Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials

Itai Dinur[1], Orr Dunkelman[1,2,⋆], and Adi Shamir[1]

[1] Computer Science department, The Weizmann Institute, Rehovot, Israel
[2] Computer Science Department, University of Haifa, Israel

Abstract. On October 2-nd 2012 NIST announced its selection of the Keccak scheme as the new SHA-3 hash standard. In this paper we present the first published collision finding attacks on reduced-round versions of Keccak-384 and Keccak-512, providing actual collisions for 3-round versions, and describing an attack which is 2^{45} times faster than birthday attacks for 4-round Keccak-384. For Keccak-256, we increase the number of rounds which can be attacked to 5. All these results are based on a generalized *internal differential attack* (introduced by Peyrin at Crypto 2010), and use it to map a large number of Keccak inputs into a relatively small subset of possible outputs with a surprisingly large probability. In such a *squeeze attack* it is easier to find random collisions in the reduced target subset by a standard birthday argument.
Keywords: Hash function, cryptanalysis, SHA-3, Keccak, collisions, internal differentials, squeeze attack.

1 Introduction

One of the stated reasons for the recent selection of Keccak by NIST as the new SHA-3 hash standard was its exceptional resistance to cryptanalytic attacks [10]. Even though it was a prime target for several years and many cryptanalysts have tried to break it (see [1, 2, 4, 8, 11–14, 16, 20, 21]), there was very limited progress so far in finding collisions even in greatly simplified versions of its various flavors. In particular, there were no published collision finding attacks *on any number of rounds* of its two largest flavors (Keccak-384 and Keccak-512), and only three published collision finding attacks on Keccak-256 ([16, 21] attacked two rounds, and [12] doubled the number of rounds to 4). 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 extremely small, as was rigorously shown in [11]. We bypass this seemingly insurmountable barrier by using a different kind of differential property, whose probability is not bounded by such a proof.[1] By using the new property, we provide in this paper either the first or an improved

⋆ The second author was supported in part by the Israel Science Foundation through grant No. 827/12.

[1] While we do not actually go beyond the bound mentioned in [11], its proof does not apply to the type of differential properties we consider in this paper.

attack on all these flavors: For Keccak-384 and Keccak-512 we describe practical attacks (with actual collisions) on three rounds, and impractical attacks on four rounds of Keccak-384. For Keccak-256 we increase the number of rounds which can be attacked from 4 to 5. The previous collision attacks and our new results are summarized in Table 1.

Reference	Keccak-224	Keccak-256	Keccak-384	Keccak-512
[16, 21]	2 (practical)	2 (practical)	-	-
[12]	4 (practical)	4 (practical)	-	-
This paper	-	5 (2^{115})	3 (practical) 4 (2^{147})	3 (practical)

Table 1. Collision attacks on round-reduced Keccak: the number of rounds attacked with the corresponding time complexity in parentheses

Our new attacks use many ideas which were already known in some limited form, but improves and combines them in new ways. They are a special type of the very general notion of *subset cryptanalysis*, which tries to track the statistical evolution of a certain set of values (which could be single states, pairs of states, or a collection of states with "don't care parts") through the various operations in the cryptographic scheme. In general, the goal in subset cryptanalysis is to find a subset of inputs which are mapped with larger than expected probability to some pre-fixed subset of all possible outputs. This is a widely used technique, which includes as special cases most of our standard cryptanalytic attacks, including differential, integral, and linear attacks, both in the single key and in the related key cases. The first step in subset cryptanalysis is to construct a *subset characteristic* which associates a triplet (input subset, output subset, transition probability) to each internal operation f of the cryptosystem. The transition probability specifies the probability that a random state chosen from the input subset will be a member of the output subset after applying f. Based on standard randomness assumptions, the total probability of the characteristic is calculated by multiplying the various transition probabilities. Subset cryptanalysis is typically used in order to construct a distinguisher, which makes it possible to extract information about the last subkey of a cryptosystem.

Previous examples of subset cryptanalysis include partitioning cryptanalysis [15] which divides the plaintext space and the output space (or the one-before the last round value space) into sets which are related with non-trivial probabilities. Other works track the development of the "subset" through the cryptographic primitive by looking for invariants, e.g., fixed-points or fixed subsets. For example, in [19] a subset of invariant values under the encryption process (in weak key classes) in PRINTcipher are identified. Another example is the subset of special states identified in [18] which contains states whose left half is equal to the right half, and is an invariant of the encryption under keyless

AES. We also note the close relationship between our approach and many of the self-similarity properties identified over the years. Slide attacks [6] (as well as the original flavor of related-key attacks [5]) is built over pairs of plaintexts which are shifted versions of each other in the encryption process. In many cases (e.g., Feistel ciphers), it is easy to rewrite the slide requirement as a relation between the slid pairs by defining the subsets according to the slid relation.

Squeeze Attacks In the case of hash functions, we can use subset characteristic in a different way, which we call a *squeeze attack*. To motivate this attack, assume that the hash function maps a set S of possible inputs into a set D of possible outputs. By the birthday paradox, we have to try a subset $S' \subseteq S$ of size $\sqrt{|D|}$ of inputs before we expect to find the first collision in D. Consider now the variant of this attack in which we discard all the outputs we generate which do not fall into a particular subset $D' \subseteq D$. Since D' is smaller than D we need fewer samples in it in order to find a collision, but finding each sample is more expensive. To find which effect is stronger, assume that the probability of picking an input in S' whose output is in D' is p, and that D' contains a fraction q of the points in D. The number of outputs in D' we need is $\sqrt{|D'|} = \sqrt{q|D|}$, and the number of inputs in S' we have to try is $\sqrt{q|D|}/p$. When the mapping is random, $p = q$ and this variant of the attack is worse than the birthday bound for all D' which are smaller than D. However, if we can exploit some non-random behavior of the hash function in order to find sets S' and D' for which $p^2 > q$, we can get an improved collision finding algorithm. We call it a squeeze attack since we are forcing a larger than expected number of inputs to squeeze into a smaller subset of possible outputs in which collisions are more likely. By memorizing only such outputs and discarding all the other outputs we generate, we can reduce both the time and the space needed to find collisions in the given hash function (see Figure 1). The analysis above shows that *any* subset characteristic for which $p^2 > q$ suffices for an efficient squeeze attack on a hash function, provided only that we can generate sufficiently many inputs in the initial subset of the characteristic. This is more flexible than standard differential cryptanalysis of hash functions, where a high-probability differential characteristic can be directly used in a collision attack only if it leads to a zero difference in the output value.

The squeeze attack was used in several previous attacks, but usually in cases where p was 1, in which the idea was beneficial for any $q < 1$ (e.g., in [7]). In this paper, we apply the squeeze attack to Keccak with $p \ll 1$. Our starting point is the observation that most of the operations in Keccak have potentially dangerous symmetry properties. The designers of Keccak were fully aware of this fact, and decided to use asymmetric round constants precisely in order to avoid this problem. However, the constants they chose were of very low Hamming weight, and thus their effect was small, changing a fully symmetric state into an almost symmetric state.

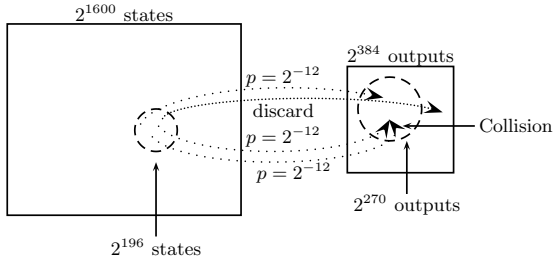


Fig. 1. A Squeeze Attack with $|S| = 2^{1600}, |S'| = 2^{196}, |D| = 2^{384}, |D'| = 2^{270}, p = 2^{-12}$

Generalized Internal Differential Cryptanalysis. In this paper, we generalize the technique of *internal differential cryptanalysis* developed by Peyrin [22] in the cryptanalysis of the Grøstl hash function. While in standard differential attacks we consider two different plaintexts, and follow the evolution of the difference between them, in internal differential attacks we consider only one plaintext, and follow the statistical evolution of the differences between its parts. In the case of Keccak, we use internal differential cryptanalysis in order to follow the statistical evolution of almost symmetric states through the first few rounds of Keccak. For example, if the symmetry we consider is that the first half of the state should be equal to the second half of the state, then we follow the evolution of small differences between these two parts through the various cryptographic operations. Note that fully symmetric states have a zero internal difference, which remains zero as the state goes through symmetry preserving operations, whereas almost symmetric states have a low Hamming weight internal difference, which in many cases remains low Hamming weight after such operations.

Our approach generalizes and extends the original idea presented in [22] in several ways: first, internal differential cryptanalysis was previously shown to be applicable to hash functions with explicitly defined and completely separate data-paths. In this paper, we show that it is applicable in a much broader setting, where the cryptosystem is not necessarily built using separate data-paths, but still admits differential relations in the internal state that we can follow and control. Second, in [22] Peyrin considers differences between two halves of the state, whereas most of our attacks consider more complex internal structures which divide the state into more than two parts. This approach requires definitions of new objects that capture the notion of these generalized difference relations and allow us to analyze them. In addition to these generalizations, we introduce several new techniques such as aggregating multiple internal differences, which allow us to extend our subset characteristics, and thus attack more rounds of reduced Keccak.

2 Description of Keccak

In this section, we briefly describe the sponge construction and the Keccak hash function. More details can be found in the Keccak specification [4]. The sponge construction [3] works on a state of b bits, which is split into two parts: the first part contains the first r bits of the state (called the outer part) and the second part contains the last $c = b - r$ bits of the state (called the inner part).

