Unaligned Rebound Attack: Application to Keccak

Alexandre Duc[1], Jian Guo[2], Thomas Peyrin[3], and Lei Wei[3]

[1] Ecole Polytechnique Fédérale de Lausanne, Switzerland
[2] Institute for Infocomm Research, Singapore
[3] Nanyang Technological University, Singapore
alexandre.duc@epfl.ch
{ntu.guo,thomas.peyrin}@gmail.com
wl@pmail.ntu.edu.sg

Abstract. We analyze the internal permutations of Keccak, one of the NIST SHA-3 competition finalists, in regard to differential properties. By carefully studying the elements composing those permutations, we are able to derive most of the best known differential paths for up to 5 rounds. We use these differential paths in a rebound attack setting and adapt this powerful freedom degrees utilization in order to derive distinguishers for up to 8 rounds of the internal permutations of the submitted version of Keccak. The complexity of the 8 round distinguisher is $2^{491.47}$. Our results have been implemented and verified experimentally on a small version of Keccak. This is currently the best known differential attack against the internal permutations of Keccak.

Key words: Keccak, SHA-3, hash function, differential cryptanalysis, rebound attack.

1 Introduction

Cryptographic hash functions are used in many applications such as digital signatures, authentication schemes or message integrity and they are among the most important tools in cryptography. Informally, a hash function H is a function that takes an arbitrarily long message as input and outputs a fixed-length hash value of size n bits. Even if hash functions are traditionally used to simulate the behavior of a random oracle [3], classical security requirements are collision resistance and (second)-preimage resistance. Namely, it should be impossible for an adversary to find a collision (two distinct messages that lead to the same hash value) in less than $2^{n/2}$ hash computations, or a (second)-preimage (a message hashing to a given challenge) in less than 2^n hash computations. Of course, in the ideal case an attacker should also not be able to distinguish the hash function from a random oracle.

Recently, most of the standardized hash functions [23, 20] have suffered from serious collision attacks [27, 26]. As a response the NIST launched in 2008 the SHA-3 competition [21] that will lead to the future hash function standard. 5 candidates made it to the final round, and Keccak [9] is among them. Compared to its opponents, this hash function presents the particularity to be a sponge function [5]. The submitted versions of Keccak to the SHA-3 competition use as main component an internal permutation P of 1600 bits. In the original submission [6] the internal permutation used 18 rounds and the tweaked versions [7] went up to 24 rounds.

Like any construction that builds a hash function from a subcomponent, the cryptographic quality of this internal permutation is very important for a sponge construction. Therefore, this permutation P should not present any structural flaw, or should not be distinguishable from a randomly chosen permutation. Previous cryptanalysis have not endangered the Keccak security so far. Zero-sum distinguishers [2] can reach an important number of rounds, but generally with a very high complexity. For example, the latest results [11] provide zero-sum partitions distinguishers for the full 24-round 1600-bit internal permutation with a complexity of 2^{1590}. When looking at smaller number of rounds the complexity would decrease, but it is unclear how one can describe the partition of a 1600-bit internal state without using the Keccak round inside the definition of the partition. Moreover, such zero-sum properties seem very hard to exploit when the attacker aims at the whole hash function. On the other side, more classical preimage attack on 3 round using SAT-solvers have been demonstrated [19]. Finally, Bernstein recently published [4] a $2^{511.5}$

computations (second)-preimage attack on 8 rounds that allows a workload reduction of only half a bit over the generic complexity with an important memory cost of 2^{508}.

Our contributions. In this article, we analyze the differential cryptanalysis resistance of the KECCAK internal permutation. More precisely, we first introduce a new and generic method that looks for good differential paths for all the KECCAK internal permutations, and we obtain the currently best known differential paths. We then describe a simple method to utilize the available freedom degrees which allows us to derive distinguishers for reduced variants of the KECCAK internal permutations with low complexity. Finally, we apply the idea of rebound attack [18] to KECCAK. This application is far from being trivial and requires a careful analysis of many technical details in order to model the behavior of the attack. This technique is in particular much more complicated to apply to KECCAK than to `AES` or to other 4-bit Sbox hash functions [22, 14]. The model introduced has been verified experimentally on a small version of KECCAK and we eventually obtained differential distinguishers for up to 8 rounds of the submitted version of KECCAK to the `SHA-3` competition. In order to demonstrate why differential analysis is in general more relevant than zero-sum ones in regards to the full hash function, we applied our techniques to the recent KECCAK challenges [25] and managed to obtain the currently best known collision attack for up to two rounds.

Outline. In Section 2, we first briefly describe the KECCAK family of hash functions. We describe our differential path search algorithm in Section 3 and we derive simple differential distinguishers from it in Section 4. We present our theoretical model and we apply the rebound attack on KECCAK in Section 5. We show how to reduce the complexity of the attack in Section 6. Finally, we present our results and draw conclusions in Section 7.

2 The KECCAK Hash Function Family

KECCAK [9, 10] is a family of variable length output hash functions based on the sponge construction [5]. In KECCAK family, the underlying function is a permutation chosen from a set of seven KECCAK-f permutations, denoted as KECCAK-$f[b]$ where $b \in \{1600, 800, 400, 200, 100, 50, 25\}$ is the permutation width as well as the internal state size of the hash function. The KECCAK family is parametrized by an r-bit bitrate and c-bit capacity with $b = r + c$.

A fixed length n-bit hash value for a KECCAK hash function is obtained by truncating the output of the hash function to the first n bits and this function is denoted by KECCAK$_n$. For these variants, if at least 2^n (second) preimage resistance security is desired then the parameter c is chosen such that $c \geq 2n$. The proposed version of KECCAK for the SHA-3 standardization uses KECCAK-$f[1600]$ as internal permutation. The default variant of KECCAK family is denoted by KECCAK-[] and it has parameters $r = 1024$, $c = 576$ for any output length n.

2.1 The domain extension algorithm

Sponge is an iterated construction to build a function $F : \{0, 1\}^* \to \{0, 1\}^*$ by using a fixed length transformation or permutation f whose width is fixed to b bits. This construction operates on a state of b. The initial value of the state is zero and the input message is padded such that it is a multiple of r-bit message blocks (and the last message block is different than zero).

Message processing using sponge construction has two stages: absorbing phase and squeezing phase. In the absorbing phase, r-bit message blocks are xor-ed with the first r bits of the b-bit state, interleaved with the applications of f until all blocks are processed. In the squeezing phase, depending on the required number of output bits n, the first r bits of the state are returned as output blocks, interleaved with the applications of f until n bits are returned.

2.2 The KECCAK-f permutations

The internal state of the KECCAK family can be viewed as a bit array of 5×5 lanes, each of length $w = 2^\ell$ where $\ell \in \{0, 1, 2, 3, 4, 5, 6\}$ and $b = 25w$. The state can also be described as a three dimensional array of bits defined by $a[5][5][w]$. A bit position (x, y, z) in the state is given by $a[x][y][z]$ where x and y coordinates are taken over modulo 5 and the z coordinate is taken over modulo w. A *lane* of the internal state at *column* x and *row* y is represented by $a[x][y][\cdot]$, while a *slice* of the internal state at *width* z is represented by $a[\cdot][\cdot][z]$.

Keccak-$f[b]$ is an iterated permutation consisting of a sequence of n_r rounds indexed from 0 to $n_r - 1$ and the number of rounds are given by $n_r = 12 + 2\ell$. A round $\mathbf{R}$ consists of a transformation of five step mappings and is defined by:

$$\mathbf{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

These step mappings are discussed below where the operation + indicates bitwise addition.

$\boldsymbol{\theta}$ mapping. This linear mapping intends to provide diffusion for the state and is defined for every x, y and z by:

$$\theta : a[x][y][z] \leftarrow a[x][y][z] + \bigoplus_{y'=0}^{4} a[x-1][y'][z] + \bigoplus_{y'=0}^{4} a[x+1][y'][z-1]$$

That is, the bitwise sum of the two columns $a[x-1][\cdot][z]$ and $a[x+1][\cdot][z-1]$ is added to each bit $a[x][y][z]$ of the state.

$\boldsymbol{\rho}$ mapping. This linear mapping intends to provide diffusion between the slices of the state through intra-lane bit translations. For every x, y and z:

$$\rho : a[x][y][z] \leftarrow a[x][y][z + T(x, y)]$$

where $T(x, y)$ is a translation constant. That is, all bit positions in each lane are translated by a constant amount that depends on the column x and row y considered.

$\boldsymbol{\pi}$ mapping. This linear mapping intends to provide diffusion in the state through transposition of the lanes. More precisely, it is defined for every x, y and z as:

$$\pi : a[x'][y'][z] \leftarrow a[x][y][z], \text{ with } \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Since this results in transposition of bits into a same slice, this mapping is an intra-slice transposition.

$\boldsymbol{\chi}$ mapping. This is the only non-linear mapping of Keccak-f and is defined for every x, y and z by:

$$\chi : a[x][y][z] \leftarrow a[x][y][z] + ((\neg a[x+1][y][z]) \wedge a[x+2][y][z])$$

This mapping is similar to an Sbox applied independently to each 5-bit row of the state and can be computed in parallel to all rows. We represent by $s = 5w$ the number of Sboxes/rows in Keccak internal state. Here $\neg$ denotes bit-wise complement, and $\wedge$ the bit-wise AND.

$\boldsymbol{\iota}$ mapping. For every round $\mathbf{R}$ of the Keccak-f permutation, this mapping adds constants derived from an LFSR (see [9] for details) to the lane $a[0][0][\cdot]$. These constants are different in every round i:

$$\iota : a[0][0][\cdot] \leftarrow a[0][0][\cdot] + \mathrm{RC[i]}$$

This mapping aims at destroying the symmetry introduced by the identical nature of the remaining mappings in every round of the Keccak-f permutation.

We refer to the Keccak specifications document [9] for all the translation and round constants.

3 Finding Differential Paths for KECCAK-*f* Internal Permutations

Before describing how we use the freedom degrees in a rebound attack setting, we first study how to find "good" differential paths for all KECCAK variants. In this section, we describe our differential finding algorithms. We start by recalling several special properties of the mappings θ and χ in the round function, followed by our algorithm which provides most of the best known differential paths for the KECCAK internal permutations. In particular, we provide the currently best known 3, 4 and 5-round differential paths for KECCAK-f[1600], the internal permutation from the submitted version of KECCAK.

3.1 Special properties of θ and χ

The θ mapping updates each state bit $a[x][y][z]$ by adding the bitwise sum of the two columns $a[x-1][.][z]$ and $a[x+1][.][z-1]$. When every column sums to 0, θ acts as identity. This is noted by the KECCAK designers [9, Section 2.4.3], and the set of the states with all columns sums to 0 is called *column parity kernel*, or *CP-kernel* for short. Since θ is linear, this property applies not only to the state values, but also to differentials. While θ expands a single bit difference into at most 11 bits (2 columns and the bit itself), it has no influence, *i.e.,* acts as identity, on differences in the CP-kernel. This property will be intensively used in finding low Hamming weight bitwise differentials. Another interesting property is that θ^{-1} diffuses much faster than θ, *i.e.,* a single bit difference can be propagated to about half state bits through θ^{-1} [9, Section 2.3.2]. However, the output of θ^{-1} is extremely regular when the Hamming weight of the input is low.

The χ layer updates each bit $a[x, y, z]$ by adding $((\neg a[x+1, y, z]) \wedge a[x+2, y, z])$. It is a row-wise operation and thus can also be viewed as a 5-bit Sbox. Similar to the analysis of other Sboxes, we build the differential distribution table (DDT), as shown in Table 3 in Appendix B. We remark that when a single difference is present, χ acts as identity with the best probability 2^{-2}, while input differences with more active bits tends to lead to more possible output differences, but with lower probability. It is also interesting to note that given an input difference to χ, all possible output differences occur with same probability, however this is not the case for χ^{-1} (the DDT for χ^{-1} can be derived from Table 3).

3.2 First tools

Our goal is to derive "good" bitwise differential paths by maintaining the bit difference Hamming weight as low as possible. The ι permutation adds predefined constants to the first lane, and hence has no essential influences when such differentials are considered. For the rest of the paper, we will ignore this layer. We note that θ, ρ and π are all linear mappings, while χ acts as a non-linear Sbox. Furthermore, ρ and π do not change the number of active bits in a differential path, but only bit positions. Hence, θ and χ are critical when looking for a "good" differential path. Since χ is followed by θ in the next round (ignoring ι), we consider these two mappings together by treating a slice of the state as a unit, and try to find the potential best mapping of the slice through χ with the following two rules.

1. Given an input difference of the slice, *i.e.,* 5 row differences, find all possible output differences by looking into the DDT table. Then among all combinations of solutions of the 5 rows, choose the output combinations with minimum number of columns with odd parity.
2. In case of a draw, we select the state with the minimum number of active bits.

Rule 1 aims at reducing the amount of active bits after applying θ by choosing each slice of the output of the χ closest to the CP-kernel (*i.e.,* with even parity for most columns), and rule 2 further reduces the amount of active bits within the columns. Although this strategy may not lead to the minimum number of active bits after θ in the entire state (the full KECCAK-f[1600] state is too large to precompute the best mappings for the whole state), it finds the best slice-wise mappings with help of a table of size 2^{25} (tricks like removing the ordering of the rows reduce the table size to about 2^{18}). We call this table χ-*slice-table*.

