - 4 -
δ_2	- 2 - - - - - - - - - - - - - - - 2 - - - - - - - - - - - - - - - - - - -	- -	- - - - - - - - - - 4 - 4 -	- 4 - - - - - - - - - - - - - - - 4 -	- -
δ_3	- 1 - - - - -	- -	- 1 - - - - - - - - - - - - - - - 8 -	- - - - - - - - - - - - 8 - 2 - - - - - - - - - - - 4 - - - - - - - - - - - - -	- -
δ_4	- 9 8 - 4 - - - - - - - 1 - 4 - - - - - 2 - - - 1 - - 2 - - - 1 - - - - - 4 - - -	2 - - - - - - - - - - - - - - - - 2 - - - 2 - 8 - - - - - 4 - - 2 - - - 1 - 4 8 -	4 8 - - - - - 4 - - - 2 - 1 2 - - - - - - - - - - - - - - - - - - - 8 - - - - - - 1 - 4 - - - - - 2 - - - 1 - - -	- 4 - - - 1 2 - - - - - - - - - 4 - - - - - - - - - - - - - - - 4 - - - 2 - - - 2 - 8 - - - - - - - 2 - - - - - 8 - - - - - 4 - - - - - - - - - - - - - - - - -	- - - 8 - - - 8 2 - - - - - 1 - - - 1 - - - - 8 - - - - 2 - - - - - - - - - 4 - - - - - - - - - - - - - 4 - - - - - - - - 9- - - - - - - - - - - - - 8 - - - -

Table 7. The Difference Distribution in the first 320 bits after Extending One Round from δ_3 of the Table 6

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	5039	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	5096	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	5019	0	4984	4982	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	4982	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	5005	0	0	4953	0	0	4993	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	4942	4978	0	10000	4996	0
8	0	0	0	0	5089	0	0	0	0	0	0	0	0	0	0	0
9	0	10000	0	0	0	0	0	0	0	0	0	0	0	5011	0	0
A	0	0	10000	4925	0	5030	0	0	4971	0	0	0	0	0	0	0
B	0	0	0	5006	0	0	0	0	0	0	5068	10000	0	0	10000	0
C	0	0	0	0	5055	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	4977	0	0
E	0	0	0	4984	0	10000	0	0	10000	0	0	0	0	0	0	0
F	0	0	0	4935	0	0	0	0	0	0	10000	0	0	0	0	0
10	0	0	0	0	10000	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	10000	0	0
12	0	0	0	10000	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	10000	0	0	0	0	0	0	0	0	0	5024	0	0

Table 8. The δ_3 of Table 6 with the difference bit in row 3, column 4 mapping to 11 in χ layer

δ_3	- 1 - - - - -	- -	- 1 - - - - - - - - - - - - - - - 8 - - - - - - - - - - 2 -	- - - - - - - - - - - - 8 - 2 - - - - - - - - - - - 4 - - - - - - - - - - - - -	- -

Table 9. The Difference Distribution in the first 320 bits after Extending One Round from δ_3 of the Table 8

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	5039	5027	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	5096	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	5019	0	4984	4982	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	4982	10000	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	5005	0	0	4953	0	0	4993	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	4978	0	10000	4996	0
8	0	0	0	0	5089	0	0	0	0	0	0	0	0	0	0	0
9	0	10000	0	0	0	0	0	0	0	0	0	0	0	5011	0	0
A	0	0	10000	4925	0	5030	0	0	4971	0	0	0	0	0	0	0
B	0	0	0	5006	0	0	0	0	0	0	0	10000	0	0	10000	0
C	0	0	0	0	5055	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	4977	0	0
E	0	0	0	4984	0	10000	0	0	10000	0	0	0	0	0	0	0
F	0	0	0	4935	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	10000	0	0	0	0	0	0	0	0	0	0	0
11	0	0	5111	0	0	0	0	0	0	0	0	0	0	10000	0	0
12	0	0	0	10000	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	10000	0	0	0	0	0	0	0	0	0	5024	0	0

7 Conclusion

We have analysed the propagation properties of Keccak constituent functions. For low weight input differences this enables to derive a number of fixed or strongly biased difference bits after two rounds. Combined with the concept of double kernel [17] this leads to several differential distinguishers over four rounds of Keccak. Some of these distinguishers are flexible enough to be extended via the TDA algorithm in [10] to efficient differential distinguishers of the Keccak hash function when reduced to six rounds, despite the quite low probabilities of individual characteristics. We have discovered a few properties of Keccak that contribute to a better understanding of this hash function. The results found in this paper pose no threat to the security of full round Keccak.