Given a message, it is first padded and cut into r-bit blocks, and the b state bits are initialized to zero. The sponge construction then processes the message in two phases: In the absorbing phase, the message blocks are processed iteratively by XORing each block into the first r bits of the current state, and then applying a fixed permutation on the value of the b-bit state. After processing all the blocks, the sponge construction switches to the squeezing phase. In this phase, n output bits are produced iteratively, where in each iteration the first r bits of the state are returned as output and the permutation is applied to the state.

The Keccak hash function uses multi-rate padding: given a message, it first appends a single 1 bit. Then, it appends the minimum number of 0 bits followed by a single 1 bit, such that the length of the result is a multiple of r. Thus, multi-rate padding appends at least 2 bits and at most $r + 1$ bits.

The Keccak versions submitted to the SHA-3 competition have $b = 1600$ and $c = 2n$, where $n \in \{224, 256, 384, 512\}$. The 1600-bit state can be viewed as a 3-dimensional array of bits, a[5][5][64], and each state bit is associated with 3 integer coordinates, a[x][y][z], where x and y are taken modulo 5, and z is taken modulo 64.

The Keccak permutation consists of 24 rounds, which operate on the 1600 state bits. Keccak uses the following naming conventions, which are helpful in describing its round function:

- A row is a set of 5 bits with constant y and z coordinates, i.e. $a[*][y][z]$, or $r(y, z)$.
- A column is a set of 5 bits with constant x and z coordinates, i.e. $a[x][*][z]$.
- A lane is a set of 64 bits with constant x and y coordinates, i.e. $a[x][y][*]$.
- A slice is a set of 25 bits with a constant z coordinate, i.e. $a[*][*][z]$.

Each round of the Keccak permutation consists of five mappings $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$. The five mappings given below are applied for each x,y, and z (where the state addition operations are over $GF(2)$):

1. θ is a linear map, which adds to each bit in a column, the parity of two other columns.
   θ: $a[x][y][z] \leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1]$
2. ρ rotates the bits within each lane by T(x,y), which is a predefined constant for each lane.
   ρ: $a[x][y][z] \leftarrow a[x][y][z + T(x, y)]$

3. π reorders the lanes.
   π: $a[x][y][z] \leftarrow a[x'][y'][z]$, where $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \end{pmatrix}$
4. χ is the only non-linear mapping of Keccak, working on each of the 320 rows independently.
   χ: $a[x][y][z] \leftarrow a[x][y][z] + ((\neg a[x+1][y][z]) \wedge a[x+2][y][z])$
   Since χ works on each row independently, it can be viewed as an Sbox layer which simultaneously applies the same 5 bits to 5 bits Sbox to the 320 rows of the state. We note that the Sbox function is an invertible mapping, and some of our techniques are based on the known observation that the algebraic degree the algebraic degree of each output bit of χ as a polynomial in the five input bits is only 2.
5. ι adds a 64-bit round constant to the first lane of the state.
   ι: a[0][0][*]$\leftarrow$a[0][0][*]$+RC[i_r]$
   Since we analyze in this paper round-reduced variants of Keccak with at most 5 rounds, we are only interested in the first five round constants: 0000000000000001, 0000000000008082, 800000000000808a, 8000000080008000, 000000000000808b (given respectively in hexadecimal using the little-endian format). Note that all five round constants have a low Hamming weight and the first round-constant has a Hamming weight of only 1.

3 Notations

Given a message M, we denote its length in bits by $|M|$. Unless specified otherwise, in this paper we assume that $|M| = r - 2$, namely we consider only single-block messages of maximal length. Given M, we denote the initial state of the Keccak permutation as the 1600-bit word $\overline{M} \triangleq M||11||0^{2n}$, where $||$ denotes concatenation.

The first three operations of Keccak's round function are linear mappings, and we denote their composition by $L \triangleq \rho \circ \pi \circ \theta$. We sometimes refer to L as a "half round" of the Keccak permutation, where $\iota \circ \chi$ represents the other half. We denote the Keccak nonlinear function on 5-bit words defined by varying the first index by $\chi_{|5}$. The difference distribution table (DDT) of this function is a two-dimensional 32×32 integer table, where all the differences are assumed to be over $GF(2)$. The entry $DDT(\delta^{in}, \delta^{out})$ specifies the number of input pairs to this Sbox with difference δ^{in} that produce the output difference δ^{out} (i.e., the size of the set $\{x \in \{0,1\}^5 \mid \chi_{|5}(x) + \chi_{|5}(x + \delta^{in}) = \delta^{out}\}$).

Given a set S of internal states of Keccak, we define the action of each of Keccak's mappings on the set by applying it to every element of the set (e.g., $\theta(S) = \{\theta(u) | u \in S\}$).

4 Description of Our Basic Techniques

Given a subset characteristic for the compression function of a given hash function, we can describe our basic squeeze attack in the following way:

1. Pick an arbitrary message for which the values entering the compression function are in the initial subset of the characteristic.
2. Apply the compression function. If the subset characteristic is satisfied, compute the output of the compression function. Otherwise, discard the message and go back to Step 1.
3. Store the output in a table (along with the message). In case a collision is found, stop and output the collision. Otherwise, go back to Step 1.

If the size of the output set is 2^d (i.e., $|D'| = q|D| = 2^d$ using the notation of the Introduction), then after $2^{d/2}$ messages for which the characteristic is followed, we expect a collision due to the birthday paradox. Hence, when the probability of the subset characteristic is p, the time complexity of finding a collision is $p^{-1} \cdot 2^{d/2}$ and the memory complexity is[2] $2^{d/2}$. To optimize the attack, we need a subset characteristic for which p is as high as possible and d is as small as possible.

4.1 Internal Difference Sets

A very interesting observation concerning Keccak is that four out of its five internal mappings (all but ι), are translation invariant in the direction of the z axis (as was already noted in the Keccak submission paper [4]). Namely, if one state is the rotation of another state with respect to the z-axis (i.e., satisfies $b[x][y][z] = a[x][y][z+i]$, for some value of i), then applying to them any of the θ, ρ, π, χ operations, maintains this property. To exploit this symmetry, we pick subsets which are invariant with respect to the rotation along the z-axis with all the non-trivial possible choices of i. Namely, given a rotation index $i \in \{1, 2, 4, 8, 16, 32\}$, the subsets are all the states for which $a[x][y][z] = a[x][y][z+i]$.

In most of the remainder of this section, we assume for the sake of simplicity that $i = 16$, but note that all of our definitions extend naturally to any $i \in \{1, 2, 4, 8, 16, 32\}$. For $i = 16$, a symmetric state $a[x][y][z]$ is composed of four repetitions of slices 0–15 (see Example 1). Each such sequence of slices (0–15, 16–31, 32–47, 48–63) is called a *consecutive slice set* or CSS in short. Applying any of the four operations θ, ρ, π, χ to a symmetric state in which all CSS's are equal, does not disturb its symmetry. The application of ι interferes with this symmetry, since the round constants are not the same among the consecutive slice sets. However, given the low weight of the constants used by ι, the state remains close to being symmetric.

To deal with ι, we have to extend our point of view, and consider states for which the equality "almost holds". The subsets used in our subset characteristics are *internal differences*, which measure how close the state is to a symmetric state. Generally speaking, this can be done by computing the XOR differences between the first consecutive slice set, and each of the three other ones, denoted by the triplet $(\Delta_1, \Delta_2, \Delta_3)$. We define an internal difference in Keccak to be

[2] Notice that we can use either Floyd's cycle finding algorithm [17] or the parallel collision search algorithm [23] to reduce the memory complexity of the attack, depending on the relative sizes of its domain and range subsets.

```
|169D169D169D169D|A965A965A965A965|3EC73EC73EC73EC7|9025902590259025|C264C264C264C264|
|A34BA34BA34BA34B|0F330F330F330F33|4902490249024902|3D683D683D683D68|613D613D613D613D|
|C684C684C684C684|B368B368B368B368|589B589B589B589B|5F335F335F335F33|E27AE27AE27AE27A|
|22E822E822E822E8|3D583D583D583D58|B37AB37AB37AB37A|104710471047 1047|D525D525D525D525|
|60F360F360F360F3|C3E4C3E4C3E4C3E4|37FA37FA37FA37FA|8193819381938193|69BA69BA69BA69BA|
```

The state is described as a matrix of 5×5 lanes of 64 bits, ordered from left to right, where each lane is given in hexadecimal using the little-endian format. Each lane of the state consists of 4 repetitions of a 16-bit word.
Example 1: A symmetric state with $i = 16$

the set of states with a fixed value of $(\Delta_1, \Delta_2, \Delta_3)$. Obviously, when all 4 CSS's are equal, the differences between them are zero and the subset is called a *zero internal difference*.

Alternatively, we can define an internal difference set as a coset in a group, using a single *representative state* v and adding to it all the fully symmetric states: $\{v+w|w \text{ is symmetric}\}$. In general, given a rotation index i, we represent an internal difference using the pair $[i, v]$ (or $[16, v]$ in case $i = 16$). Obviously, this representation is redundant as we can select any $u \in [16, v]$ as the representative state. However, as shown in the next subsection, it allows us to describe the evolution of an internal difference $[i, v]$ through Keccak's linear mappings in a very compact way.

Since an internal difference does not place any constraint on the value of the first CSS, it will sometimes be convenient to choose a *canonical representative state* for which this value is zero, and we denote it by $\hat{v}$. Namely, for an internal difference defined by $(\Delta_1, \Delta_2, \Delta_3)$, the *values* of the four CSS's in the canonical representative state are $\mathbf{0}$, Δ_1,Δ_2 and Δ_3, respectively.