3.3 Algorithm for differential path search

Denote $\lambda = \pi \circ \rho \circ \theta$ (all linear mappings), and the state at round i before (resp. after) applying the linear layer λ as a_i (resp. b_i). We start our search from a_1, *i.e.,* the input state to the second round, and compute backwards for one round, and few rounds forwards, as shown below.

$$a_0 \xleftarrow{\lambda^{-1}} b_0 \xleftarrow{\chi^{-1}} \mathbf{a_1} \xrightarrow{\lambda} b_1 \xrightarrow{\chi} a_2 \xrightarrow{\lambda} b_2 \xrightarrow{\chi} a_3 \xrightarrow{\lambda} b_3 \cdots$$

The forward part is longer than the backward part because the diffusion of θ^{-1} is much better than for θ, thus, it will be easier for us to control the bit differences Hamming weight for several forward rounds (instead of backward rounds).

We choose a_1 from the CP-Kernel. However, it is impossible to enumerate all $(\binom{5}{0} + \binom{5}{2} + \binom{5}{4})^{320} = 2^{1280}$ combinations. We further restrict to a subset of the CP-Kernel with at most 8 active bits and each column having exactly 0 or 2 active bits. Note also that any bitwise differential path is invariant through position rotation along the z axis, so we have to run through a set of size about 2^{36}. A brute-force search on this set using our two rules stated previously finds 3-round differential paths with probability 2^{-32}, 4-round differential paths with probability 2^{-142} and 5-round paths with probability 2^{-709} for Keccak-f[1600], with examples shown in Table 6, 7 and 8 in Appendix D, respectively. We provide also in Table 1 all the best differential path probabilities found for all Keccak internal permutation sizes.

Table 1. Best differential path results for each version of Keccak internal permutations, for 1 up to 5 rounds. The detailed differential paths for Keccak-f[1600] are shown in Appendix D. Paths in bold are new results we found with the method presented in this paper.

b	best differential path probability (successive differential complexity of the rounds)				
	1 rd	2 rds	3 rds	4 rds	5 rds
100	2^{-2} (2)	2^{-8} (4 - 4)	2^{-19} (4 - 8 - 7)	2^{-30} (4 - 8 - 10 - 8)	2^{-54} (4 - 8 - 10 - 8 - 24)
200	2^{-2} (2)	2^{-8} (4 - 4)	2^{-20} (4 - 8 - 8)	2^{-46} (11 - 9 - 8 - 8)	2^{-108} (8 - 16 - 20 - 16 - 48)
400	2^{-2} (2)	2^{-8} (4 - 4)	2^{-24} (8 - 8 - 8)	$\mathbf{2^{-84}}$ (16 - 14 - 12 - 42)	2^{-216} (16 - 32 - 40 - 32 - 96)
800	2^{-2} (2)	2^{-8} (4 - 4)	$\mathbf{2^{-32}}$ (4 - 4 - 24)	$\mathbf{2^{-109}}$ (12 - 12 - 12 - 73)	$\mathbf{2^{-432}}$ (32 - 64 - 80 - 64 - 198)
1600	2^{-2} (2)	2^{-8} (4 - 4)	$\mathbf{2^{-32}}$ (4 - 4 - 24)	$\mathbf{2^{-142}}$ (12 - 12 - 12 - 106)	$\mathbf{2^{-709}}$ (16 - 16 - 16 - 114 - 547)

Comment. In the reference document [9], among other methods, the Keccak designers looked for special differential paths to motivate their choice of the step mappings. In their model, the input states to the χ of the first two rounds are forced to fall entirely inside the CP-kernel. The best 3-round path they could find for Keccak-f[1600] has probability 2^{-35}. Note also that no 4-round and 5-round path is obtained. Their results were up to now the best published differential paths for Keccak. As shown in Table 1, our model outperforms in most cases the Keccak designers model by removing this strong restriction on χ and using a more appropriate slice-wise mapping. As a result, we found the best known 3, 4 and 5-round differential paths for the bigger versions of Keccak.

4 Simple Distinguishers for the Reduced Keccak Internal Permutations

Once the differential paths obtained, we can concentrate our efforts on how to use at best the freedom degrees available in order to reduce the complexity required to find a valid pair for the differential trails or to increase the amount of rounds attacked. We present in this section a very simple method that allows to obtain low complexity distinguishers on a few rounds of the Keccak internal permutations.

4.1 A very simple freedom degrees fixing method

We first describe an extremely simple way of using the available freedom degrees, which are exactly the b-bit value of the internal state (since we already fixed the differential path). For all the best differential paths found from Table 1, we can extend them by one round to the left or to the right, by simply picking some valid Sboxes differential transitions. Obviously, this is going to add a lot of new constraints because the number of active Sboxes will explode in this newly added round and it will force the differential probability to be very low overall. However, we can use our available freedom degrees specifically for this round so that its cost is null. One simply handles each of the active Sboxes differential transitions for this round one by one, independently, by fixing a valid value for the active Sboxes. In terms of freedom degrees consumption for this extra round, in the worst case we have all s Sboxes active and a differential transition probability of 2^{-4} for each of them. Thus, we are ensured to have at least $2^{5s-4s} = 2^s$ freedom degrees remaining after handling this extra round.

Let us assume for example that we start from the 4-round differential path for Keccak-$f[1600]$, with differential probability 2^{-142}. One can obtain a 5 round path by adding one round to the left but the number of conditions for this new round will be huge. Therefore, we simply identify all the active Sboxes for this fifth round and we fix the value for them so that the expected differential transitions are verified. Overall, the attacker finds a valid pair of internal state values for the 5-round differential path only with complexity 2^{142} (we are ensured to have at least 2^{320} freedom degrees to handle the 2^{-142} probability). Note that some more involved freedom degree methods (such as message modification [26]) might even allow to also control some of the conditions of the original differential path, thus further reducing the complexity.

4.2 The generic case

At the present time, we are able to find valid pairs of internal state values for some differential paths on a few rounds with a rather low complexity. Said in other words, we are able to compute internal state value pairs with a predetermined input/output difference. A direct application from this is to derive distinguishers. For a randomly chosen permutation of b bits, finding a pair of inputs with a predetermined difference that maps to a predetermined output difference costs 2^b computations. Indeed, since the input and the output differences are fixed, the attacker can not apply any birthday-paradox technique. Those distinguishers are called "limited-birthday distinguishers" and can be generalized in the following way (we refer to [12] for more details): for a randomly chosen b-bit permutation, the problem of mapping an input difference from a subset of size I to an output difference from a subset of size J requires $\max\{\sqrt{2^b/J}, 2^b/(I \cdot J)\}$ calls to permutation (while assuming without loss of generality since we are dealing with a permutation that $I \leq J$).

Using the freedom degrees technique from the previous section and reading Table 1, we are for example able to obtain a distinguisher for 5 rounds of the Keccak-$f[1600]$ internal permutation with complexity 2^{142} (while the generic case is 2^{1600}).

4.3 Extending the differential path

Since for many of our distinguishers, the gap between our attack and the generic case complexity is very big, we can try to reach a few more rounds without increasing the complexity. Indeed, by analyzing how the differences will propagate forward from the output and backward from the input of our differential path, we will be able to determine the size of the possible input differences set and the possible output differences set.

For the forward case (i.e. when adding a round to the right), we start from the fully determined difference on the output of the differential path. We first apply the linear layers θ, ρ and π on this output difference and we obtain the difference mask at the input of χ. Now, for each active Sbox, knowing exactly its input difference, we can check with the DDT from χ (Table 3 in the Appendix B) that only a subset of the 2^5 possibles output differences can be reached. Therefore, the size Γ^{out} of the set of reachable output differences after applying this extra round is bounded and this bound can be computed exactly using the DDT from χ.

For the backward case (i.e. when adding a round to the left), we start from the fully determined difference on the input of the differential path. Then, reading at the DDT from χ^{-1}, one can check that one active Sbox can produce at most a certain small subset of the 2^5 possible input differences. Therefore, the size Γ^{in} of the set of reachable input differences after inverting this χ layer is bounded and this bound can be computed exactly using the DDT from χ^{-1}. Note that continuing to invert the extra round by computing θ^{-1}, ρ^{-1} and π^{-1} will not modify the size of this set.

To conclude, using a r-round path from Table 1 with differential probability p, we extend it by one more round in order to find valid internal state pairs for this new $(r+1)$-round differential path with p^{-1} computations (see Section 4.1). Then, using the limited-birthday distinguishers, one can derive a $(r+3)$-round distinguisher for the Keccak internal permutation with complexity p^{-1}, if

$$p^{-1} < \max\{\sqrt{2^b/J}, 2^b/(I \cdot J)\}$$

where $I = \Gamma^{\mathsf{out}}$ and $J = \Gamma^{\mathsf{in}}$ if $\Gamma^{\mathsf{out}} \leq \Gamma^{\mathsf{in}}$; $I = \Gamma^{\mathsf{in}}$ and $J = \Gamma^{\mathsf{out}}$ otherwise.

Continuing our example with the 4-round differential path for Keccak-f[1600] from Table 1, we have $\Gamma^{\mathsf{in}} = 2^{1064}$ and $\Gamma^{\mathsf{out}} = 2^{576}$ for a 7-round distinguisher of the internal permutation. Thus the generic complexity to find such constraint pairs is 2^{268} computations which is much higher than 2^{142}. All the distinguishers we obtain with this method are summarized in Table 2.

Note that the reader might be concerned by the fact that the sizes Γ^{in} and Γ^{out} of the reachable differences sets can be very big and might be not easy to describe in a compact way in our distinguisher. However, we emphasize that all the reachable differences on the output (resp. input) are actually built from the independent combinations of all the possible output differences (resp. input differences) of all active Sboxes in the last round (resp. first round). Therefore, the description of this set is easily done by identifying the reachable output differences (resp. input differences) for all the Sboxes independently.

5 The Rebound Attack on Keccak

The rebound attack is a freedom degrees utilization technique that was first proposed by Mendel *et al.* in [18] as an analysis of round-reduced `Grøstl` and `Whirlpool`. It was then improved in [17, 16, 12, 24] to analyze `AES` and `AES`-like permutations and also ARX ciphers [15].

With the help of rebound techniques, we show in this section how to extend the number of attacked rounds significantly, but for a higher complexity. We will see that the application of the rebound attack for Keccak seems quite difficult. Indeed, the situation for Keccak is not as pleasant as the `AES`-like permutations case where the utilization of truncated differential paths (i.e. path for which one only checks if one cell is active or inactive, without caring about the actual difference value) makes the application of rebound attacks very easy to handle.

5.1 The original rebound attack

Let P denote a permutation, which can be divided into 3 sub-permutations, *i.e.,* $P = E_F \circ E_I \circ E_B$. The rebound attack works in two phases.

- **Inbound phase or controlled rounds**: this phase usually starts with several chosen input/output differences of E_I that are propagated through linear layers forward and backward. Then, one can carry out meet-in-the-middle (MITM) match for differences through a single Sbox layer in E_I and generate all possible value pairs validating the matches.
- **Output phase or uncontrolled rounds**: With all solutions provided in the inbound phase, check if any pair validates as well the differential paths for both the backward part p_B and the forward part p_F.

In most cases, the inbound phase can be done fast due to the MITM nature and generates solution pairs with very low average complexity. Hence, attackers usually choose the position of E_I in the differential path so that it covers a low probability portion of the trail in order to increase the success probability of the outbound phase. For example when dealing with the 128-bit `AES` internal permutation, this MITM is performed on sixteen parallel 8-bit Sboxes. If the match is done with k Sboxes being active and since a random 8-bit input/output difference can be matched with probability 1/2 through a single `AES` Sbox, one needs to try at least 2^k input/output difference pairs of E_I in order to hope having one matching candidate for the inbound phase. However, once a match is found one can generate about 2^k solution values from it, thus leading to an average cost of about one operation per solution for the controlled rounds. The final goal of the attacker is then to generate enough inbound phase solution values such that one of them also verifies the forward and backward outbound trails, i.e at least $p_B^{-1} \cdot p_F^{-1}$ pairs need to be tested if p_B and p_F are the respective backward and forward differential probability (DP) of the outbound trail.

The SuperSbox technique [16, 12] extends the E_I from one Sbox layer to two Sbox layers for an AES-like permutation, by considering two consecutive AES-like rounds as one with column-wise SuperSboxes. This technique is possible due to the fact that one can swap few linear operations with the Sbox in AES, so that the two layers of Sboxes in two rounds become close enough to form one SuperSbox layer. However, in the case of KECCAK, it seems very hard to form any partition into independent SuperSboxes. For the same reason, using truncated differential paths seems very difficult for KECCAK, as it has recently been shown in [8].

During the application of rebound attacks, one has to start with several input/output differences of E_I to complete the inbound phase. For AES-like permutations one can start with truncated differences and thus it is much more handy because this view simplifies a lot the analysis. Indeed, a truncated differential for AES-like permutations can be seen as a collection of several bit differentials, all with the same success probability and the same properties in regards to rebound attacks. Thus, whatever the difference masks considered for the input/output of the controlled rounds, the probabilities p_B and p_F will remain the same, so will the probability to get a match in E_I or the number of solutions that can be generated from a match. This will not be the case for KECCAK as we can not use truncated differential paths and the analysis will be much more involved.