Acknowledgement: We are grateful to the anonymous reviewers for their comments that have helped to improve the presentation of this paper.

References

1. J.P. Aumasson and W. Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. NIST mailing list, 2009.
2. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In CCS,Proceedings of the 1st ACM conference on Computer and communications security, pp. 62-73. ACM, 1993.
3. D. J. Bernstein. Second preimages for 6 (7? (8??)) rounds of keccak? NIST mailing list, 2010.

4. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The Keccak SHA-3 submission. Submission to NIST (Round 3), 2011.
5. C. Boura and A. Canteaut. Zero-Sum Distinguishers for Iterated Permutations and Application to Keccak-f and Hamsi-256. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, Selected Areas in Cryptography, volume 6544 of Lecture Notes in Computer Science, pp. 1–17. Springer, 2010.
6. C. Boura, A. Canteaut, and C. De Cannière. Higher Order Differential Properties of Keccak and Luffa. In Antoine Joux, editor, Fast Software Encryption - 2011. LNCS vol. 6733. pp. 252–269. 2011.
7. S. Chang, R. Perlner, W. E. Burr, M. S. Turan, J. M. Kelsey, S. Paul, and L. E. Bassham. Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition. http://csrc.nist.gov/groups/ST/hash/sha-3/Round3/documents/Round3 Report NISTIR 7896.pdf, 2012.
8. J. Daemen and G. Van Assche. Differential Propagation Analysis of Keccak. Fast Software Encryption - 2012. LNCS vol. 7549. pp. 422–441. 2012.
9. I. Dinur, O. Dunkelman, A. Shamir. Improved Practical Attacks on Round-Reduced Keccak. To appear, Journal of Cryptology.
10. I. Dinur, O. Dunkelman, A. Shamir. New Attacks on Keccak-224 and Keccak-256. FSE - 2012. LNCS vol. 7549, pp. 442–461. 2012.
11. I. Dinur, O. Dunkelman, A. Shamir. Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials. FSE - 2013. LNCS vol. , pp. . 2013.
12. M. Duan and X. Lai. Improved Zero-Sum Distinguisher for Full Round Keccak-f Permutation. Cryptology ePrint Archive, Report 2011/023, 2011.
13. A. Duc. et al. Unaligned Rebound Attack – Application to Keccak. Available at http://eprint.iacr.org/2011/420.
14. P. Morawiecki, J. Pieprzyk, M. Srebrny, and M. Straus. Preimage attacks on the round-reduced Keccak with the aid of differential cryptanalysis.Cryptology ePrint Archive, http://eprint.iacr.org/2013/561.pdf.
15. P. Morawiecki, J. Pieprzyk, M. Srebrny. Rotational cryptanalysis of round-reduced Keccak.In FSE, 2013. Available at: http://eprint.iacr.org/2012/546.pdf.
16. National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard. http://csrc.nist.gov, April 1995.
17. María Naya-Plasencia, Andrea Röck, and Willi Meier. Practical Analysis of Reduced-Round Keccak. In Daniel J. Bernstein and Sanjit Chatterjee, editors, Progress in Cryptology - INDOCRYPT 2011, volume 7107 of LNCS. Springer, Heidelberg, 2011.
18. T. Peyrin. Improved Differential Attacks for ECHO and Grostl. In Tal Rabin, editor, CRYPTO, LNCS vol. 6223, pp. 370–392. Springer, 2010.
19. R. L. Rivest. The MD5 message-digest algorithm. Request for Comments (RFC) 1320, Internet Activities Board, Internet Privacy Task Force, April 1992.
20. X. Wang, Y. L. Yin and H. Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, CRYPTO, LNCS vol. 3621, pp. 17-36. Springer, 2005.
21. X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, EUROCRYPT, LNCS vol. 3494, pp. 19-35. Springer, 2005.