4.2 The Evolution of Internal Differences Through Keccak's Permutation

As in standard differential cryptanalysis, we consider the difference between the CSS's, rather than the actual values. Hence, the zero internal difference passes with probability 1 all the four operations θ, ρ, π, χ, just as a zero difference in a differential characteristic passes through any operation.

Unlike a classical differential characteristic, in an internal differential characteristic, the addition of a constant (i.e., the ι operation) effects the characteristic by introducing a difference between the equal CSS's. This difference then propagates through the other operations, and its development has to be studied and controlled. Luckily, we can construct internal differential characteristics for Keccak (with good probability) that track this evolution of "distance" from a zero internal difference through the various Keccak mappings.

Given the affine nature of an internal difference, tracking its evolution through Keccak's affine mappings is trivial (and does not change the probability of the internal differential characteristic): due to the translation invariance property and the associativity of linear operations, the action of the first three mappings on $[i, v]$ is determined by their action on the representative state, i.e.,

$\theta([i,v]) = [i,\theta(v)]$, $\rho([i,v]) = [i,\rho(v)]$ and $\pi([i,v]) = [i,\pi(v)]$. Since ι simply adds a constant to each state of the set then $\iota([i,v]) = [i,\iota(v)]$ as well.

The Evolution of Internal Differences through χ. In contrast to the linear mappings, applying χ, the non-linear mapping, to a randomly selected state from an internal difference, the output internal difference depends on the actual input, i.e., the output can belong to one of several internal differences. Just as in differential cryptanalysis, we can choose a single output internal difference, and then calculate the probability of the transition from the input internal difference to this output internal difference.

When a state of an internal difference which is not symmetric enters the χ function, we have to consider the possible outcomes in terms of "distance" from the zero internal difference. To do so, we consider the rows on which χ operates using an object called a *rotated row set*. For $i = 16$, a rotated row set contains a row $r(y,z)$ in the first CSS, along with its 3 symmetric counterparts $r(y,z+16), r(y,z+32)$ and $r(y,z+48)$ in the other CSS's (see Example 2). We note that given the input internal difference, once the value of $r(y,z)$ is set, we know the value of the remaining rows as well. Hence, given the value of $r(y,z)$ we can compute the corresponding outputs, and check the resulting output internal difference.

Once we perform this operation, we can associate with each input internal difference all the possible output internal differences (and the corresponding probabilities) by trying all 32 possible values for $r(y,z)$. In the particular case where the input internal difference assigns a zero difference to all the rows of a rotated row set, it passes through the χ mapping with probability 1. Similarly to differential cryptanalysis, we call such a rotated row set *inactive* (with respect to the internal difference), whereas a rotated row set with a non-zero difference is called *active*.

For $i = 32$, each rotated row set contains exactly 2 Sboxes (rows), i.e., v specifies a single input difference for the Sbox pair. In this case one can easily use the difference distribution table of the Sbox to determine the distribution of the output difference δ^{out} given the input difference δ^{in}.

In the internal differences that we consider in this paper, most rotated row sets contain at most two distinct input values to the Sbox. We call such a rotated row set *sparse*. In active sparse rotated row sets,[3] one can divide the values $r(y,z), r(y,z+16), r(y,z+32)$ and $r(y,z+48)$ (or $r(y,z), r(y,z+i), \ldots$ for general $i \in \{1,2,4,8,16,32\}$) into two groups, each with the same input to the Sbox (see Example 3). Obviously, each group of Sboxes has the same output, leading to a sparse output internal difference as well. Since there is only a single input difference between the two groups of Sboxes, we can use the difference distribution table also in the more general case of $i \neq 32$, when a rotated row set is sparse.

[3] For inactive rotated row sets, the output internal difference is necessarily 0.

|0001000100010001|0001000100010001|0001000100010001|0001000100010001|0001000100010001|

The first five lanes of a state in which the 20 bits of the first rotated row set for $i = 16$ are set to 1. The lanes are ordered from left to right, where each lane is given in hexadecimal using the little-endian format.

Example 2: A rotated row set

|0001000100010001|0001000000010000|0000000000000000|0001000100010001|0000000100000001|

The first five lanes (given in the format of Example 2) of a state in an internal difference in which the first rotated row set is sparse for $i = 16$. The (binary) value of $r(0, 0)$ and $r(0, 32)$ is 10011, while the value of $r(0, 16)$ and $r(0, 48)$ is 11010. In this example, the internal difference fixes the difference of 01001 between the two groups of rows. The value of the other rows is zero.

Example 3: A sparse rotated row set

The Weight of Internal Differences. Finally, we give a heuristic concerning the "quality" of a given internal difference in a characteristic. The closer the internal difference is to the zero internal difference, its *weight* (i.e., the minimal Hamming weight of a state in the internal difference) is lower.[4] Since the zero internal difference contains the zero state, its weight is zero, and the weight of an internal difference measures the minimal Hamming distance between a state in the internal difference and a symmetric state. In general, a low-weight internal difference has only a few active rotated row sets, and thus passes through χ with high probability. In this paper, we construct characteristics whose internal differences have a low weight (and thus a high probability) by choosing low-weight internal differences as outputs, as well as a few additional techniques which will be described in the rest of this paper. As a preliminary example, consider Characteristic 1 in Appendix C. This characteristic starts from the zero internal difference and extends to 1.5 Keccak rounds with probability 1, where the final internal difference has a weight of 11.

5 Exploiting Internal Differential Characteristics in Collision Attacks on Keccak

In this section, we describe optimizations that allow us to devise efficient attacks on round-reduced Keccak using internal differential characteristics.

[4] While there may be many states with minimal Hamming weight in an internal difference, we can calculate one of them from an arbitrary state w in the internal difference: we iterate over all sets of 4 bits (in case $i = 16$), each containing one bit in the first CSS and its symmetric counterparts in the other 3 CSS's. For each such set, we compute the majority of its bits in w, and complement it if its majority is 1.

5.1 Choosing the Value of the Rotation Index

Recall that a subset characteristic maps an input, selected from a subset of inputs to the compression function, to a restricted output set of size 2^d with probability p. In order to find a collision, we have to try about $p^{-1} \cdot 2^{d/2}$ such inputs, and in the case of Keccak, we need the ability to generate $p^{-1} \cdot 2^{d/2}$ single-block messages M, such that $\overline{M}$ is a member of the initial internal difference. In our basic attack, we use a zero internal difference $(i, \mathbf{0})$ for a fixed $i \in \{1, 2, 4, 8, 16, 32\}$ (i.e., we restrict our messages M such that $\overline{M} \in (i, \mathbf{0})$), which implies that we are free to choose the value of the first i bits in each lane of the outer (controllable) part of the initial state, not including the lane containing the padding, in which we can only choose the value of the first $i - 2$ bits.[5] Exploiting the fact that the initialization sets the inner (uncontrollable) part of state to 0, and the fact that we can control the values of $r/64$ lanes when the rate is r, we can generate $2^{r \cdot (i/64) - 2}$ initial states which are symmetric. Hence, we have to ensure that $2^{r \cdot (i/64) - 2} \geq p^{-1} \cdot 2^{d/2}$.

As we decrease the value of i, we increase the number of constraints on the internal differences, leading to a smaller expected output subset size, thus reducing the complexity of the attack. On the other hand, a value of i which is too small leads to an insufficient number of possible messages for a collision attack. Hence, we choose the smallest $i \in \{1, 2, 4, 8, 16, 32\}$ such that $2^{r \cdot (i/64) - 2} \geq p^{-1} \cdot 2^{d/2}$ holds. We note that the value of i determines how a state is partitioned into rotated row sets, and thus it may also affect the probability p of a characteristic (i.e., we need to calculate p separately for each value of i).

5.2 Extending Internal Differential Characteristics

Constructing an internal differential characteristic which spans many rounds of Keccak reduces its probability significantly, leading to an inefficient collision attack which requires the evaluation of many messages. Thus, instead of covering all the attacked rounds, we extend the internal differential characteristic up to some point (in our attacks, one and a half rounds before the output), and continue to exploit Keccak's properties (such as the limited diffusion of its Sbox layer) in order to bound the size of the output subset (which is crucial in squeeze attacks). This is done by extending the internal differential characteristic to a subset characteristic without restricting its subsets to a particular form of $[i, \hat{v}]$. In fact, since the output subset is not an internal difference (and actually not even affine), in the final part we use subset cryptanalysis in its most general form.

Aggregating Internal Differences Using Affine Subspaces. Assume that the internal difference part of the subset characteristics ends just before the χ layer with an internal difference $[i, \hat{v}]$. We aggregate all the potential values of

[5] The calculation for the padded lane does not apply for the case of $i = 1$, but we do not use this value in our attacks on Keccak.

$[i, \hat{u}]$, the output internal difference of χ, into an affine subspace by considering each rotated row set independently, and computing $\hat{u}$ in symbolic form (i.e., by allocating linear variables). We then continue and apply L to the symbolic form of $\hat{u}$, and thus maintain the knowledge of the affine subspace up to the χ function of the next round.

Since the computed symbolic form of $\hat{u}$ may include some impossible values, this may increase the bound on the size of the output size. However, due to the limited diffusion properties of χ, it is easy to show explicitly that every single-bit difference in a rotated row set can result in allocation of at most two variables, hence the number of allocated variables for $\hat{u}$ is upper-bounded by twice the weight of $[i, \hat{v}]$. Moreover, when a rotated row set is sparse (with respect to $[i, \hat{v}]$), its Sboxes assume only (at most) two values, with a single input difference which is fixed by $\hat{v}$. Since the algebraic degree of the Keccak Sbox is only 2, all the possible output differences of the Sboxes form an affine subspace (as observed in the Keccak reference document [4]). Thus, when all the rotated row sets with respect to $[i, \hat{v}]$ are sparse, the aggregated internal difference does not include impossible internal differences.