5.2 Applying the rebound attack for KECCAK internal permutations

Assume that we know a set of n_B differential trails (called *backward trails*) on nr_B KECCAK rounds and whose DP is higher or equal to p_B. For the moment, we want all these backward paths to share the same input difference mask Δ_B^{in} and we denote by $\Delta_B^{\mathsf{out}}[i]$ the output difference mask of the i-th trail of the set. Similarly, we consider that we also know a set of n_F differential trails (called *forward trails*) on nr_F KECCAK rounds and whose DP is higher or equal to p_F. We want all those forward paths to share the same output difference mask Δ_F^{out} and we denote by $\Delta_F^{\mathsf{in}}[i]$ the input difference mask of the i-th trail of the set.

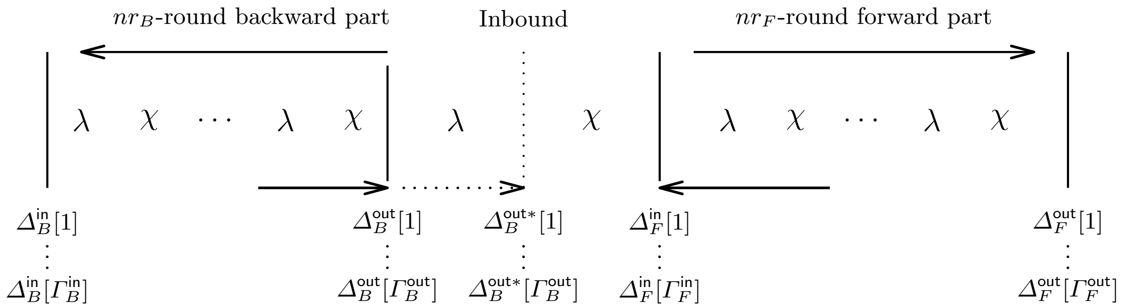

Fig. 1. Rebound attack on KECCAK

Our goal here is to build a differential path on $nr_B + nr_F + 1$ KECCAK rounds (thus one Sbox layer of inbound), by connecting a forward and a backward trail with the rebound technique, and eventually to find the corresponding solution values for the controlled round. We represent by p_{match} the probability that a match is possible from a given element of the backward set and a given element of the forward set, and we denote by N_{match} the number of solution values that can be generated once a match has been obtained.

For this connection to be possible, we need the inbound phase to be a valid differential path, that is we need to find a valid differential path from a $\Delta_B^{\mathsf{out}*}$ to a Δ_F^{in}. By using random $\Delta_B^{\mathsf{out}*}$ and Δ_F^{in} this will happen in general with very small probability, because we need the very same set of Sboxes to be active/inactive in both forward and backward difference masks so we can have a chance to get a match. Even if the set of active Sboxes matches, we still require the differential transitions through all the active Sboxes to be possible.

We can generalize a bit this approach by allowing a fixed set of differences Δ_B^{in} (resp. Δ_F^{out}) instead of just one. We call Γ_B^{in} (resp. Γ_B^{out}) the *size* of the set of possible Δ_B^{in} (resp. Δ_B^{out}) values for the backward paths. Similarly, we call Γ_F^{in} (resp. Γ_F^{out}) the size of the set of possible Δ_F^{in} (resp. Δ_F^{out}) values for the forward paths. In fact, the number of possible differences in the backward or forward parts will form a butterfly shape (see Figure 2). We call Γ_B^{mid} (resp. Γ_F^{mid}) the minimum number of differences in the backward (resp. forward) part.

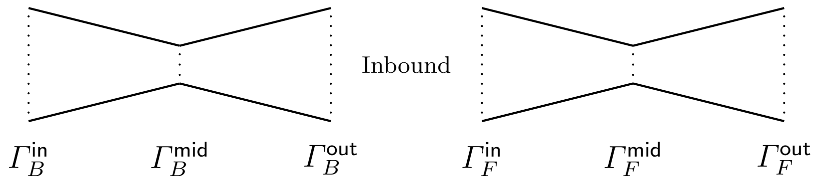

Fig. 2. Number of differences for the rebound attack on KECCAK.

The total complexity C to find one valid internal state pair for the $(nr_B + nr_F + 1)$-round path is

$$C = n_F + n_B + \frac{1}{p_{\mathsf{match}}} \cdot \left\lceil \frac{1}{p_F \cdot p_B \cdot N_{\mathsf{match}}} \right\rceil + \frac{1}{p_B \cdot p_F}\ , \tag{1}$$

with

$$\Gamma_B^{\mathsf{out}} \cdot \Gamma_F^{\mathsf{in}} = \frac{1}{p_{\mathsf{match}}} \cdot \left\lceil \frac{1}{p_F \cdot p_B \cdot N_{\mathsf{match}}} \right\rceil\ . \tag{2}$$

The first two terms are the costs to generate the backward and forward paths. The term $\lceil \frac{1}{p_F \cdot p_B \cdot N_{\mathsf{match}}} \rceil$ denotes the number of time we will need to perform the inbound and each inbound costs $1/p_{\mathsf{match}}$. The last term is the cost for actually performing the outbound phase. The condition (2) is needed since we need enough differences to perform the inbound phase.

Roadmap. For a better understanding of the behavior of the Sboxes in the rebound attack, we will introduce some useful lemmas in Section 5.3. We explain how to prepare the forward and backward differential paths in Section 5.4 and describe the inbound and outbound phases in Section 5.5 and 5.6 respectively. We explain how to relate Sections 5.4, 5.5 and 5.6 in Section 5.7, we show also how we can reduce the complexity of the attack and we give a numerical application of our model. Finally we construct distinguishers from the differential paths in Section 5.8.

5.3 An ordered Buckets and Balls Problem

We model the active/inactive Sboxes match as a **limited capacity ordered buckets and balls problem**: the $s = 5w$ ordered buckets ($s = 320$ for KECCAK-$f[1600]$) limited to capacity 5 will represent the s 5-bit Sboxes and the x_B (resp. x_F) balls will stand for the Hamming weight of the difference in $\Delta_B^{\mathsf{out}*}$ (resp. in Δ_F^{in}). Given a set B of s buckets in which we randomly throw x_B balls and a set F of s buckets in which we randomly throw x_F balls, we call the result a **pattern-match** when the set of empty buckets in B and F after the experiment are the same.[4] Before computing the probability of having a pattern-match, we need the following lemma.

Lemma 1. *The number of possible combinations $b_{\mathsf{bucket}}(n, s)$ to place n balls into s buckets of capacity* 5 *such that no bucket is empty is*

$$b_{\mathsf{bucket}}(n, s) := \begin{cases} \displaystyle\sum_{i=\lceil n/5 \rceil}^{s} (-1)^{s-i} \binom{s}{i} \binom{5i}{n} & \textit{if } s \le n \le 5s \\ 0 & \textit{else.} \end{cases} \tag{3}$$

[4] Note that the *position of the balls in the buckets* is significant. This is why we refer to an ordered buckets and balls problem.

Proof. First note that the number of combinations verifies the following recurrence relation:

$$b_{\textsf{bucket}}(n,s) = \binom{5}{1} b_{\textsf{bucket}}(n-1,s-1) + \binom{5}{2} b_{\textsf{bucket}}(n-2,s-1) + \cdots + \binom{5}{5} b_{\textsf{bucket}}(n-5,s-1) ,$$

with $b_{\textsf{bucket}}(x,s) = 0$ when $x \leq 0$ and $b_{\textsf{bucket}}(x,1) = \binom{5}{x}$ when $0 < x \leq 5$ and 0 else.

Let's consider the following generating function:

$$G_s(x) \coloneqq \sum_{k\geq 0} b_{\textsf{bucket}}(k,s) x^k .$$

We have

$$G_1(x) = \binom{5}{1} x + \binom{5}{2} x^2 + \cdots + \binom{5}{5} x^5 = (x+1)^5 - 1 .$$

Hence, $\sum_{k\geq 1} b_{\textsf{bucket}}(k,s)x^k =$

$$\begin{aligned}
&\sum_{k\geq 1} \left[\binom{5}{1} b_{\textsf{bucket}}(k-1,s-1) + \binom{5}{2} b_{\textsf{bucket}}(k-2,s-1) + \cdots + \binom{5}{5} b_{\textsf{bucket}}(k-5,s-1) \right] x^k \\
&= \sum_{k\geq 1} \binom{5}{1} b_{\textsf{bucket}}(k-1,s-1)x^k + \sum_{k\geq 2} \binom{5}{2} b_{\textsf{bucket}}(k-2,s-1)x^k + \cdots + \sum_{k\geq 5} \binom{5}{5} b_{\textsf{bucket}}(k-5,s-1)x^k \\
&= \sum_{k\geq 0} \binom{5}{1} b_{\textsf{bucket}}(k,s-1)x^k \cdot x + \sum_{k\geq 0} \binom{5}{2} b_{\textsf{bucket}}(k,s-1)x^k \cdot x^2 + \cdots + \sum_{k\geq 0} \binom{5}{5} b_{\textsf{bucket}}(k,s-1)x^k \cdot x^5 \\
&= G_{s-1}(x) \cdot \left(\binom{5}{1} x + \binom{5}{2} x^2 + \cdots + \binom{5}{5} x^5 \right) = G_{s-1}(x) \cdot \left((x+1)^5 - 1 \right) = G_s(x) .
\end{aligned}$$

The last equality follows from $b_{\textsf{bucket}}(0,s) = 0$. Let $A \coloneqq (x+1)^5 - 1$. We have

$$G_s(x) = G_{s-1}(x)A = G_1(x)A^{s-1} = A^s = \sum_{i=0}^{s} (-1)^{s-i} \binom{s}{i} \left((x+1)^5\right)^i = \sum_{i=0}^{s} \sum_{j=0}^{5i} (-1)^{s-i} \binom{s}{i} \binom{5i}{j} x^j . \quad (4)$$

The number of combinations $b_{\textsf{bucket}}(n,s)$ we are looking for is the coefficient of x^n in the expression $G_s(x) = \sum_{k\geq 0} b_{\textsf{bucket}}(k,s)x^k$. By summing in (4) the terms contributing to x^n we obtain the wanted result. $\square$

Using (3), we can derive the probability $p_{\textsf{bucket}}$ that every bucket contains at least one ball when n balls are thrown into s buckets with capacity 5:

$$p_{\textsf{bucket}}(n,s) \coloneqq \frac{b_{\textsf{bucket}}(n,s)}{\binom{5s}{n}} . \quad (5)$$

The expected number of active buckets when n balls are thrown into s buckets is given by

$$\frac{\sum_{i=0}^{s} b_{\textsf{bucket}}(n,s-i) \cdot \binom{s}{i} \cdot (s-i)}{\binom{5s}{n}} . \quad (6)$$

We can now relate this lemma to the more general pattern-match problem.

Lemma 2. *Given a set of s buckets B of capacity 5 in which we throw x_B balls and a set of s buckets F in which we throw x_F balls, the probability that B and F have the same pattern of empty buckets is given by*

$$p_{pattern}(s,x_B,x_F) = \frac{1}{\binom{5s}{x_B}\binom{5s}{x_F}} \sum_{i=0}^{s} b_{\textsf{bucket}}(x_B,s-i) b_{\textsf{bucket}}(x_F,s-i) \binom{s}{i} ,$$

where $b_{\textsf{bucket}}(x,s)$ is defined as in (3). The average number $n_{pattern}$ of non-empty buckets if both experiments results follow the same pattern is given by

$$n_{pattern}(s,x_B,x_F) = \frac{\sum_{i=0}^{s} b_{\textsf{bucket}}(x_B,s-i) b_{\textsf{bucket}}(x_F,s-i) \binom{s}{i} (s-i)}{\sum_{i=0}^{s} b_{\textsf{bucket}}(x_B,s-i) b_{\textsf{bucket}}(x_F,s-i) \binom{s}{i}} .$$

This model tells us that when the number of balls (*i.e.,* active bits) is not too small on both sides, most of the matches happen when (almost) all the Sboxes are active. Figure 8 in Appendix A depicts this behavior.

A More General Problem. We can also look into a more general problem, i.e., we characterize more precisely how the bits are distributed into the Sboxes.

Lemma 3. *The probability p_{dist} of distributing randomly n active bits into s 5-bit Sboxes such that exactly A_i Sboxes contain i bits, for $i \in [1,5]$ is*

$$p_{\mathsf{dist}}(A_1, A_2, A_3, A_4, A_5) := \frac{s!\binom{5}{1}^{A_1}\binom{5}{2}^{A_2}\binom{5}{3}^{A_3}\binom{5}{4}^{A_4}\binom{5}{5}^{A_5}}{(s - A_1 - A_2 - A_3 - A_4 - A_5)!A_1!A_2!A_3!A_4!A_5!\binom{5s}{n}}, \tag{7}$$

with $n = A_1 + 2A_2 + 3A_3 + 4A_4 + 5A_5$.

Important Remark. Since most matches happen when all the Sboxes are active, in order to simplify the analysis, we will consider from now on that we will only use forward and backward paths such that *all Sboxes are active for the χ layer of the inbound phase.*

5.4 The differential paths sets

In this section, we explain how we generate the forward and backward paths because this will have an impact on the derivation of p_{match} and N_{match} (this will be handled in the next two sections).