In order to distinguish between explicit binary vectors, and symbolic forms, we denote the explicit vector by $\hat{u}$ and the symbolic form by $\hat{\mathbf{u}}$. We note that $[i, \hat{\mathbf{u}}]$ still represents an affine subspace whose dimension is increased compared to $[i, \hat{u}]$ by the number of allocated variables.

Bounding the Size of the Output Subset Beyond the Last χ Mapping. Assume that we have an affine subspace of the form $[i, \hat{\mathbf{u}}]$ as an input to χ, after which χ and ι are applied, and the state is truncated and sent to the output. Our goal is to upper bound the size of the output subset without reducing its probability (which may happen if we restrict it to an affine subspace).

Clearly, the final application of ι does not affect the size of the output subset, and can be ignored. In order to obtain a good bound, we exploit the limited diffusion of χ which maps each row to itself and in particular, maps each set of 64 rows (320 consecutive bits) of the form $a[*][y][*]$ to itself. As the output consists of the first n bits of the final state, we want to bound the number of its possible values by computing the size of the subset before the last χ mapping when projected to its first $320\lceil n/320 \rceil$ bits. Namely, for output sizes of 224, 256, 384 and 512, it is sufficient to compute the size of the subset before the χ mapping on its first 320, 320, 640 and 640 bits respectively. For $n = 384$ we can achieve a better bound by using a more specific property of χ: each bit $a[x][y][z]$ at the output of χ, depends only on the 3 input bits $a[x][y][z]$, $a[x+1][y][z]$ and $a[x+2][y][z]$. Thus, the 64 bits of the lane $a[x][y][*]$ at the output of χ, depend only on the 3 input lanes $a[x][y][*]$, $a[x+1][y][*]$ and $a[x+2][y][*]$. In the case of $n = 384$, the first 320 bits are mapped to themselves by χ, and the remaining 64 bits depend only on 192 bits. Thus, in order to upper bound the output subset size it is sufficient to compute the size of the subset when projected to its first 320+192=512 bits.

We now show how to bound the size of the n-bit output subset, given that it depends only on the first n' bits before the χ mapping, and the affine subspace at the entry to χ is represented by $[i, \hat{\mathfrak{u}}]$. We first assign the variables of $\hat{\mathfrak{u}}$ an arbitrary value (e.g., zero). We denote the resultant binary vector by $\hat{u}$, and obtain a basic bound in this simplified case, where $[i, \hat{u}]$ is an internal difference: recall that each rotated row set can assume at most 32 values, hence each set of 320 bits of the form $a[*][y][*]$ can assume at most $32^i = 2^{5i}$ values. Thus, for $n = 224$ and $n = 256$ we obtain a basic bound of 2^{5i}, and for $n = 512$ we obtain a basic bound of $2^{2(5i)}$. For $n = 384$, the computation can be split into two parts: the 320 LSBs of the output can assume at most 2^{5i} values, and the 64 MSBs depend on 3 input lanes and can assume at most $min(2^{64}, 2^{3i})$ values. This gives a basic bound of $2^{5i} \cdot min(2^{64}, 2^{3i})$ for $n = 384$. In all cases, we emphasize that the bound only depends on i and n (which determines n'), rather than on the actual values of the n' bits of $\hat{u}$.

In the symbolic case, the n' bits of $\hat{\mathfrak{u}}$ are expressions, and the basic bound applies independently for each possible value of these n' bits. Consequently, in order to upper bound the output subset size, we need to multiply the basic bound by the number of possible values that the n' expressions can assume. Since the expressions are affine, we can easily compute their number of possible values by computing their dimension using simple linear algebra.

In order to minimize the dimension of the n' expressions at the output, we have to minimize the number of variables allocated in the previous round, when extending the internal differential characteristic. Since we do not allocate any variables to inactive rotated row sets, this can be assured if the final internal difference of the characteristic (before the variable allocation) is of low weight. Thus, in addition to the influence of the weight of the internal differences on the probability p of a characteristic, the weight also plays a role in bounding the size of the output subset 2^d.

6 Collision Attacks on Round-Reduced Keccak-384 and Keccak-512

In this section, we present the details of our practical 3-round collision attacks on Keccak-384 and Keccak-512 and our non-practical 4-round attack on Keccak-384. Although our techniques can be applied to all variants of Keccak, actual collisions were already presented for 4 rounds of Keccak-224 and Keccak-256 in [12], and thus we focus first on Keccak-384 and Keccak-512, for which there are no previously published collision attacks on any number of rounds.

6.1 Practical Collisions in 3-Round Keccak-512

In order to find actual collisions in 3-round Keccak-512, we used the internal differential characteristic given in Characteristic 1 in Appendix C. This characteristic spans only the first Keccak round and the L mapping of the second round (i.e., the first 1.5 rounds), and has a probability of 1. In our attack, we choose

$i = 4$, and we use the techniques of Section 5.2 in order to bound the size of the output subset: we apply the variable allocation technique to the final internal difference of the characteristic (whose weight is 11) to allocate 22 variables and to extend the characteristic beyond the χ mapping of the second round. The basic bound on the size of the output subset is $2^{2 \cdot 5 \cdot i} = 2^{40}$, and the dimension of the first $n' = 640$ linear expressions is 22 (the maximal possible dimension of linear expressions with 22 variables). This gives a bound of $2^{40+22} = 2^{62}$ on the size of the output subset. Since the probability of the characteristic is 1, we have to try about 2^{31} single-block messages which give initial states in the zero internal difference $[4, \mathbf{0}]$, in order to find a collision with good probability. Since $n = 512$, in this case we have $r = 576$ and we can choose a sufficient number of $2^{r \cdot (i/64) - 2} = 2^{34}$ messages that satisfy the constraints. We implemented the attack and obtained actual collisions in 3-round Keccak-512. A concrete example (found in less than an hour on a single PC) is given in Collision 1 in Appendix D.

6.2 Practical Collisions in 3-Round Keccak-384

For Keccak-384, we can easily use the same characteristic (Characteristic 1 from Appendix C). However, we prefer to use a different characteristic which leads to a more efficient attack, and is also used as a basis for our 4-round attack of Keccak-384 (described in the next section). The idea is to choose a low-weight initial internal difference that limits the increase in the weight caused by the second-round θ mapping, and thus reduces the weight of the internal difference at the entry to the second-round χ mapping.[6] In particular, we make sure that θ acts as the identity on some low Hamming weight vector in the internal difference after the first round.

Searching for Internal Differential Characteristics. The most interesting set of states which are fixed-points of θ is the *column parity kernel* or *CP-kernel*, which was defined in the Keccak submission document [4]: a 1600-bit state is in the CP-kernel if all of its columns have an even parity, which makes such a state a fixed-point of θ. Denote the initial internal difference in our characteristic by $[i, v_0]$ and the internal difference obtained after one round by $[i, v_1]$. We require that there exists some low Hamming weight state $u_1 \in [i, v_1]$ in the CP-kernel, and also set a similar constraint on $[i, v_0]$, which (unlike the attack on Keccak-512) is not zero. Namely, we require that there exists a low Hamming weight state $u_0 \in [i, v_0]$ in the CP-kernel (otherwise θ will significantly increase the weight of the internal difference already in the first round).

Techniques to find state differences that stay in the CP-kernel for two consecutive rounds were described in [11, 14, 21] in order to construct low Hamming weight classical differential characteristics. Here, we use these techniques in a straightforward way in order to construct low Hamming weight internal differential characteristics that fulfill the two constraints: As done in several previous

[6] We note that the rate r of Keccak-384 is much larger than the rate of Keccak-512, and thus we could not choose a similar initial internal difference for Keccak-512.

paper which analyze standard differential characteristics of Keccak, we first assume that the χ mappings act as an identity on the input internal differences (this is typically possible when the input internal difference is of low weight). As a result, the evolution of the internal differential characteristic is completely linear and deterministic, and if we ignore the ι constants, then it is identical to the evolution of a standard differential characteristic with the same initial state-difference (which in our case represents an internal difference). Thus, we can use the previous techniques to find good internal differential characteristics which ignore the ι constants. Finally, we post-filter these characteristics by trying to cancel the ι constants using the additional degrees of freedom offered by the χ mappings.

The best internal differential characteristic that we found for Keccak-384 (which spans 1.5 rounds) is given in Characteristic 2 in Appendix C. Note that its final internal difference has a weight of 6, which is lower compared to the weight of 11 of the final internal difference in Characteristic 1. On the other hand, the characteristic has a probability of 2^{-12} due to the transition through the first χ mapping, whereas Characteristic 1 has probability 1. However, we can easily reduce the workload of finding initial states that conform to this characteristic from the trivial 2^{12} to 1 (as described next), while losing only 12 degree of freedom.

Reducing the Workload of Finding Messages Conforming to the First χ Transition. When the input to the first χ mapping is a state which belongs to a non-zero internal difference $[i, \hat{v}]$, this transition is associated which a probability which is lower than 1. However, we can reduce the workload of finding messages conforming to the first χ transition: we analyze each rotated row set independently and restrict its inputs to an affine subspace for which the first χ transition occurs with probability 1. Due to the fact that L is affine, we can compute an affine subspace of initial states in the first internal difference of the characteristic that satisfy the first χ transition.

Note that we restrict the initial states to an affine subspace that may not include all the values which guarantee the first χ transition. Thus, this optimization can also be detrimental by reducing the available degrees of freedom further compared to the non-optimized method of trying arbitrary states in the initial internal difference. Nevertheless, due to the limited diffusion properties of χ, the transition of every single-bit difference in a rotated row set of $\hat{v}$ depends on the values of at most two state bits. Hence, the total number of state bits whose values we restrict (and the total number of degrees of freedom that we lose as a result) in order to guarantee the first χ transition is upper-bounded by twice the weight of $[i, \hat{v}]$. Indeed, in Characteristic 2 the weight of the internal difference at the input to the first χ mapping is 6, and we lose 12 degree of freedom.

The Full Attack. In our 3-round attack on Keccak-384, we choose $i = 4$, and calculate the bound on the output subset as follows: we use the variable allocation technique to allocate 12 variables (which is the maximal number since the final

internal difference has a weight of 6) and extend the characteristic beyond the χ mapping of the second round. The basic bound on the size of the output subset is $2^{8 \cdot 4} = 2^{32}$, and the dimension of the first $n' = 512$ linear expressions is 12. This gives a bound of $2^{32+12} = 2^{44}$ on the size of the output subset. Since the workload to find initial states that conform to Characteristic 2 is 1, we have to try (at most) 2^{22} such initial states in order to find a collision with high probability. For $n = 384$, we have $r = 832$ and we can choose a sufficient number of $2^{-12} \cdot 2^{r \cdot (i/64)-2} = 2^{38}$ messages that satisfy the constraints. We implemented the attack and obtained actual collisions in 3-round Keccak-384. A concrete example (found in less than a minute on a single PC) is given in Collision 2 in Appendix D.

6.3 A Collision Attack on 4-Round Keccak-384

In this subsection, we briefly present a collision attack on 4-round Keccak-384. The attack is based on the 2.5-round internal differential characteristic given in Characteristic 3 in Appendix C, which is an extension by one round of the 1.5-round characteristic used in the 3-round attack on Keccak-384. The analysis of the attack is given in Appendix A, and shows that the expected time complexity of the attack is bounded by 2^{147}. This is non-practical, but 2^{45} times faster than the birthday bound of 2^{192}.

7 A Collision Attack on 5-Round Keccak-256

The target difference algorithm (TDA) was developed in [12] as a technique to link a differential characteristic (which starts from an arbitrary state difference) to the initial state of the Keccak permutation, using one permutation round. More precisely, the initial state difference of the characteristic is called the target difference, and the algorithm outputs many single-block message pairs which satisfy the target difference after one permutation round. Hence, a differential characteristic leading to a collision at the output after k rounds can be leveraged to a collision attack on $k + 1$ rounds of Keccak.

In this section, we present a 5-round collision attack on Keccak-256 which is based on an analogous variant of the TDA for internal differential cryptanalysis, and is called a *target internal difference algorithm* (TIDA). Analogously to the TDA, the TIDA is a technique that links an internal differential characteristic (which starts from an arbitrary internal difference) to the initial state of the Keccak permutation, using one permutation round. Thus, the initial internal difference of the internal differential characteristic is called the *target internal difference*, and the algorithm outputs single-block messages whose internal state belongs to the target internal difference after one permutation round.

Both the TDA, and the TIDA proposed in this paper are heuristic randomized algorithms, and we cannot formally prove their success. Given a subset characteristic (which is an extension of an internal differential characteristic) spanning k rounds of the Keccak permutation, a collision attack on $k+1$ rounds of Keccak consists of the following steps:

1. Run the TIDA on the target internal difference (derived from the first internal difference of the characteristic) with fresh randomness until it succeeds to output single-block messages satisfying the target internal difference after one permutation round.
2. Let M be the next message outputted by the TIDA (if no more messages remain, return to Step 1):
   (a) Run the Keccak permutation on $\overline{M}$. If the evolution of the state from the second round conforms with the internal differential characteristic, continue and calculate the output of the hash function. Otherwise, discard M and go to Step 2.
   (b) Store the output in a hash table next to M, and check if it collides with an output of a different message. If a collision is found, output the colliding message pair, otherwise go to Step 2.

In order to analyze the time complexity of the attack, we have to estimate the amortized time complexity of finding one message that satisfies the target internal difference after one permutation round. The amortized time is calculated as the ratio between the execution time of the TIDA and the number of messages that it returns in a single execution. If we assume that the amortized time is smaller than 1 (i.e., the amortized time is less than the execution time of the Keccak permutation), and the time of a single execution of the TIDA in Step 1 is not too large, then the time complexity analysis of the attack is similar to the analysis of the basic attack given in Section 4. Given that the size of the output set is 2^d values, then the memory complexity of the attack is $2^{d/2}$, similarly to the basic attack given in Section 4.

Our 5-round collision attack on Keccak uses the internal differential characteristic given in Characteristic 4 in Appendix C, which covers rounds 1–3.5. This characteristic is leveraged in order to attack 5 rounds using the techniques of Section 5, while the TIDA is used to find messages in the initial internal difference of the characteristic (after 1 Keccak round). The full details and analysis of the attack are given in Appendix B. Based on extensive simulations of the critical part of the attack, its estimated time complexity is at most 2^{115}, which is 2^{13} times faster than the birthday bound of 2^{128}.

8 Conclusions and Future Work

In this paper, we presented the first collision attacks on round-reduced Keccak-384 and Keccak-512, and for Keccak-256, we increased the number of rounds which can be attacked from 4 to 5. Our algorithms are based on a squeeze attack which uses internal differential cryptanalysis (which is a special case of subset cryptanalysis) in order to map a large subset of inputs into a small pre-fixed subset of all possible outputs, for which the birthday bound is significantly reduced.

Internal differential cryptanalysis is also very useful in attack scenarios which are different than the squeeze attack. For example, it is possible to use internal differential cryptanalysis in preimage attacks on hash functions, given that the

target output is contained in a specific subset of outputs. Moreover, one can think of several other attacks based on internal differential cryptanalysis (such as impossible internal differential cryptanalysis and rebound attacks), which are analogous to attacks in the standard differential setting.

An important future item is to construct better internal differential characteristics for Keccak, or prove that they do not exist (and thus extend the work of [11]). More generally, subset cryptanalysis, and in particular internal differential cryptanalysis, seems to be a fruitful research direction. It may improve the cryptanalytic toolbox, suggest better attacks on various schemes, and shed some light on the types of constants which are hazardous to the security of cryptosystems.

Acknowledgements: The authors would like to thank the anonymous referees for their very helpful comments on the preliminary version of this paper.

References

1. Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. NIST mailing list, 2009.
2. Daniel J. Bernstein. Second preimages for 6 (7? (8??)) rounds of keccak? NIST mailing list, 2010.
3. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.
4. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak SHA-3 submission. Submission to NIST (Round 3), 2011.
5. Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptology*, 7(4):229–246, 1994.
6. Alex Biryukov and David Wagner. Slide Attacks. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999.
7. Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque. Another Look at Complementation Properties. In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *Lecture Notes in Computer Science*, pages 347–364. Springer, 2010.
8. Christina Boura and Anne 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*, pages 1–17. Springer, 2010.
9. Anne Canteaut, editor. *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*. Springer, 2012.
10. Shu-jen Chang, Ray Perlner, William E. Burr, Meltem Sönmez Turan, John M. Kelsey, Souradyuti Paul, and Lawrence 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.
11. Joan Daemen and Gilles Van Assche. Differential Propagation Analysis of Keccak. In Canteaut [9], pages 422–441.

12. Itai Dinur, Orr Dunkelman, and Adi Shamir. New Attacks on Keccak-224 and Keccak-256. In Canteaut [9], pages 442–461.
13. Ming Duan and Xuajia Lai. Improved Zero-Sum Distinguisher for Full Round Keccak-f Permutation. Cryptology ePrint Archive, Report 2011/023, 2011.
14. Alexandre Duc, Jian Guo, Thomas Peyrin, and Lei Wei. Unaligned Rebound Attack: Application to Keccak. In Canteaut [9], pages 402–421.
15. Carlo Harpes and James L. Massey. Partitioning Cryptanalysis. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 13–27. Springer, 1997.
16. Ekawat Homsirikamol, Pawel Morawiecki, Marcin Rogawski, and Marian Srebrny. Security margin evaluation of sha-3 contest finalists through sat-based attacks. In Agostino Cortesi, Nabendu Chaki, Khalid Saeed, and Slawomir T. Wierzchon, editors, *CISIM*, volume 7564 of *Lecture Notes in Computer Science*, pages 56–67. Springer, 2012.
17. Donald E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms, 2nd Edition*. Addison-Wesley, 1981.
18. Tri Van Le, Rüdiger Sparr, Ralph Wernsdorf, and Yvo Desmedt. Complementation-Like and Cyclic Properties of AES Round Functions. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *AES Conference*, volume 3373 of *Lecture Notes in Computer Science*, pages 128–141. Springer, 2004.
19. Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2011.
20. Pawel Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced Keccak. Cryptology ePrint Archive, Report 2012/546, 2012. http://eprint.iacr.org/.
21. 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.
22. Thomas Peyrin. Improved Differential Attacks for ECHO and Grøstl. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 370–392. Springer, 2010.
23. Paul C. van Oorschot and Michael J. Wiener. Improving implementable meet-in-the-middle attacks by orders of magnitude. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 229–236. Springer, 1996.

A Appendix: Details of the Collision Attack on 4-Round Keccak-384

We present the details of the 4-round collision attack on Keccak-384. As stated in Section 6.3, the attack is based on the 2.5-round internal differential characteristic given in Characteristic 3 in Appendix C. In this attack, we choose a rotation index value of $i = 16$ and the total probability of the characteristic is 2^{-24}. However, as in the case of the 3-round attack, the workload required to find a conforming message can easily be reduced from 2^{24} to 2^{12}, exploiting 2^{12} degrees of freedom.