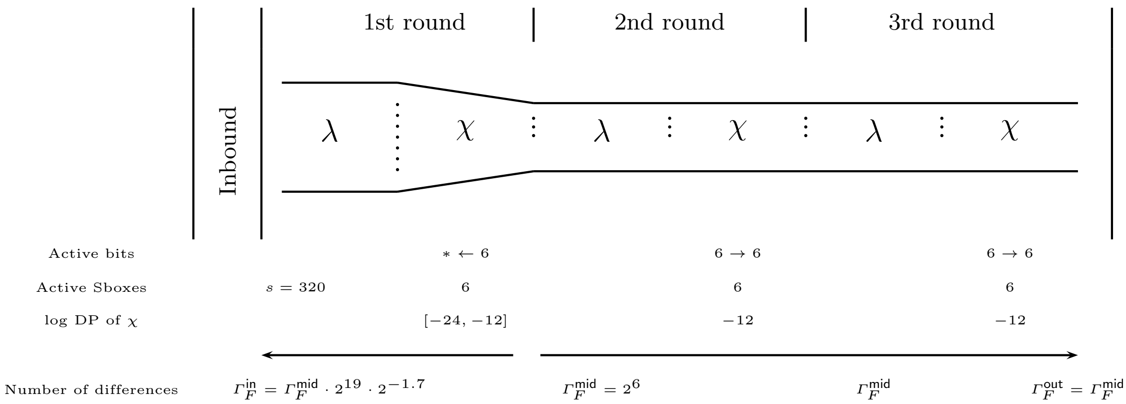

Fig. 3. The forward trails. Values are taken from the 3 round differential path from Table 7. The distance between the two lines reflects the number of differences.

The forward paths. For the forward paths set, we start by choosing a differential trail computed from the previous section and we derive a set from it by exhausting all the possible Sbox differential transitions for the inverse of the χ layer in its first round (all the paths will be the same except the differences on their input and on the input of the χ layer in the first round). For example, we can use the 3 first rounds of the 4-round differential path from Table 7 in the Appendix which have a total success probability 2^{-36} and present 6 active Sboxes during the χ layer of the first round. Note that we didn't choose the best 3-round differential path (with success probability 2^{-32}) since it will not provide enough paths (due to its input difference Hamming weight being too small). We randomize the χ^{-1} layer differential transitions for the 6 active Sboxes of the first round, and we obtain about 2^{19} distinct trails in total. We analyzed that all the trails of this set have a success probability of at least 2^{-48} (this is easily obtained with the χ^{-1} DDT). Moreover, note that they will all have the same output difference mask (at the third round), but distinct input masks (at the first round). Since we previously forced the requirement that all Sboxes must be active for the inbound match, we check experimentally that $2^{17.3}$ of the 2^{19} members of the set fulfill this condition.[5] We call τ_F the ratio of paths that verify this condition over the total number of paths, i.e., $\tau_F = 2^{-1.7}$. Overall, we built a set of $2^{17.3}$ forward differential paths on $nr_F = 3$ KECCAK-$f[1600]$ rounds, all with DP higher or equal to $p_F = 2^{-48}$. We can actually generate 64 times more paths by observing that they are equivalent by translation along the KECCAK lane (the z axis). However, these paths will have distinct output difference masks (the same difference mask rotated along the z axis), and we have $\Gamma_F^{\mathsf{out}} = \Gamma_F^{\mathsf{mid}} = 2^6$. The *total amount of input differences* Γ_F^{in} is, hence, $\Gamma_F^{\mathsf{in}} := \Gamma_F^{\mathsf{mid}} \cdot 2^{17.3} = 2^{23.3}$ and we have to generate in total $n_F = \tau_F \cdot \Gamma_F^{\mathsf{in}} = 2^{25}$ forward differential paths.

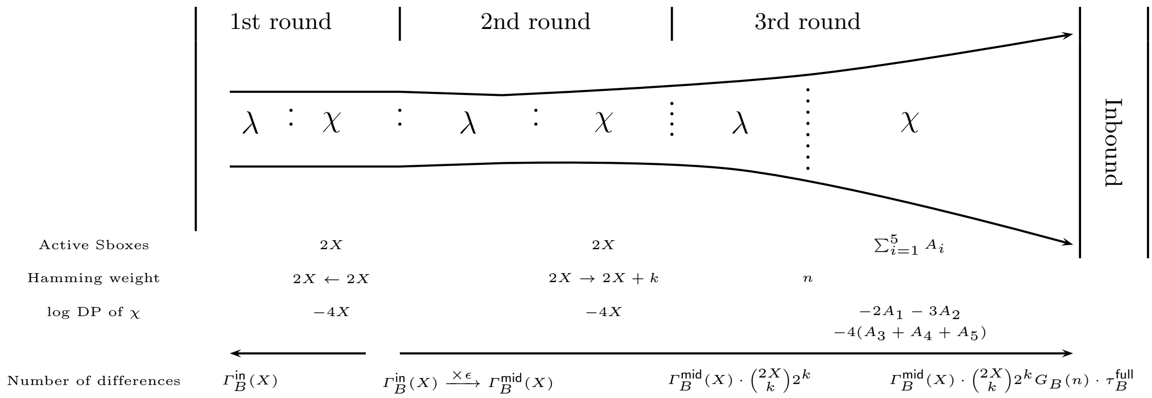

Fig. 4. The backward trails. The distance between the two arrows reflects the number of differences.

The backward paths. Applying the same technique to the backward case does not generate a sufficient amount of output differences Γ_B^{out}, crucial for a rebound-like attack. Thus, concerning the backward paths set, we build another type of 3-round trails. We need first to ensure that we have *enough differential paths* to be able to find a match in the inbound phase, i.e., we want $\Gamma_B^{\mathsf{out}} \cdot \Gamma_F^{\mathsf{in}} = 1/p_{\mathsf{match}} \cdot \lceil \frac{1}{p_F \cdot p_B \cdot N_{\mathsf{match}}} \rceil$ following (2). Moreover, we will require these path to verify two conditions:

1. First, we need to filter paths that have not all Sboxes active in the χ layer of the inbound phase. Using (5), this happens with a probability about $\tau_B^{\mathsf{full}} := p_{\mathsf{bucket}}(800, 320) = 2^{-15.9}$ if we assume that about half of the bits are active. This assumption will be verified in our case (and was verified in practice) since our control on the diffusion of the active bits will be reduced greatly.
2. Moreover, *all* the paths we collect should have a *DP of at least p_B such that the number of solutions N_{match} generated in the inbound phase is sufficient.* Indeed, we must have $N_{\mathsf{match}} \geq 1/(p_F \cdot p_B)$ in order to have a good success probability to find one solution for the entire path. We call τ_B^{DP} the probability that a path verifies this property. Hence, we need $p_B \geq p_B^{\min} = 1/(p_F \cdot N_{\mathsf{match}})$. We will show in Section 5.7 that $N_{\mathsf{match}} = 2^{509}$ and we previously showed that $p_F = 2^{-48}$. Hence, $p_B^{\min} = 2^{48-509} = 2^{-461}$.

These two filters induce a ratio $\tau_B := \tau_B^{\mathsf{full}} \cdot \tau_B^{\mathsf{DP}}$ of "good" paths. We have $n_B \cdot \tau_B = \Gamma_B^{\mathsf{out}}$, where n_B is the number of paths we need to generate. Thus, we need to generate $n_B^{\min} := 1/(p_{\mathsf{match}} \cdot \lceil \frac{1}{p_F \cdot p_B \cdot N_{\mathsf{match}}} \rceil \cdot \Gamma_F^{\mathsf{in}} \cdot \tau_B)$ trails to perform the rebound. We will show in Section 5.7 that $p_{\mathsf{match}} = 2^{-498.11}$, that $\lceil \frac{1}{p_F \cdot p_B \cdot N_{\mathsf{match}}} \rceil = 1$ and that $\tau_B = 2^{-17.37}$. We also know that $\Gamma_F^{\mathsf{in}} = 2^{23.3}$. Hence, $n_B^{\min} = 2^{498.11+17.37-23.3} = 2^{492.18}$.

We show now how we generated these paths. Fig. 4 can help the reading. We start at the beginning of the second round by forcing X columns of the internal state to be active and each active column will contain only 2 active bits (thus a total of $2X$ active bits). Therefore, we can generate $\binom{5}{2}^X \cdot \binom{s}{X}$ distinct starting differences and each of them will lead to a distinct input difference of the backward path. This implies that

[5] The small amount of filtered forward paths (a factor $2^{1.7}$) is due to the regularity of the output of θ inverse. Thus, most of the paths have all Sboxes active when the Hamming weight of the input is low.

$\Gamma_B^{\mathsf{in}} = \binom{5}{2}^X \cdot \binom{s}{X}$. Note also that all active columns are in the CP-Kernel and thus applying the θ function on this internal state will leave all bit-differences at the same place. Then, applying the ρ and π layers will move the $2X$ active bits to random locations before the Sbox layer of the second round. If X is not too large, we can assume that for a good fraction of the paths, all active bits are mapped to distinct Sboxes and thus we obtain $2X$ active Sboxes, each with one active bit on its input. We call ϵ this fraction of paths which is given by

$$\epsilon \coloneqq p_{\mathsf{dist}}(2X, 0, 0, 0, 0) , \tag{8}$$

where p_{dist} is given by Lemma 3.[6] We will need to take ϵ into account when we count the total number of paths we can generate. This position in the paths, i.e., after the linear layer of the second round, is the part with the lowest amount of distinct differences. Hence, we call the number of differences at this point $\Gamma_B^{\mathsf{mid}}(X) = \Gamma_B^{\mathsf{in}}(X) \times \epsilon$.

Looking at the DDT from χ (Table 3 in Appendix B), one can check that with one active input bit in an Sbox, there always exists:

- two distinct transitions with probability 2^{-2} for the KECCAK Sbox such that we observe 2 active bits on its output (we call it a $1 \mapsto 2$ transition)
- one single transition with probability 2^{-2} and one single active bit on its output (a $1 \mapsto 1$ transition). This transition is in fact the identity.

We need to define how many $1 \mapsto 1$ and $1 \mapsto 2$ transitions we have to use, since there is a tradeoff between the number of paths obtained and the DP of these paths. Whatever choices we make, we always have that the success probability of this χ transition (in the second round) is 2^{-4X}. Let k be the number of $1 \mapsto 2$ transitions among the $2X$ possible ones. We will observe $2X + k$ active bits after χ. Before the χ transition, we have $\Gamma_B^{\mathsf{mid}}(X)$ different paths from the initial choice. For each of these paths, we can now select $\binom{2X}{k}$ distinct sets of $1 \mapsto 2$ transitions each of which can generate 2^k different paths. These $2X + k$ bits are expanded through θ to *at most* $11 \cdot (2X + k) = 22X + 11k$ bits. However, this expansion factor (every active bit produces 11 one) is smaller when the number of bits increases. Let n be the number of obtained active bits at the input of the Sboxes in the third round. At the beginning of the third round, we have $2X + k$ active bits. For KECCAK-f[1600], given $2X + k$ active bits at the input of θ, we get

$$n \approx u - \frac{u \cdot (u-1)}{1600} \tag{9}$$

bits at the output, with $u \coloneqq 11(2X + k)$ for X small enough. Indeed, the $2X + k$ bits are first multiplied by 11 due to the property of θ. We suppose now that these u active bits are thrown randomly and we check for collisions. Given u bits, we can form $u \cdot (u - 1)/2$ different pairs of bits. The probability that a pair collides is 2^{-1600}, hence, we have about $u \cdot (u-1)/(2 \cdot 1600)$ collisions of two bits. In a 2-collision, two active bits are wasted (they become inactive). Hence, we can remove $u \cdot (u - 1)/1600$ from the overall number of active bits. For small X, we can neglect collisions between three, four and five active bits, since the bits before θ are most likely separated and will not collide. Hence we verify (9). This model has been verified in simulations for the values we are using.

We need now to evaluate the number of active Sboxes in the χ layer of the third round. However, in order to precisely evaluate the DP of this layer (that we want to be higher than $p_B^{\min}$) and the expansion factor we get on the amount of distinct differential paths, we also need to look at how the bits are distributed into the input of the Sboxes. The probability p_{dist} of distributing randomly n active bits into s 5-bit Sboxes such that exactly A_i Sboxes contain i bits, for $i \in [1, 5]$ is given by Lemma 3.

Lemma 4. *Suppose that we have n active bits before χ in the third round. Then, if $n \leq s$, the expected number of* useful *(i.e., which have $\mathsf{DP} \geq p_B^{\min}$) paths $G_B(n)$ we can generate verifies*

$$G_B(n) \geq \sum_{A_5=0}^{\lfloor n/5 \rfloor} \sum_{A_4=0}^{\lfloor (n-5A_5)/4 \rfloor} \sum_{A_3=0}^{\lfloor (n-5A_5-4A_4)/3 \rfloor} \sum_{A_2=0}^{\lfloor (n-5A_5-4A_4-3A_3)/2 \rfloor} F(X, A_1, A_2, A_3, A_4, A_5) \cdot 2^{2A_1+3A_2+3.58A_3+4(A_4+A_5)} , \tag{10}$$

[6] Simulations verified this behavior in practice for the parameters we use in our attack.

where $A_1 := n - 5A_5 - 4A_4 - 3A_3 - 2A_2$ *and*

$$F(X, A_1, A_2, A_3, A_4, A_5) := \begin{cases} p_{\mathsf{dist}}(A_1, A_2, A_3, A_4, A_5) & \textit{if } 2^{-8X-2A_1-3A_2-4(A_3+A_4+A_5)} \geq p_B^{\min} \\ 0 & \textit{else.} \end{cases} \tag{11}$$

Note that we use $F(\dots)$ *to filter the paths that have a too low DP.*

Proof. Given the number of active input bits in every Sbox, it is easy to compute the number of paths we can generate by looking into the DDT.[7] We find that for an input Hamming weight of 1 (resp. 2), there are always 2^2 (resp. 2^3) possible output differences. For an Hamming weight of 3, half of the input differences can produce 2^3 differences and half 2^4 differences. Hence, the expected value is $2^{3.58}$. For input Hamming weights of 4 and 5, we can always produce 2^4 differences. Thus, the total expected number of paths we can generate when we have A_i Sboxes with an input Hamming weight of i is $2^{2A_1+3A_2+3.58A_3+4(A_4+A_5)}$.