Although the algorithm of the 4-round attack on Keccak-384 is very similar to the 3-round attack algorithm, its analysis is more complicated, since the final internal difference of Characteristic 3 has a relatively high weight of 88 (compared to the weight of 6 of the final internal difference of Characteristic 2). Thus, we have more options to allocate variables in order to extend the characteristic. In particular, we have a large number of 20 non-sparse rotated row sets, whose 4 Sboxes assume (exactly) 3 values. However, it is easy to verify that in 18 out of the 20 non-sparse rotated row sets, the possible values of $\hat{u}$, restricted to these rotated row sets, are still contained in an affine subspace whose dimension is at most 4.

Exploiting this observation, we can allocate 140 variables for all but 2 rotated row sets in order to extend the final internal difference of Characteristic 3 beyond the third χ mapping. We have only 2 remaining rotated row sets, which can assume $32 \cdot 32 = 2^{10}$ values, and thus contribute a multiplicative factor of at most 2^{10} to the output subset size.[7] The basic bound for $n = 384$ and $i = 4$ is equal to $2^{8 \cdot 16} = 2^{128}$ (as calculated in Section 5.2), and the dimension of the first $n' = 512$ linear expressions in the 140 variables is 132. In total, we obtain a bound of $2^{10+128+132} = 2^{270}$ on the size of the output subset, and the expected time complexity of the attack is thus bounded by $2^{12} \cdot 2^{270/2} = 2^{147}$ (see Figure 1). This is 2^{45} times faster than the 2^{192} complexity of the birthday attack.

Finally, we have to verify that we have sufficiently many degrees of freedom to find a collision. Indeed, we have to try about 2^{147} initial states, while we have $2^{-12} \cdot 2^{r \cdot (i/64)-2} = 2^{196}$ states that satisfy the constraints.

B Appendix: Details of the Collision Attack on 5-Round Keccak-256

In this appendix, we describe the details of the collision attack on 5-Round Keccak-256. We start by giving a short overview of the TDA (more details can be found in [12]), and then describe its closely related variant, the TIDA.

B.1 The Target Difference Algorithm

The main element that motivates the TDA is the large number of degrees of freedom for Keccak-224 and Keccak-256. This observation allows the algorithm to restrict the set of solutions to a smaller subset that can be found relatively easily. The 1600-bit target difference that enters the algorithm is denoted by Δ_T, while the difference in the initial state is denoted by Δ_I. The algorithm fixes the input difference to the first Sbox layer $L(\Delta_I)$, which also fixes Δ_I, and implies

[7] The value of the 2 rotated row sets does not influence the basic bound and the bound computed using the symbolic calculation, and thus we set them to zero in order to proceed with the calculation of the bound on the output subset size. However, we need to multiply the result of the calculation by a factor of 2^{10}.

that all the message pairs that the algorithm outputs have the same difference. Given $L(\Delta_I)$ at the input to the first Sbox layer and Δ_T at its output, the set of values that satisfy the transition is an affine subspace (due to the fact that the algebraic degree of χ is only 2), and hence the algorithm actually outputs an affine subspace of message pairs. Thus, the algorithm has two phases, where in the first phase (called the difference phase) it fixes Δ_I, and in the second phase (called the value phase) it outputs the affine subspace of message pairs by solving a system of linear equations. Hence, in the difference phase the number of degrees of freedom is reduced by fixing $L(\Delta_I)$, while in the value phase there is no loss of degrees of freedom.

In the difference phase, the TDA tries to simultaneously satisfy two constraints: the first constraint restricts Δ_I to the initial state of the Keccak permutation (i.e., fixes the state bits that the adversary does not control to zero). The second difference constraint ensures that the difference transition $L(\Delta_I) \rightarrow \Delta_T$ is possible. Similarly, in the value phase, the TDA tries to simultaneously satisfy two value constraints. The first constraint restricts the value of the initial state of the messages to the initial state of the Keccak permutation. The second value constraint ensures that the difference transition $L(\Delta_I) \rightarrow \Delta_T$ occurs by restricting the values of the states before the first Sbox layer to the particular affine subspace defined by the transition. Consequently, both value constraints can be formulated using a set of linear equations, while the second difference constraint cannot.

Although there are many solutions for most target differences, for many target differences which have a large number of inactive Sboxes, the difference phase has no solutions, and thus there are no solutions at all: every inactive Sbox in Δ_T, restricts all the 5 corresponding bits of $L(\Delta_I)$ to zero (whereas an active Sbox gives several options for the values of these 5 bits). Consequently, a large number of inactive Sboxes restricts many bits of $L(\Delta_I)$ to zero and as a result the two difference constraints may not be simultaneously satisfiable. On the other hand, the dimension of the affine space of message pairs outputted by the algorithm is expected to increase with the number of inactive Sboxes in Δ_T, given that the algorithm finds a valid Δ_I in the difference phase. This is because inactive Sboxes do not place any constraint of the actual values of the message pairs before the first Sbox layer, whereas active Sboxes restrict them to a smaller affine subspace, reducing the dimension the solution space.

B.2 The Target Internal Difference Algorithm

Given a *target internal difference* $[i, t_1]$ after the first Keccak Sbox layer, our goal is to find messages M such that $\chi \circ L(\bar{M}) \in [i, t_1]$. The first step in constructing the algorithm is to choose the values of the rotation index $i \in \{1, 2, 4, 8, 16, 32\}$ and the rate r so that the algorithm will have sufficiently many degrees of freedom: since the number of linear equations which define an internal difference

$[i, v]$ is[8] $1600 - 25i$ and $|M| = r - 2$, we have $r + 25i - 1602$ degrees of freedom. It is easy to check that we have a positive number of degrees of freedom only for $i = 32$ and for the Keccak versions with $n \in \{224, 256, 384\}$: we have 350, 286 and 30 degrees of freedom for these output sizes, respectively.[9] For $i = 32$, the state is split into two halves and the target internal difference specifies the difference between them. This closely resembles the case where the target difference specifies the difference between two full states, and allows us to easily adapt the techniques used by the TDA to our new TIDA.

We denote the internal difference of the initial state by $[32, t_0]$ and the internal difference after the application of L by $[32, t_{0.5}]$. Analogously to the TDA, our new variant fixes $[32, t_{0.5}]$, which also fixes $[32, t_0]$, and implies that all the messages that the algorithm outputs are in the same internal difference. Moreover, given $[32, t_{0.5}]$ and $[32, t_1]$, the set of values that satisfy each difference transition for a rotated row set (i.e., the input difference and the output difference of the two rows) is an affine subspace. Consequently, the TIDA outputs an affine subspace of messages.

Similarly to the TDA, the TIDA has two phases, where in each phase we try to simultaneously satisfy two constraints. The details of the TIDA are a straightforward adaptation of those of the TDA (which are given in [12]), and are thus not described in this paper.

B.3 Constructing and Extending the Internal Differential Characteristic

In order to exploit the TIDA, we use an internal differential characteristic starting from the second Keccak round, and link it to the initial state of Keccak using the TIDA. The considerations in constructing the characteristic that is used in this attack are different from those that are used to construct the characteristics used in Section 6: first, the initial internal difference in the characteristic used in this section starts from the second Keccak round and hence does not have to conform to the initial state of Keccak. In addition, the characteristic must handle different ι constants. The internal differential characteristic that we use for the attack is given in Characteristic 4 in Appendix C. This characteristic spans 2.5 rounds of the permutation (starting from the second round) and has a probability of 2^{-37}. Note that since the characteristic starts from the second round, its initial internal difference is obtained after applying the first round ι mapping to the target internal difference.

We use the techniques of Section 5.2 in order to bound the size of the output subset after additional 1.5 rounds: we use the variable allocation technique on the final internal difference of the characteristic (whose weight is 12) to allocate

[8] The internal difference $[i, v]$ equates each bit of the state (which is not in the first CSS) to its symmetric counterpart in the first CSS. Since a CSS contains $25i$ bits, $[i, v]$ defines $1600 - 25i$ equations.

[9] However, we note that for Keccak-384 we only have a few dozens of degrees of freedom, which are insufficient for our algorithm.

24 variables and extend the characteristic beyond the χ mapping of the fourth round. The basic bound on the size of the output subset is $2^{5i} = 2^{160}$, and the dimension of the first $n' = 320$ linear expressions is the maximal possible dimension of 24. This gives a bound of $2^{160+24} = 2^{184}$ on the size of the output subset. Since the probability of the characteristic is 2^{-37}, the time complexity bound of the simple collision attack given at the beginning of Section is at least $2^{37+184/2} = 2^{129}$, even if we assume that the amortized time complexity of finding one message that satisfies the target internal difference is less than 1. Thus, even under optimal assumptions, this time complexity is slightly worse than the expected time complexity of 2^{128} of generic collision attacks for Keccak-256. In order to speed up the attack, we have to use additional techniques, but before describing them, we analyze the execution of the TIDA on the specific target internal difference derived from Characteristic 4.

B.4 Executing the Target Internal Difference Algorithm

The target internal difference calculated from Characteristic 4 has a relatively large number of 26 inactive rotated row sets. Analogously to the TDA, a large number of inactive rotated row sets may make the difference phase of the TIDA difficult to solve, but if we manage to find a solution, then the dimension of the affine message subspace outputted by the algorithm is expected to be large.

In this attack, we choose the target internal difference carefully such that the TIDA should be able to solve it in a reasonably short time, and still output an affine subspace of a relatively large dimension. To check this property, we performed hundreds of simulations of the TIDA with the chosen target internal difference on a standard PC, each time feeding the algorithm with fresh randomness. In all of our simulations, the TIDA was able to solve the challenge in less than 30 seconds, while outputting an affine subspace whose dimension ranged between 78 and 111, and was typically around 90. Thus, according to our simulations, the amortized time complexity of finding one message that satisfies the target internal difference is significantly less than 1. We provide concrete evidence for the success of our simulations by giving an example of a message that satisfies Characteristic 4 in Message 1 in Appendix D.

B.5 Speeding Up the 5-Round Attack on Keccak-256 Using Message Modification

In order to speed up the attack, we reduce the amortized time that is required in order to find a message that satisfies the internal difference of Characteristic 4 after round 2 of the Keccak permutation. Since the probability of the first χ transition in Characteristic 4 is 2^{-21}, the amortized time to find such a message is about 2^{21} in the attack given at the beginning of Section B.3, and we reduce it by using message modification within the subspace of messages outputted by the TIDA.

The main observation that we use for the message modification is that we can easily find a basis for the typical subspace outputted by the TIDA, which

contains many low Hamming weight 1600-bit vectors of a certain type. These vectors are non-zero vectors of the lowest Hamming weight which are symmetric (i.e., in the zero internal difference for $i = 32$) and also have even column parity. We call these vectors low Hamming weight symmetric even parity vectors, or LHSE vectors in short. Since a LHSE vector contains 2 non-zero columns, each with a Hamming weight of 2, it has a total Hamming weight of 4, and it is easy to check that there are exactly 1600 such vectors.

The reason that a typical subspace outputted by the TIDA contains many LHSE vectors can be explained as follows: take an arbitrary initial state w from the subspace outputted by the TIDA, and assume that we add to it a LHSE state w' such that the result $w + w'$ is a valid initial state of the Keccak permutation (note that we can choose many such vectors from the 1600 LHSE vectors). In order for $w+w'$ to also be outputted by the TIDA, it must be in the same internal difference as w, which is true since w' is symmetric. In addition, $w + w'$ must satisfy the target internal difference after the first χ mapping. This occurs with a reasonably high probability since w satisfies it, and $L(w) + L(w + w') = L(w')$ is a symmetric vector with a low Hamming weight of 4. Thus, $(\chi \circ L(w)) + (\chi \circ L(w+w'))$ has a good chance to be symmetric, implying that $w+w'$ also satisfies the target internal difference after the first χ mapping.

The message modification algorithm uses a basis of the output of the TIDA with many LHSE vectors. Since there are only 1600 LHSE vectors, we can compute such a basis S exhaustively from the original basis outputted by the TIDA, and we denote by $S_1 \subseteq S$ the set of LHSE vectors in S.[10] To simplify our notations, given an initial state w, we define the predicate $\phi(w)$ to be true if w satisfies the internal difference of Characteristic 4 after round 2 of the Keccak permutation. Assume that we found a vector w in the output subspace of the TIDA such that $\phi(w)$ holds. Let $w' \in S_1$, then the Hamming distance of $R(w)$ and $R(w + w')$ is small, implying that the Hamming distance of the states $L(R(w))$ and $L(R(w + w'))$ (which are input to the second χ mapping) is still reasonably small (although larger due to L). Thus, since $\phi(w)$ holds and the second χ transition does not depend on the values of many bits of $L(R(w))$, there is a good chance that those bits have the same value in $L(R(w + w'))$, implying that $\phi(w + w')$ holds as well. Moreover, if $\phi(w'')$ holds for another vector $w'' \in S_1$, then there is a good chance that $\phi(w + w' + w'')$ holds as well. All these heuristic arguments imply that when we find w such that $\phi(w)$ holds, it is a good idea to test additional vectors in the following carefully selected subspace:

1. Choose an arbitrary initial state w from the subspace spanned by S.
2. Check if $\phi(w)$ holds, if not go to step 1.
3. Let $S_2 = \{w' \in S1 | \phi(w + w') \text{ holds}\}$.
4. Iterate over all states w'' spanned by S_2, and check for each state $w + w''$ if it satisfies Characteristic 4. If so, store its output and check for a collision, as in the basic attack.

[10] Based on extensive simulations, the size of S_1 is typically more than half of the size of S

In order to get a realistic estimate of the amortized time that is required in order to find a message that satisfies the internal difference of Characteristic 4 after round 2 of the Keccak permutation, we performed simulations on hundreds of subspaces which were actually returned by the TIDA. For each such subspace (whose typical dimension is about 90), we tested a total number of about 2^{30} messages using the message modification algorithm, out of which in a typical subspace, about 2^{23} messages satisfied the internal difference of Characteristic 4 after round 2. Thus, we estimate that the amortized time required in order to find a message that satisfies the internal difference of Characteristic 4 after round 2 to be about $2^{30-23} = 2^7$, which gives an improvement factor of 2^{14} compared to the 2^{21} amortized time in the basic attack given at the beginning of Section B.3. We note that there are several possible improvements to this message modification algorithm, which may speed up the attack further (up to the maximal possible factor of 2^{21}). For example, we can discard subspaces outputted by the TIDA for which the size of the set S_1 is small. Another improvement possibility is to use the TIDA states to carefully "correct" states w for which $\phi(w)$ does not hold, and thus efficiently find states for which ϕ holds in the first step of the algorithm. However, we decided not to include these potential improvements in our claimed results, due to their highly speculative nature and the fact that even without them we are already well below the birthday threshold.

B.6 The Full Attack

As noted above, the message modification technique improves the 2^{129} time complexity of the basic attack by a multiplicative factor which is between 2^{14} and 2^{21}, and thus we estimate the time complexity of the full attack to be between 2^{108} and 2^{115}. Since the bound on the size of the output subset is 2^{184}, the memory complexity is about $2^{184/2} = 2^{92}$.

Finally, we have to verify that we have sufficiently many degrees of freedom in order to find a collision. Namely, we have to make sure that we can find about 2^{129} messages conforming to the target internal difference (even though when we use message modification we do not test all of them). Since the subspace of maximal dimension that we found using the TIDA has a dimension of 111 (which is smaller than 129), we have to execute the TIDA many times in order to collect sufficiently many messages. In Keccak-256 we have 286 degrees of freedom, hence we do not expect this to be a problem. However, we can also deal with the unlikely case that the TIDA is able to find only a very small set of all the possible solutions:[11] in this case we use the additional degrees of freedom in messages containing multiple blocks. According to our simulations, in about 20% of the cases when the c bits of the inner part of the state are chosen at random (e.g., by applying the Keccak permutation to an arbitrary single-block message), the TIDA (and the message modification algorithm) perform very similarly to the case where the c bits are zero in the initial permutation state.

[11] We can easily detect this case since we store the messages that satisfy Characteristic 4.

The 20% success rate gives us hundreds of additional degrees of freedom, which are definitely sufficient when our goal is to find a single collision.

C Appendix: Internal Differential Characteristics for Keccak

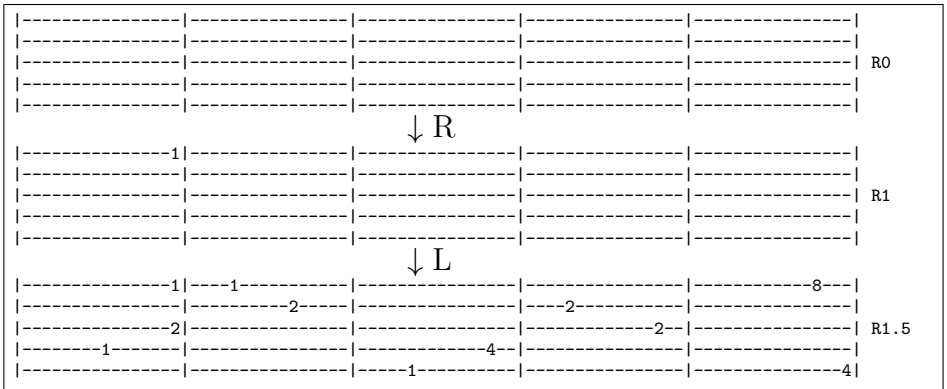
In this section, we provide the precise internal differential characteristics (labeled as Characteristics 1–4) which we use in our collision attacks on round-reduced Keccak.

An internal difference $[i, v]$ is represented by a state with the lowest Hamming weight. Each state is given as a matrix of 5×5 lanes of 64 bits, ordered from left to right, where each lane is given in hexadecimal using the little-endian format. The symbol '-' is used in order to denote a zero 4-bit value.

The internal differential characteristics are given as a sequence of internal differences. The operation performed in each transition is specified between the representative states and round numbers are specified to the right of the states.