Moreover, we count only the paths that verify $p_B \geq p_B^{\min}$ by discarding all the paths that have a DP smaller than $p_B^{\min}$ using the filter $F(\dots)$. The DP of the complete path is given by

$$2^{-4X-4X-2A_1-3A_2-4(A_3+A_4+A_5)} . \tag{12}$$

Indeed, in the first round, since $2X$ Sboxes are active with one active bit in each active Sbox, we can choose a transition that has a probability 2^{-2} per active Sbox (see the DDT of χ^{-1}). Hence, the DP for the first round is 2^{-4X}. For the second round, since we still have one active bit per Sbox, we have a DP of 2^{-4X} as well. For the third round, an analysis of the DDT shows that, when we have 1 (resp. 2) active bit in the input, the DP of the SBox is always 2^{-2} (resp. 2^{-3}). For a Hamming weight of 3, there are two different DPs depending on the input. We considered the worst case, which is 2^{-4}. For a Hamming weight of 4 and 5, the DP is always 2^{-4}. Hence, the DP of the complete path verifies (12).

Now, from Lemma 3, we deduce that the paths occur with probability $p_{\mathsf{dist}}(A_1, A_2, A_3, A_4, A_5)$. Hence, the expected number of paths we will get is the sum of all the probabilities of the path that are not discarded by the filter. $\square$

In practice, we compute $G_B(n)$ by summing over all possible values of $A_1, \dots, A_5$, such that $n = A_1 + 2A_2 + 3A_3 + 4A_4 + 5A_5$.

We have now reached the inbound round and we discard all the paths that do not have all Sboxes active. Hence, we keep only a fraction of $\tau_B^{\mathsf{full}} = 2^{-15.9}$ paths.

It is now easy to see that

$$\tau_B^{\mathsf{DP}} := \sum_{A_5=0}^{\lfloor n/5 \rfloor} \sum_{A_4=0}^{\lfloor (n-5A_5)/4 \rfloor} \sum_{A_3=0}^{\lfloor (n-5A_5-4A_4)/3 \rfloor} \sum_{A_2=0}^{\lfloor (n-5A_5-4A_4-3A_3)/2 \rfloor} F(X, A_1, A_2, A_3, A_4, A_5) \tag{13}$$

whith $F(\dots)$ defined in (11) since this is exactly the fraction of path we keep.

To summarize, we have now reached the inbound round and we are able to generate

$$\Gamma_B^{\mathsf{out}} = \epsilon \cdot \binom{5}{2}^X \cdot \binom{s}{X} \cdot \binom{2X}{k} \cdot 2^k \cdot G_B(n) \cdot \tau_B^{\mathsf{full}} \tag{14}$$

differences that have a good DP and all Sboxes active and the total number of paths we have to generate is $n_B = \Gamma_B^{\mathsf{out}} / \tau_B = \Gamma_B^{\mathsf{out}} / (\tau_B^{\mathsf{full}} \cdot \tau_B^{\mathsf{DP}})$.

By playing with the filter bound, we noticed the following behavior. The stronger the filter is (i.e., the higher we set the bound on the DP), the higher the expected value of the Hamming weight at the input of the Sboxes of the inbound phase will be. This behavior will allow us to reduce the complexity of our attack in Section 5.7, where we discuss the numerical application. Hence, instead of filtering at $p_B^{\min}$, we will filter at a higher value to get better results.

[7] We considered the average case here since we already have a lot of paths to start with at the input of the third round.

Summary. At this point, we started with n_F (resp. n_B) forward (resp. backward) paths from which we kept only Γ_F^{in} (resp. Γ_B^{out}) candidates that have a DP greater than p_F (resp. p_B) and all Sboxes actives during the inbound.

5.5 The inbound phase

Now that we have our forward and backward sets of differential paths, we need to estimate the average probability p_{match} that two trails can match during the inbound phase of the rebound attack. We recall that we already enforced all Sboxes to be active during this match, so p_{match} only takes into account the probability that the differential transitions through all the s Sboxes of the internal state are possible.

A trivial method to estimate p_{match} would be to simply consider an average case on the Keccak Sbox. More precisely, the average probability that a differential transition is possible through the Keccak Sbox, given two random non-zero 5-bit input/output differences is equal to $2^{-1.605}$. Thus, one is tempted to derive $p_{\mathsf{match}} = 2^{-1.605 \cdot s}$. However, we observed experimentally that the event of a match greatly depends on the **Hamming weight of the input of the Sboxes** and this can be easily observed from the DDT of the χ layer (for example with an input Hamming weight of one the match probability is $2^{-2.95}$, while for an input Hamming weight of four the match probability is $2^{-0.95}$). Note that *this effect is only strong regarding the input of the Sbox* (i.e. the backward paths), but there is no strong bias on the differential matching probability concerning the output Hamming weight.

Therefore, in order to model more accurately the input Hamming weight effect on the matching event, we first divide the backward paths *depending on their Hamming weight* and treat each class separately. More precisely, we look at each possible input Hamming weight division among the s Sboxes. To represent this division, we only need to look at the number of Sboxes having a specific input Hamming weight (their relative position do not matter). We denote by c_i the number of Sboxes having an input Hamming weight i and we need the following equations to hold

$$\sum_{i=1}^{5} c_i = s \tag{15}$$

since we forced that all Sboxes are active during a match. Moreover, for a Hamming weight value w, we have

$$\sum_{i=1}^{5} i \cdot c_i = w \ . \tag{16}$$

The set of divisions c_i verifying (15) and (16) is denoted by C_w. The number of possible $5s$-bit vectors satisfying $(c_1, \dots, c_5)$ (i.e., c_1 Sboxes with 1 active bit, c_2 with 2 etc.) is denoted $b_c(c_1, \dots, c_5)$ and

$$\begin{aligned} b_c(c_1,\dots,c_5) &= \binom{s}{c_1} \cdot \binom{s-c_1}{c_2} \cdot \dots \cdot \binom{s-c_1-c_2-c_3-c_4}{c_5} \cdot \binom{5}{1}^{c_1} \cdot \dots \cdot \binom{5}{5}^{c_5} \\ &= \frac{s!}{c_1! c_2! \dots c_5!} \cdot 5^{c_1+c_4} \cdot 10^{c_2+c_3} \ . \end{aligned} \tag{17}$$

We can now compute the probability of having a match p_{match} depending on the input Hamming weight divisions:

Theorem 1. *The probability p_{match} of having a match is*

$$p_{\mathsf{match}} = \sum_{w=s}^{5s} \Pr[Hw_{\mathsf{total}} = w | \mathsf{full}] \cdot \sum_{(c_1,\dots,c_5)\in C_w} \frac{b_c(c_1,\dots,c_5)}{b_{\mathsf{bucket}}(w,s)} \prod_{i=1}^{5} \left(\sum_{y\in\{0,1\}^5} \sum_{\substack{v\in\{0,1\}^5: \\ \mathsf{Hw}(v)=i}} \frac{P_{\mathsf{out}}(y) \cdot \mathbb{1}_{\mathsf{DDT}[v][y]}}{\binom{5}{i}} \right)^{c_i} , \tag{18}$$

where $P_{\mathsf{out}}(y)$ is the measured probability distribution of having y at the output of an Sbox when we enforce all Sboxes to be active, $\Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}]$ *is the measured probability distribution of the Hamming weight of the input of the Sboxes when all Sboxes are active, $b_c(\dots)$ is given by* (17)*, $b_{\mathsf{bucket}}(w,s)$ by Lemma 1 and $\mathbb{1}_{\mathsf{DDT}[v][y]}$ is set to one if the entry $[v][y]$ is non-zero in the DDT of the χ layer and zero otherwise.*[8]

[8] Note that $\Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}]$ greatly depends on the backward paths we choose and that these paths depends on p_{match}. We explain how to solve this cyclic dependency in Section 5.7.

Proof. Let full be the event denoting that all Sboxes are active at the inbound phase. We have

$$p_{\mathsf{match}} \coloneqq \Pr[\mathsf{match}|\mathsf{full}] = \sum_{w} \Pr[\mathsf{match}|\mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] \cdot \Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}] \;.$$

We define $p_{\mathsf{match}}(w) \coloneqq \Pr[\mathsf{match}|\mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}]$. We have

$$p_{\mathsf{match}}(w) = \sum_{(c_1,\ldots,c_5)\in C_w} \Pr[\mathsf{match}|(c_1,\ldots,c_5), \mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] \cdot \Pr[(c_1,\ldots,c_5)|\mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] \;. \tag{19}$$

We easily find that

$$\Pr[(c_1,\ldots,c_5)|\mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] = \frac{b_c(c_1,\ldots,c_5)}{b_{\mathsf{bucket}}(w,s)} \;, \tag{20}$$

since $b_c(c_1,\ldots,c_5)$ is the number of possible combinations of vectors verifying $c_1,\ldots,c_5$ and $b_{\mathsf{bucket}}(w,s)$ the number of possible combinations of vectors for which all Sbox are active. It remains to compute $\Pr[\mathsf{match}|(c_1,\ldots,c_5), \mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] = \Pr[\mathsf{match}|(c_1,\ldots,c_5), \mathsf{full}]$, since $(c_1,\ldots,c_5)$ have all a total Hamming weight of w. We can now consider every Sbox independently. Hence,

$$\Pr[\mathsf{match}|(c_1,\ldots,c_5), \mathsf{full}] = \prod_{i=1}^{5} \left(\Pr[\mathsf{match}|\mathsf{Hw}_{\mathsf{SBox}} = i, \mathsf{full}]\right)^{c_i} \tag{21}$$

and

$$\Pr[\mathsf{match}|\mathsf{Hw}_{\mathsf{SBox}} = i, \mathsf{full}] = \sum_{y\in\{0,1\}^5} \sum_{\substack{v\in\{0,1\}^5:\\ \mathsf{Hw}(v)=i}} \frac{P_{\mathsf{out}}(y)\cdot \mathbb{1}_{\mathsf{DDT}[v][y]}}{\binom{5}{i}} \;.$$

$\square$

We continue now with our example of the Keccak-f[1600] internal permutation. The measured distributions along with some intermediate values are given in Appendix C. Applying Theorem 1, we find that $p_{\mathsf{match}} = 2^{-490.15}$. Said in other words, we require to test $1/p_{\mathsf{match}}$ backward/forward paths combinations in order to have a good chance for a match. Note that in the next section, we will actually put an extra condition on the match in order to be able to generate enough values in the worst case during the outbound phase.

5.6 The outbound phase

Now that we managed to obtain a match with complexity $1/p_{\mathsf{match}}$, we need to estimate how many solutions can be generated from this match. Again, one is tempted to consider an average case on the Keccak Sbox: the average number of Sbox values verifying a non-zero random input/output difference such that the transition is possible is equal to $2^{1.65}$. The overall number of solutions would then be $2^{1.65\cdot s}$. However, as for p_{match}, this number highly depends on the Hamming weight of the input of the Sboxes and this can be easily observed from the DDT of the χ layer (for example with an input Hamming weight of one the average number of solutions is 2^3, while for an input Hamming weight of four the average number of solutions is 2^1).

In order to obtain the expected number of values N_{match} we can get from a match, we proceed like in the previous section and divide according to the input Hamming weight.

Theorem 2. *Let N be a random variable denoting the number of values we can generate. Let also* full *be the event denoting that all the Sboxes are active for the inbound phase. Given a Hamming weight of w at the input of the Sboxes, we can get $N_w \coloneqq \mathbb{E}[N|\mathsf{match}, \mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}]$ values from a match, with*

$$N_w = \frac{1}{p_{\mathsf{match}}(w)} \sum_{(c_1,\ldots,c_5)\in C_w} \prod_{i=1}^{5} Z^{c_i} \cdot \frac{b_c(c_1,\ldots,c_5)}{b_{\mathsf{bucket}}(w,s)} \;, \tag{22}$$

with

$$Z := \frac{1}{\binom{5}{i}^2} \left(\sum_{\substack{v\in\{0,1\}^5:\\ \mathsf{Hw}(v)=i}} \mathsf{DDT}[v] \right) \sum_{y\in\{0,1\}^5} \sum_{\substack{v\in\{0,1\}^5:\\ \mathsf{Hw}(v)=i}} P_{\mathsf{out}}(y) \cdot \mathbb{1}_{\mathsf{DDT}[v][y]} ,$$

where $\mathsf{DDT}[v]$ *is the value of the non-zero entries in line* v *of the DDT,* $P_{\mathsf{out}}(y)$ *is the measured probability distribution of having* y *at the output of an Sbox when we enforce all Sboxes to be active,* $p_{\mathsf{match}}(w)$ *is given by* (19)*,* $b_c(\dots)$ *is given by* (17)*,* $b_{\mathsf{bucket}}(w, s)$ *is given by Lemma 1 and* $\mathbb{1}_{\mathsf{DDT}[v][y]}$ *is set to one if the entry* $[v][y]$ *is non-zero in the DDT of the* χ *layer and zero otherwise.*

Proof. We have

$$\begin{aligned} N_w &= \sum_{(c_1,\dots,c_5)\in C_w} \mathbb{E}[N|\mathsf{match}, (c_1,\dots,c_5), \mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] \cdot \Pr[(c_1,\dots,c_5)|\mathsf{match}, \mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] \\ &= \sum_{(c_1,\dots,c_5)\in C_w} N_{\mathsf{match}}(c_1,\dots,c_5) \cdot \frac{\Pr[\mathsf{match}|(c_1,\dots,c_5), \mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] \cdot \Pr[(c_1,\dots,c_5)|\mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}]}{\Pr[\mathsf{match}|\mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}]} \\ &= \sum_{(c_1,\dots,c_5)\in C_w} N_{\mathsf{match}}(c_1,\dots,c_5) \cdot \frac{\Pr[\mathsf{match}|(c_1,\dots,c_5), \mathsf{full}] \cdot \Pr[(c_1,\dots,c_5)|\mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}]}{p_{\mathsf{match}}(w)} , \end{aligned}$$

where $N_{\mathsf{match}}(c_1,\dots,c_5) := \mathbb{E}[N|\mathsf{match}, (c_1,\dots,c_5), \mathsf{full}]$. Note that the remaining terms can be computed from (20) and (21). Like before, we can now consider each Sbox independently. Thus

$$N_{\mathsf{match}}(c_1,\dots,c_5) = \prod_{i=1}^{5} \left(\mathbb{E}[N_{\mathsf{SBox}}|\mathsf{match}, \mathsf{Hw}_{\mathsf{SBox}} = i, \mathsf{full}]\right)^{c_i} ,$$

where N_{SBox} is a random variable denoting the number of values we can obtain for a single Sbox. Note that no output distribution needs to be considered, since for a fixed input the non-zero values of the DDT are always the same. We call this non-zero value $\mathsf{DDT}[v]$. Then,

$$\mathbb{E}[N_{\mathsf{SBox}}|\mathsf{match}, \mathsf{Hw}_{\mathsf{SBox}} = i, \mathsf{full}] = \frac{1}{\binom{5}{i}} \sum_{\substack{v\in\{0,1\}^5:\\ \mathsf{Hw}(v)=i}} DDT[v] .$$

□

One would be tempted to take the expected value of all the N_w and compute N_{match} as

$$\sum_{w} \mathbb{E}[N|\mathsf{match}, \mathsf{Hw}_{\mathsf{total}} = w, \mathsf{full}] \cdot \Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{match}, \mathsf{full}] .$$

This expectancy would be fine if we were expecting a high number of matches. This is however not necessarily our case. Hence, we need to ensure that the number of values we can generate from the inbound is sufficient. To do this, first note that N_w decreases exponentially while w increases. Similarly, $p_{\mathsf{match}}(w)$ increases exponentially while w increases. This is depicted in Figure 10 in Appendix C. Thus, we are more likely to obtain a match at a high Hamming weight which will lead to an insufficient N_{match}.

To solve this issue, we proceed as follows. First, we compute N_w for every w. We check then the maximum Hamming weight $w_{\max}$ we can afford such that $N_{w_{\max}} \geq 1/(p_B \cdot p_F)$. This way, we are ensured to obtain enough solutions from the match. However, we need to update our definition of a match: a match occurs only when the Hamming weight of the input is lower than $w_{\max}$. Hence, instead of summing over all possible values of w, we sum only up to $w_{\max}$ and (18) becomes

$$p_{\mathsf{match}} = \sum_{w=s}^{w_{\max}} \Pr[Hw_{\mathsf{total}} = w|\mathsf{full}] \cdot \sum_{(c_1,\dots,c_5)\in C_w} \frac{b_c(c_1,\dots,c_5)}{b_{\mathsf{bucket}}(w,s)} \prod_{i=1}^{5} \left(\sum_{y\in\{0,1\}^5} \sum_{\substack{v\in\{0,1\}^5:\\ \mathsf{Hw}(v)=i}} \frac{P_{\mathsf{out}}(y) \cdot \mathbb{1}_{\mathsf{DDT}[v][y]}}{\binom{5}{i}} \right)^{c_i} . \tag{23}$$

The number of values we can then obtain from the inbound is $N_{\mathsf{match}} \geq N_{w_{\max}}$.

We can now apply this model to the Keccak-$f[1600]$ internal permutation. Some useful intermediate results and relevant $N_{w_{\max}}$ (with their associated p_{match}) are shown in Appendix C.

5.7 Finalizing the Attack and Improvements

In Section 5.4 , we showed how to choose the backward paths given the probability of having a match in the inbound phase (p_{match}) and the number of solution we can generate from this match (N_{match}). In Sections 5.5 and 5.6, we showed how to compute p_{match} and N_{match}. However, in these computations, we needed the probability distribution of the Hamming weight of the input of the Sbox, $\Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}]$. This probability depends greatly on the paths we select in Section 5.4.

To solve this circular dependency, we performed several iterations of the following algorithm until we found some parameters that verify all equations. First, we estimated roughly $\Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}]$ by taking some random backward paths with limited complexity. Using the worst case cost of these paths, we were able to select $w_{\max}$ from Table 5 such that the number of values generated from the inbound is sufficient. Then, we computed p_{match} and N_{match}. With this first guess, we searched for an X and a k such that the we can find a match with a good probability and such that we can generate enough values from the inbound. Then, we computed $\Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}]$ using these new paths generated by X, k and p_B and started our algorithm again with this new distribution. After some iterations, we found a set of filtered backwards paths that provided a sufficient p_{match} and N_{match}.

As discussed in Section 5.4, we noticed the following interesting behavior. By increasing p_B, the expectation of $\Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}]$ is higher. This leads then to a smaller N_{match} and a greater p_{match}. Furthermore, less values need to be generated from the inbound phase since the worst case cost of the backward paths is lower. By taking advantage of this behavior, we were able to reduce significantly the complexity of our attack.

When $(X, k) = (8, 9)$, we have that the number of backward input differences is $\Gamma_B^{\mathsf{in}}(X) = 2^{77.7}$ and $\epsilon = 0.736$. Thus, we have $\Gamma_B^{\mathsf{mid}} = \Gamma_B^{\mathsf{in}} \cdot \epsilon = 2^{77.26}$. If we filter all paths that have a DP smaller than 2^{-461}, i.e., we set $p_B = 2^{-461}$, we get for $(X, k) = (8, 9)$ at least $\epsilon \cdot \Gamma_B^{\mathsf{in}}(X) \cdot \binom{2X}{k} \cdot 2^k \cdot G_B(n) \cdot \tau_B^{\mathsf{full}} = 2^{475.07}$ distinct differences using (14) for the inbound (for these parameters, the difference Hamming weight at the input of the χ layer in the third round is $n = 227.9$). With these parameters, since we remove the paths with a DP lower than p_B, we keep $\tau_B^{\mathsf{DP}} = 36\%$ of the paths, following (13). Hence, we filter the backward paths with a ratio $\tau_B = \tau_B^{\mathsf{full}} \cdot \tau_B^{\mathsf{DP}} = 2^{-15.9} \cdot 0.36 = 2^{-17.37}$. We have also $p_B = 2^{-461}$ and $p_F = 2^{-48}$. Therefore, we need $N_{\mathsf{match}} \geq 2^{509}$. If we consult Table 5, we find that we have to set $w_{\max} = 956$. This leads to $p_{\mathsf{match}} = 2^{-498.11}$. This implies that the minimum total number of backward paths we need to generate is $n_B^{\min} = 2^{492.18}$. All these paths apply on $nr_B = 3$ KECCAK-$f[1600]$ rounds, all with DP higher or equal to $p_B^{\min} = 2^{48-509} = 2^{-461}$.

To summarize, we have that the number of backward output differences is $\Gamma_B^{\mathsf{out}} = n_B^{\min} \cdot \tau_B = 2^{492.18-17.37} = 2^{474.81}$ and that the number of forward input differences is $\Gamma_F^{\mathsf{in}} = 2^{23.3}$. Hence, there is a total of $2^{498.11}$ couples of $(\Delta_B^{\mathsf{out}}, \Delta_F^{\mathsf{in}})$ for the inbound phase, which is enough since it is equal to $1/p_{\mathsf{match}}$. Once a match is found, the worst case complexity of the connected path is $1/(p_B \cdot p_F) \leq 2^{461+48} = 2^{509}$ which is equal to N_{match}. Hence, we can generate enough values from the inbound phase to find with a good probability values verifying the differential path.

The *overall complexity for the rebound attack* given by (1) is $C = 2^{509}$.

This model was verified on the KECCAK-$f[100]$ internal permutation, and by applying this attack on it, we found a 4-round result together with solution pairs, which are shown in Appendix E. This gives a 6-round distinguisher with complexity $2^{28.76}$ which is higher than the simple distinguishers for 6 rounds. However, our goal for KECCAK-$f[100]$ was to verify our model in practice, so that we can be confident for applying it to the KECCAK-$f[1600]$ version. Moreover, finding a solution as in Appendix E is hard since all s Sboxes are active in the middle of the path.

5.8 The distinguisher

We will use exactly the same type of limited-birthday distinguishers as in Section 4. Our rebound attack finds pairs of internal state values such that the input and output difference masks are fully predetermined and we already showed that this should require 2^b operations in the generic case. Therefore, we obtain a $(nr_B + nr_F + 1)$-round distinguisher for the b-bit KECCAK internal permutation considered if the total cost of the rebound attack to find one solution is lower than 2^b.

In the case of KECCAK-$f[1600]$, we have $nr_B = nr_F = 3$ but note that the backward paths utilized do not have the same input difference Δ_B^{in} and the forward paths do not have the same output difference Δ_F^{out}.

Indeed, the backward paths have been generated with $\Gamma_B^{\mathsf{in}}(X)$ distinct starting difference patterns in the second round and thus we have $\Gamma_B^{\mathsf{in}}(X)$ possible input difference masks Δ_B^{in}. Regarding the forward paths, we utilized Γ_F^{out} distinct starting points and thus we have Γ_F^{out} possible output difference masks Δ_F^{out}. Then, our rebound attack finds 7-round pairs of internal state values such that the input and output difference masks lie in a subspace of size $\Gamma_B^{\mathsf{in}}(8) = 2^{77.7}$ and $\Gamma_F^{\mathsf{out}} = 2^6$ respectively, with total complexity 2^{509} (whereas the generic case complexity is $2^{1516.3}$, as explained in Section 4.2).

However, we can attack one more round by adding an extra round to the right of the 7-round path exactly as we did for the distinguishers in Section 4.2. Namely, we have Γ_F^{out} possible output difference masks, all with 6 active bits. Figure 5 depicts the new forward paths. We computed that each one of these

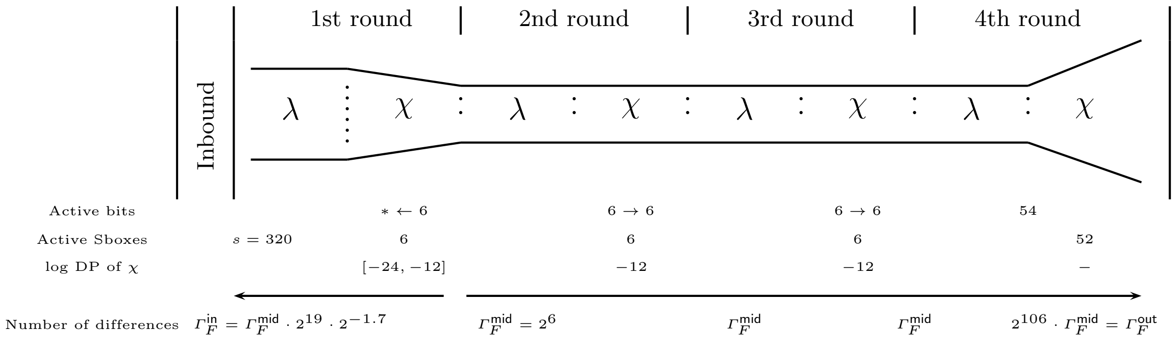

Fig. 5. The forward trails with an extended round. The distance between the two lines reflects the number of differences.

output differences for the 7-round path can be mapped to only 2^{106} output differences for the extended 8-round path. One can calculate this by analysing b_3 in Table 7. We have 50 active Sboxes with one active bit that can produce 2^2 transitions and 2 active Sboxes with two active bits that can produce 2^3 transitions (the number of transitions can be checked in Table 3). Therefore, our rebound attack finds 8-round pairs of internal state values such that the input and output difference masks lie in a subspace of size $\Gamma_B^{\mathsf{in}}(8) = 2^{77.7}$ and $\Gamma_F^{\mathsf{out}} = 2^6 \cdot 2^{106} = 2^{112}$ respectively, with total complexity 2^{509} (whereas the generic case complexity is $2^{1410.3}$).

6 Further Improvements

In the distinguisher presented in the previous section, the bottleneck of the attack came from the outbound phase. However, the gap between its complexity and the generic complexity was enormous. In this section, we reduce the complexity of the attack by narrowing this gap. We reduce the complexity of the distinguisher by relaxing one round in both the forward and the backward direction.

6.1 Relaxing the Forward Paths

The idea is to take the same forward paths than in Section 5.4 but we allow any possible transition in the last round to avoid the cost of this round (see Figure 6). Hence we get this round for free except that we will reduce the generic complexity of the generic distinguisher since we will have more possible output differences. In short, we have to pay only for two rounds with a worst case DP of 2^{-36}.