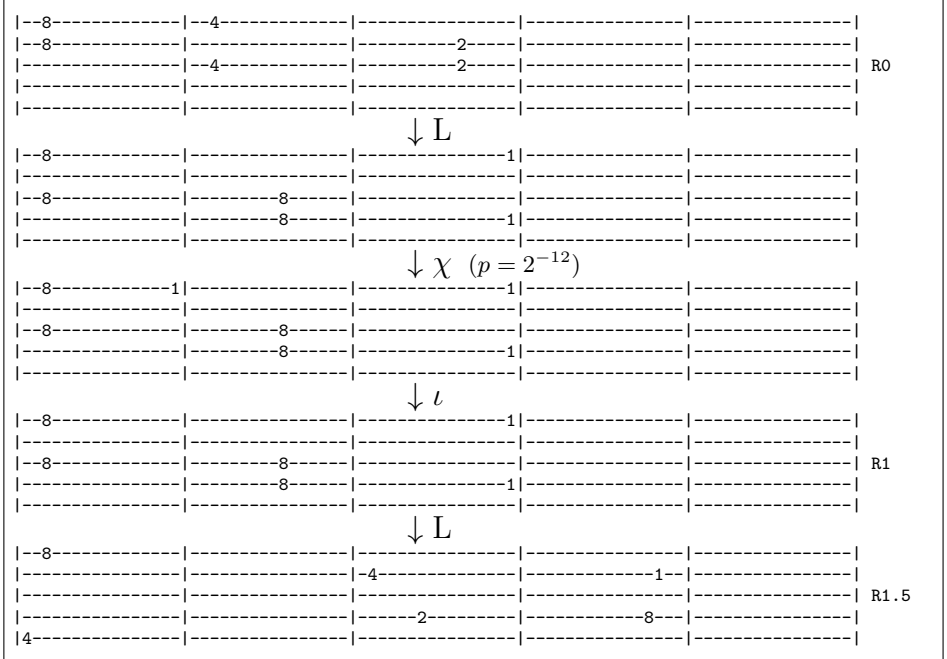
D Appendix: Examples of Actual Collisions and a Message Conforming to Characteristic 4

We give examples of actual collisions for three rounds of Keccak-384 and Keccak-512 (labeled as Collisions 1, 2), and a message conforming to Characteristic 4 (labeled as Message 1). The padded messages and output values are given in blocks of 32-bits ordered from left to right, where each block is given in hexadecimal using the little-endian format.



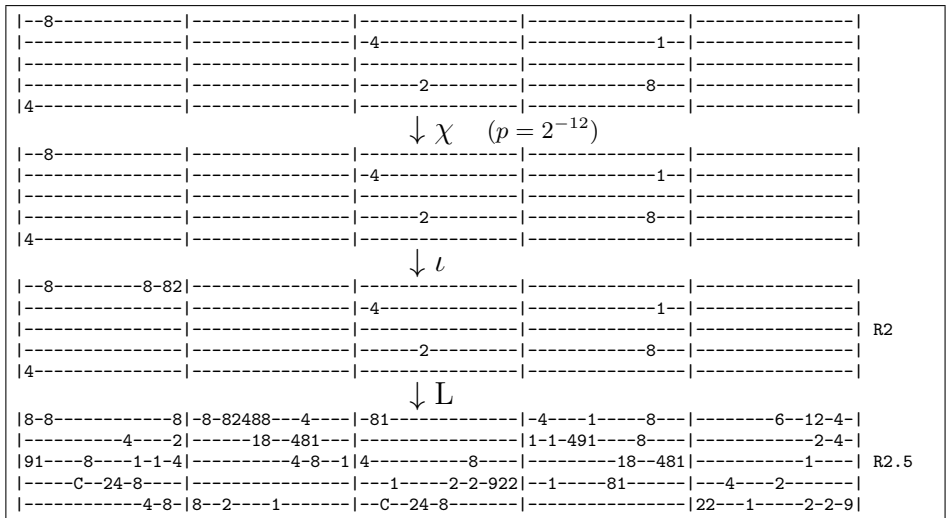
The characteristic has a rotation index value of $i = 4$, as described in Section 6.1.

Characteristic 1: The 1.5-round internal differential characteristic with probability 1 used in order to find collisions in 3-round Keccak-512



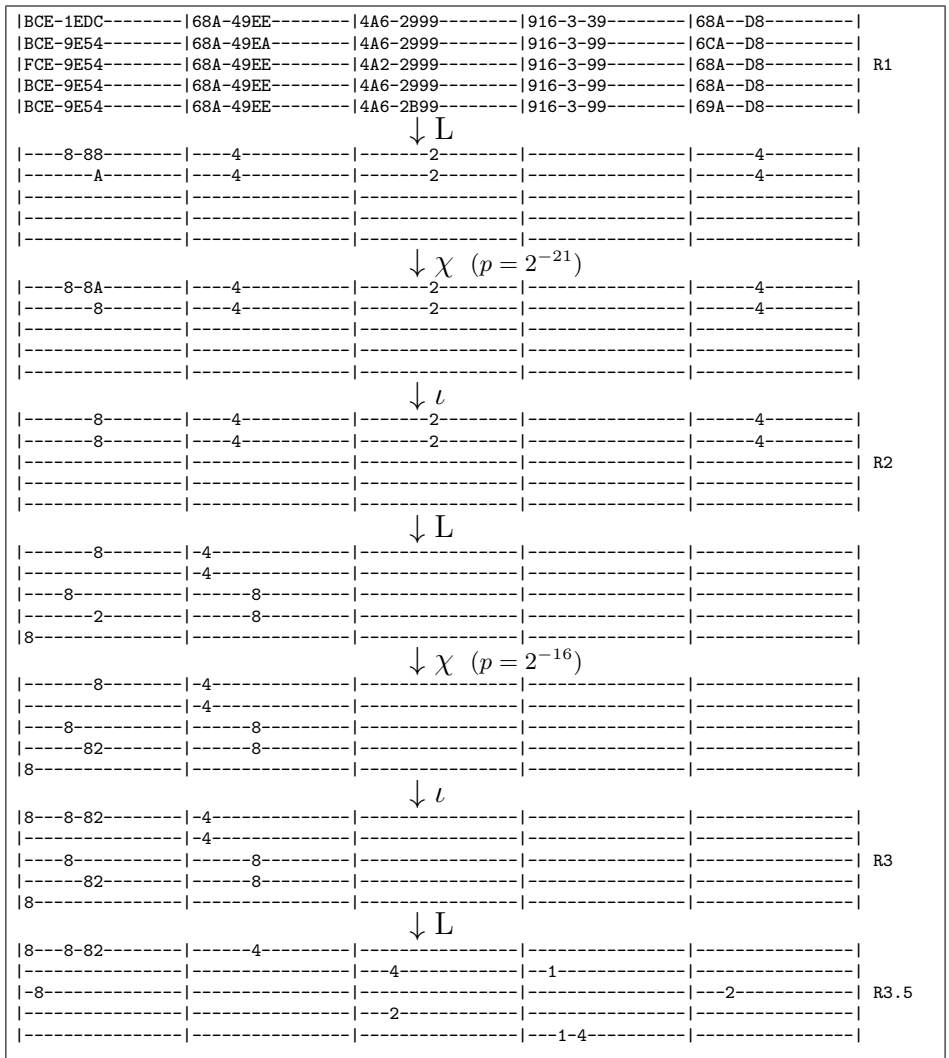
The characteristic has a rotation index value of $i = 4$ for the 3-round attack on Keccak-384, as described in Section 6.2.

Characteristic 2: The 1.5-round internal differential characteristic with probability 2^{-12} used in order to find collisions in 3-round Keccak-384



The characteristic has a rotation index value of $i = 16$ (this applies to the full 2.5-round characteristic used in the 4-round attack) and probability 2^{-12}, as described in Section 6.3. The total probability of the full 2.5-round characteristic is 2^{-24}.

Characteristic 3: The 1-round extension of Characteristic 2 used in the collision attack on 4-round Keccak-384



The internal differential characteristic has a rotation index value of $i = 32$, as described in Section B.

Characteristic 4: The characteristic with probability 2^{-37} used in the 5-round collision attack on Keccak-256

```
M1=

88888888 88888888 66666666 66666666 AAAAAAAA AAAAAAAA 77777777 77777777 BBBBBBBB BBBBBBBB
BBBBBBBB BBBBBBBB 11111111 11111111 88888888 88888888 CCCCCCCC CCCCCCCC

M2=

AAAAAAAA AAAAAAAA 88888888 88888888 EEEEEEEE EEEEEEEE 99999999 99999999 99999999 99999999
99999999 99999999 88888888 88888888 CCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC

Output=

56BCC94B C4445644 D7655451 5DD96555 71FA7332 3BA30B23 958408C5 64407664 41805414 11190901
6ABAA8BA A8ABAEFA 7EF8AEEE ECCE68DC 4EC8ACEC DD5D5CCC
```

The messages were found using Characteristic 1.

Collision 1: A collision in 3-round Keccak-512

```
M1=

FFFFFFFF FF7FFFFF BBBBBBBB BBFBBBBB 44444444 44444444 FFFFFFFF FFFFFFFF 99999999 99999999
44444444 44C44444 44444444 44444444 44644444 44444444 AAAAAAAA AAAAAAAA 66666666 66666666
44444444 44444444 DDDDDDDD DD9DDDDD DDFDDDDD DDDDDDDD

M2=

33333333 33B33333 55555555 55155555 AAAAAAAA AAAAAAAA 77777777 77777777 44444444 44444444
66666666 66E66666 EEEEEEEE EEEEEEEE 11311111 11111111 CCCCCCCC CCCCCCCC FFFFFFFF FFFFFFFF
11111111 11111111 99999999 99D99999 DDFDDDDD DDDDDDDD

Output=

99999991 11199999 4440C444 405C60DC 00000000 0C100010 777677F7 73F77767 3550F597 55D57155
66666664 66666666
```

The messages were found using Characteristic 2.

Collision 2: A collision in 3-round Keccak-384

```
M=

30DFA98A B8CF5C2F A9CB0FDC 220346FD 311F1F9C 619AE337 51E70A2E FFEE7106 235FFE41 F6356D1E
7420E3BE ABC66CF9 4DB24896 A33188ED F77C08E9 1CFDE28A 27947A3B 676C430D AA73B6BE A0D27BE7
DCD08DE8 CBC00830 D58570AB 4F139EDA 81CFF59E 9E0BDA5B 73BDBE0F D3E314D9 AF9458E0 223EF917
8712F441 621C4F1F 8D0C1C43 CD89C138
```

Message 1: A message conforming to Characteristic 4