We analyze now the impact of this modification on Γ_F^{out}, the set of reachable output differences. At the entrance of the third round, every Sbox has one single active bit. Hence, according to the DDT, there are only 4 different possibilities at the output of the Sboxes. Since we have 6 active Sboxes in the third round, the number of possible differences at the output of the third round is multiplied by $4^6 = 2^{12}$. Thus, the number of differences at the output of the third round is $\Gamma_F^{\mathsf{mid}} \cdot 2^{12} = 2^6 \cdot 2^{12} = 2^{18}$.

We need now to look at the fourth round to obtain Γ_F^{out} and compute the generic complexity of the distinguisher. In the third round, every active Sbox can produce at most 3 active bits at its output, since each active Sbox has only one single active bit at its input. Hence, the maximum Hamming weight at the

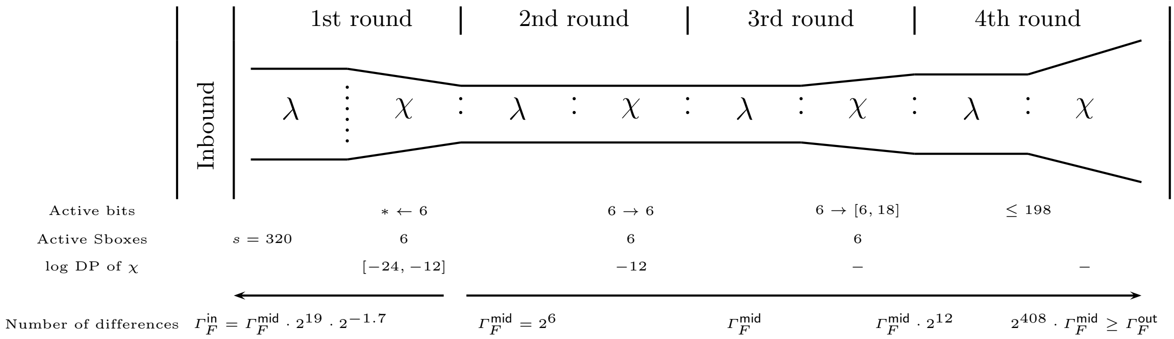

Fig. 6. The forward trails when we relax the 3rd round. The distance between the two lines reflects the number of differences.

output is $3 \cdot 6 = 18$. Each of these active bits can be expanded to at most 11 bits through θ and hence, we have at most $11 \cdot 18 = 198$ active bits at the input of the Sboxes of the fourth round. In the worst case, each of these bits will be in a different Sbox and will produce four possible differences. Hence, we have $\Gamma_F^{\mathsf{out}} \leq \Gamma_F^{\mathsf{mid}} \cdot 2^{12} \cdot 4^{198} = 2^{18} \cdot 2^{396} = 2^{414}$.

To conclude, we reduced the complexity of the attack by a factor 2^{12} and increased the number of possible outputs for the distinguisher by a factor 2^{302}.

6.2 Relaxing the Backward Paths

We can do the same kind of operation for the backward paths by relaxing the first round of each path (see Figure 7). Instead of picking only $1 \mapsto 1$ transitions, i.e., transition that maps one bit to a single bit in the

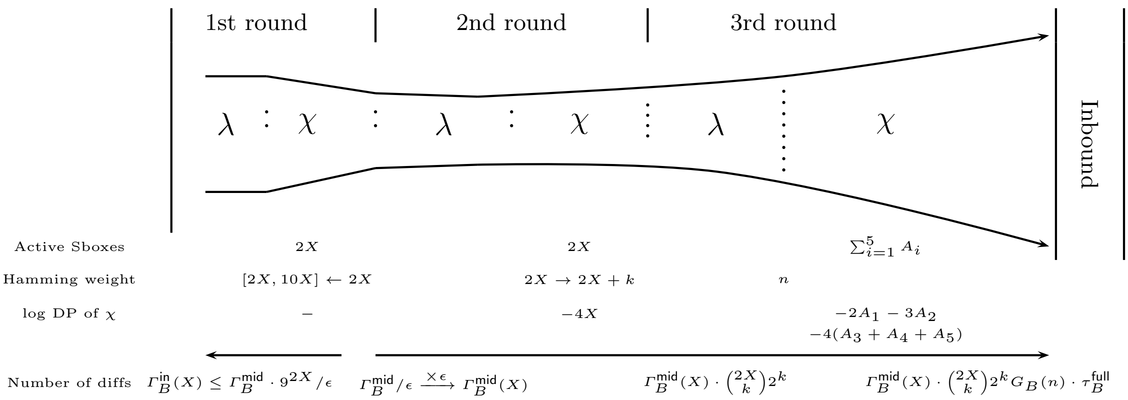

Fig. 7. The backward trails when we relax the first round. The distance between the two arrows reflects the number of differences.

χ^{-1} layer, we allow any possible transition and, hence, do not have to pay for it. This reduces the complexity of the backward paths by a factor 2^{4X}. Each Sbox with one single active bit at its output can have 9 possible input differences and the maximum possible of input differences that can occur for a given input difference is 12 (see χ^{-1} DDT). Since we have $2X$ active Sboxes, the number of possible input differences is increased by a factor of at most 9^{2X}. Therefore, $\Gamma_B^{\mathsf{in}} \leq \Gamma_B^{\mathsf{mid}} \cdot 9^{2X}/\epsilon$ and we reduced the complexity by a factor 2^{4X}.

6.3 Finding New Parameters

Now that we have reduced the cost of the forward paths by 2^{12} and the cost of the backward paths by 2^{4X}, we do not need to generate that many values in the outbound phase. Furthermore, the bottleneck of the attack is now in the inbound phase, i.e., it comes from p_{match}.

Our goal is now to increase p_{match} so that the complexity of the attack is reduced. There are two ways of increasing p_{match}.

- Recall that a way of increasing p_{match} is to allow again a match to happen at a higher Hamming weight at the input of the Sboxes of the inbound round. These matches are much more probable and will increase p_{match}. However, as discussed in Section 5.6, N_{match} drops in this case. This is not an issue anymore, since we need much less values to verify the differential paths.
- Another solution is to modify the Hamming weight at the input of the Sboxes of the inbound round. This can be done by modifying X or k.[9] We will see that we can reduce k since less paths will be filtered because of their DP.

We found that for $(X, k) = (8, 8)$, we can get at least $\epsilon \cdot \binom{5}{2}^X \cdot \binom{s}{X} \cdot \binom{2X}{k} \cdot 2^k \cdot G_B(n) \cdot \tau_B^{\mathsf{full}} = 2^{493.88} \cdot 2^{-15.9} = 2^{477.98}$ distinct differences with worst case DP $p_B = 2^{-450}$ and this, without almost any filtering on the DP, i.e., $\tau_B^{\mathsf{DP}} \approx 1 - 10^{-10}$. Note that for these parameters, $\epsilon = 0.736$ as before and $n = 220.61$.

The worst case total cost of the outbound phase is $1/(p_B \cdot p_F) = 2^{450+36} = 2^{486}$. Remark that we need a N_{match} much lower than before (2^{486} instead of 2^{509}). By looking into Table 5 in Appendix C we remark that any $w_{\max}$ in the table will produce enough values in the outbound phase and we select $w_{\max} = 1000$. The Hamming weight distribution at the input of the Sboxes of the inbound phase ($\Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}]$) when $k = 8$ behaves like $\mathcal{N}(847.47, 651.51)$. With this new distribution, we can compute the new p_{match} for $w_{\max} = 1000$ and we obtain $p_{\mathsf{match}} = 2^{-491.47}$ and $N_{\mathsf{match}} = 2^{486.81}$. Since Γ_F^{in} is still equal to $2^{23.3}$, the total number of backward paths we have to generate is $n_B^{\min} = 1/(p_{\mathsf{match}} \cdot \Gamma_F^{\mathsf{in}} \cdot \tau_B) = 1/(p_{\mathsf{match}} \cdot \Gamma_F^{\mathsf{in}} \cdot \tau_B^{\mathsf{full}}) = 2^{484.07}$. We have then that the total number of backward output differences is $\Gamma_B^{\mathsf{out}} = n_B^{\min} \cdot \tau_B = 2^{468.17}$. Hence, there is a total of $2^{491.47}$ couples of $(\Delta_B^{\mathsf{out}}, \Delta_F^{\mathsf{in}})$ for the inbound phase, which is enough since it is equal to $1/p_{\mathsf{match}}$. Once a match is found, the worst case complexity of the connected path is $1/(p_B \cdot p_F) = 2^{486}$ which is lower than N_{match}. Thus, we can generate enough values from the inbound phase to find with a good probability values verifying the differential path.

The new overall complexity for the rebound attack given by (1) is $C = 2^{491.47}$. Note that the bottleneck now comes from p_{match}.

We have $\Gamma_B^{\mathsf{in}} \le \Gamma_B^{\mathsf{mid}}(8) \cdot 9^{2 \cdot 8}/\epsilon = 2^{77.7+50.7} = 2^{128.4}$ and $\Gamma_F^{\mathsf{out}} \le 2^{414}$. The new generic complexity of the distinguisher is, hence, greater than $2^{1057.6}$.

7 Results and Conclusion

Table 2. Best differential distinguishers complexities for each version of KECCAK internal permutations, for 1 up to 8 rounds. Note that due to its technical complexity when applied on KECCAK, the rebound attack has only been applied to KECCAK-f[100] and KECCAK-f[1600]. In brackets, the generic complexity.

b	best differential distinguishers complexity							
	1 rd	2 rds	3 rds	4 rds	5 rds	6 rds	7 rds	8 rds
100	1	1	1	2^2	2^8	2^{19}	-	-
200	1	1	1	2^2	2^8	2^{20}	2^{46}	-
400	1	1	1	2^2	2^8	2^{24}	2^{84}	-
800	1	1	1	2^2	2^8	2^{32}	2^{109}	-
1600	1	1	1	2^2	2^8	2^{32}	2^{142}	$2^{491.47}[2^{1057.6}]$

In this article, we analysed the internal permutations used in the KECCAK family of hash functions in regards to differential cryptanalysis. We first proposed a generic method that looks for the best differential paths using CP-kernel considerations and better χ mapping. This new method provides some of the best

[9] In fact experiments have shown that lowering k leads to a better Hamming weight distribution for $X = 8$.

known differential paths for the Keccak internal permutations and we derived distinguishers with rather low complexity exploiting these trails. In particular we were able to obtain a practical distinguisher for 6 rounds of the Keccak-f[1600] permutation. Then, aiming for attacks reaching more rounds, we adapted the rebound attack to the Keccak case. This adaptation is far from trivial and contains many technical details. Our model was verified by applying the attack on the reduced version Keccak-f[100]. The main final result is a 8-round distinguisher for the Keccak-f[1600] internal permutation with a complexity of $2^{491.47}$. All our distinguisher results are summarized in Table 2. Note that our attack does not endanger the security of the full Keccak. We believe that this work will also help to apply the rebound attack on a much larger set of primitives.

This work might be extended in many ways, in particular by further refining the differential path search or by improving the inbound phase of the rebound attack such that the overall cost is reduced. Moreover, another research direction would be to analyse how the differential paths derived in this article can lead to collision attacks against reduced versions of the Keccak hash functions.

Acknowledgements. The authors would like to thank Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche for their helpful comments. Also, we are very grateful to Praveen Gauravaram, Tao Huang, Phuong Ha Nguyen, Wun-She Yap, Przemyslaw Sokolowski and Wenling Wu for useful discussions.

References

1. Masayuki Abe, editor. *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *LNCS*. Springer, 2010.
2. Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. Presented at the rump session of CHES 2009, 2009.
3. Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *CCS*, pages 62–73. ACM, 1993.
4. Daniel J. Bernstein. Second preimages for 6 (7? (8??)) rounds of Keccak?, November 2010. Available at `http://ehash.iaik.tugraz.at/uploads/6/65/NIST-mailing-list_Bernstein-Daemen.txt`.
5. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Sponge functions. ECRYPT Hash Workshop 2007, May 2007.
6. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Keccak specifications. Submission to NIST (Round 1), 2008. Available at `http://keccak.noekeon.org/Keccak-specifications.pdf`.
7. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Keccak specifications. Submission to NIST (Round 2), 2009. Available at `http://keccak.noekeon.org/Keccak-specifications-2.pdf`.
8. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On alignment in Keccak. ECRYPT II Hash Workshop, 2011.
9. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The KECCAK Reference. Submission to NIST (Round 3), 2011. Available at `http://keccak.noekeon.org/Keccak-reference-3.0.pdf`.
10. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The KECCAK SHA-3 Submission. Submission to NIST (Round 3), 2011. Available at `http://keccak.noekeon.org/Keccak-submission-3.pdf`.
11. Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of Keccak and Luffa. In *FSE*, 2011, to appear.
12. Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In Hong and Iwata [13], pages 365–383.
13. Seokhie Hong and Tetsu Iwata, editors. *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, volume 6147 of *LNCS*. Springer, 2010.
14. Dmitry Khovratovich, María Naya-Plasencia, Andrea Röck, and Martin Schläffer. Cryptanalysis of *uffa* v2 Components. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 388–409. Springer, 2010.
15. Dmitry Khovratovich, Ivica Nikolic, and Christian Rechberger. Rotational Rebound Attacks on Reduced Skein. In Abe [1], pages 1–19.
16. Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound Distinguishers: Results on the Full Whirlpool Compression Function. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 126–143. Springer, 2009.

17. Florian Mendel, Thomas Peyrin, Christian Rechberger, and Martin Schläffer. Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography*, volume 5867 of *LNCS*, pages 16–35. Springer, 2009.
18. Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *FSE*, volume 5665 of *LNCS*, pages 260–276. Springer, 2009.
19. Pawel Morawiecki and Marian Srebrny. A SAT-based preimage analysis of reduced Keccak hash functions. Presented at Second SHA-3 Candidate Conference, Santa Barbara 2010, 2010.
20. National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard. `http://csrc.nist.gov`, April 1995.
21. National Institute of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a NewCryptographic Hash Algorithm (SHA-3) Family. *Federal Register*, 27(212):62212–62220, November 2007. Available:`http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`(2008/10/17).
22. Vincent Rijmen, Deniz Toz, and Kerem Varici. Rebound Attack on Reduced-Round Versions of JH. In Hong and Iwata [13], pages 286–303.
23. Ronald L. Rivest. The MD5 message-digest algorithm. Request for Comments (RFC) 1320, Internet Activities Board, Internet Privacy Task Force, April 1992.
24. Yu Sasaki, Yang Li, Lei Wang, Kazuo Sakiyama, and Kazuo Ohta. Non-full-active Super-Sbox Analysis: Applications to ECHO and Grøstl. In Abe [1], pages 38–55.
25. Keccak team. Keccak Crunchy Crypto Collision and Pre-image Contest, 2011.
26. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
27. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

A Plot for the Pattern-match Problem

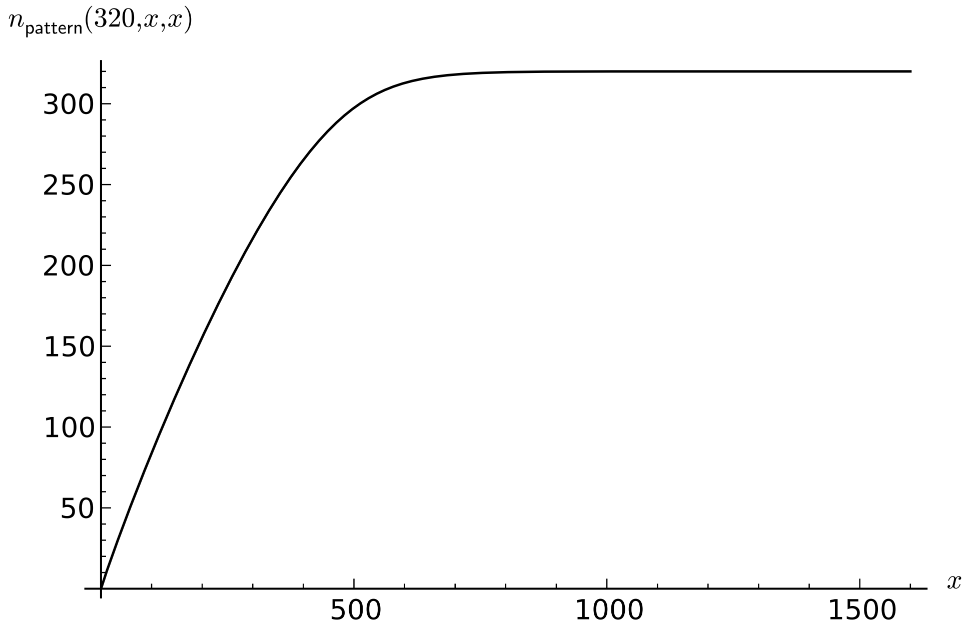

Fig. 8. Average number of non empty buckets when a pattern-match occurs for $x_B = x_F$ and $s = 320$

B Differential Distribution Table of χ and χ^{-1}

Table 3. The differential distribution table of the χ when viewed as Sbox. The first bit of a row is viewed as the least significant bit. Given input difference Δ_{in} and output difference Δ_{out} the number in the table shows the size of the solution set $\{v \mid \chi(v) \oplus \chi(v \oplus \Delta_{in}) = \Delta_{out}\}$. Differences are in hex number.

Δ_{in} \ Δ_{out}	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
00	32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
01	-	8	-	-	-	-	-	-	-	8	-	-	-	-	-	-	-	8	-	-	-	-	-	-	-	8	-	-	-	-	-	-
02	-	-	8	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	8	-	-	-	-	-	-	-	-	-	-	-	-
03	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-	-	-	4	4	-	-	-	-
04	-	-	-	-	8	8	8	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
05	-	-	-	-	4	-	4	-	-	-	-	-	4	-	4	-	-	-	-	-	-	4	-	4	-	-	-	-	-	4	-	4
06	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	4	4	4	4	-	-	-	-	-	-	-	-
07	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2
08	-	-	-	-	-	-	-	-	8	-	8	-	8	-	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
09	-	4	-	4	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-	-	-	-	-	-	4	-	4
0A	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4
0B	-	4	4	-	-	-	-	-	-	-	-	-	-	4	4	-	-	4	4	-	-	-	-	-	-	-	-	-	-	4	4	-
0C	-	-	-	-	-	-	-	-	4	4	4	4	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0D	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	-	-	-
0E	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2
0F	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	-	-	-	8	-	-	-	8	-	-	-	8	-	-	-
11	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-	-	4	-	-
12	-	-	4	4	-	-	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	-	-	4	4
13	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	-	-	-	-	4	4	4	4	-	-	-	-	4	4
15	-	4	-	-	-	-	-	4	-	4	-	-	-	-	-	4	4	-	-	-	-	-	4	-	4	-	-	-	-	-	4	-
16	-	-	4	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	4	4	-	-
17	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-	-	-	2	2	2	2	-	-
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	-	4	-	4	-	4	-	4	-	4	-	4	-	4	-
19	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2
1A	-	-	-	-	-	-	-	-	4	-	-	4	4	-	-	4	4	-	-	4	4	-	-	4	-	-	-	-	-	-	-	-
1B	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-	-	2	2	-
1C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1D	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	2	-	2	-	2	-	2	-	2	-	2	-	2	-	2	-
1E	-	-	-	-	-	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-
1F	-	2	2	-	2	-	-	2	2	-	-	2	-	2	2	-	2	-	-	2	-	2	2	-	-	2	2	-	2	-	-	2

C Distributions for the KECCAK-f[1600] rebound attack

In this section, we give all the probability distributions necessary to compute p_{match} and N_{match}. We give also some intermediary results.

C.1 P_{out}

P_{out} is the measured probability distribution at the output of an Sbox when all Sboxes are active. Recall that due to the properties of the DDT, we can consider every Sbox independently. The probability distribution is shown in Table 4.

Table 4. Distribution of P_{out} for our attack on KECCAK-f[1600]

y	$P_{\mathsf{out}}(y)$	y	$P_{\mathsf{out}}(y)$
0	0.00000000	10	0.01610571
1	0.01859357	11	0.04363845
2	0.01868644	12	0.04744619
3	0.04682032	13	0.02290569
4	0.01784977	14	0.04548532
5	0.04445059	15	0.02096362
6	0.04309371	16	0.01997166
7	0.02287948	17	0.04545379
8	0.01802242	18	0.04506549
9	0.04543792	19	0.02241770
a	0.04612980	1a	0.01806761
b	0.02005349	1b	0.04705496
c	0.04726458	1c	0.02056551
d	0.02075547	1d	0.04800789
e	0.02080299	1e	0.04603740
f	0.04910975	1f	0.01086272

C.2 $\Pr[\mathsf{Hw}_{\mathsf{total}} = w|\mathsf{full}]$

The computation of p_{match} depends greatly on the Hamming weight distribution at the input of the Sboxes of the inbound phase. We measured this distribution by taking a fair amount of samples and noticed that it behaves like a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, with mean μ and variance σ^2. This behavior is shown in Figure 9. For the backward paths we are using in our first attack (Section 5), i.e., when $X = 8$ and $k = 9$, this distribution behaves as $\mathcal{N}(847.88, 666.34)$. In the improved attack (Section 6,i.e., when $X = 8$ and $k = 8$, this distribution behaves as $\mathcal{N}(847.47, 651.51)$.

C.3 $\mathbb{E}[N_{\mathsf{SBox}}|\mathsf{match}, \mathsf{Hw}_{\mathsf{SBox}} = i]$

This values is used to compute N_{match}. It is defined as

$$\frac{1}{\binom{5}{i}} \sum_{\substack{v \in \{0,1\}^5: \\ \mathsf{Hw}(v)=i}} DDT[v] \ .$$

For $i = \{1, 2, 3, 4, 5\}$ respectively, $\mathbb{E}[N_{\mathsf{SBox}}|\mathsf{match}, \mathsf{Hw}_{\mathsf{SBox}} = i] = \{8, 4, 3, 2, 2\}$.

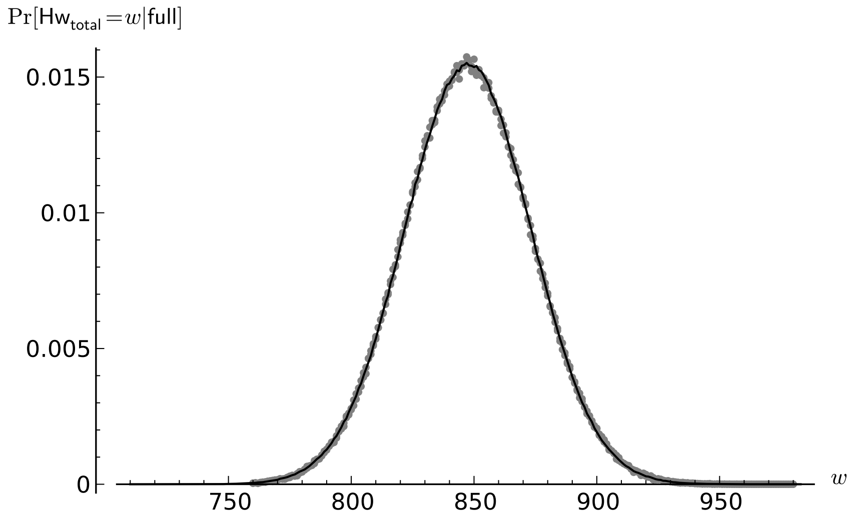

Fig. 9. $\mathsf{Hw}_{\mathsf{total}}$ (in gray) is distributed following a Gaussian distribution(in black).

C.4 Behavior of N_w and $p_{\mathsf{match}}(w)$

Figure 10 shows how N_w (resp. $p_{\mathsf{match}}(w)$) decreases (resp. increases) exponentially when w grows.

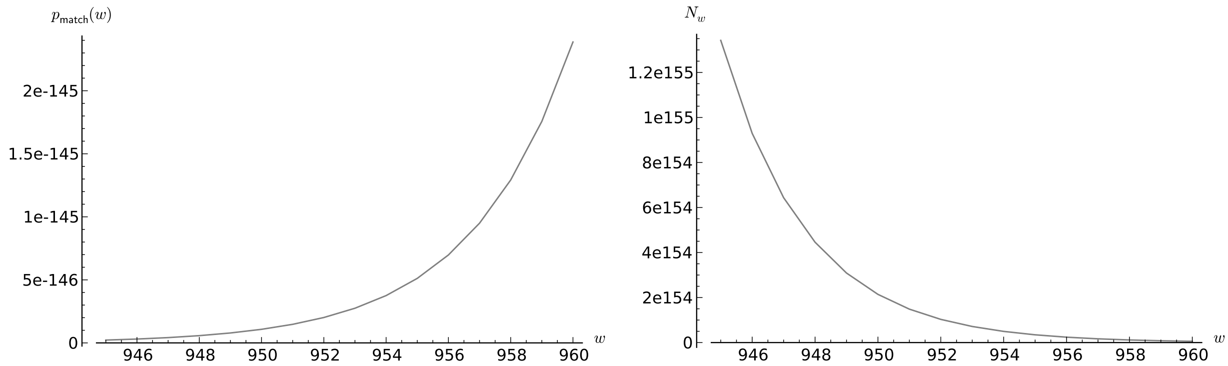

Fig. 10. Behavior of N_w and $p_{\mathsf{match}}(w)$

C.5 Different $N_{w_{\max}}$ and p_{match} for relevant $w_{\max}$

Given the complexity of the backward and forward differential path, we need to ensure that we generate enough values from the inbound phase. Table 5 shows the resulting worst case N_{match} and p_{match} for some relevant $w_{\max}$.

Table 5. Values for $N_{w_{\max}}$ and p_{match} for different relevant $w_{\max}$. For p_{match} the first Hamming weight distribution is used, i.e., the distribution when $X = 8$ and $k = 5$.

$w_{\max}$	$\log(N_{w_{\max}})$	$\log(p_{\mathsf{match}})$	$w_{\max}$	$\log(N_{w_{\max}})$	$\log(p_{\mathsf{match}})$
935	520.65	-503.58	968	503.23	-495.57
936	520.11	-503.29	969	502.71	-495.38
937	519.58	-503.00	970	502.19	-495.19
938	519.05	-502.72	971	501.67	-495.00
939	518.51	-502.44	972	501.15	-494.82
940	517.98	-502.16	973	500.64	-494.64
941	517.45	-501.88	974	500.12	-494.46
942	516.91	-501.61	975	499.60	-494.29
943	516.38	-501.34	976	499.09	-494.12
944	515.85	-501.07	977	498.57	-493.95
945	515.32	-500.81	978	498.05	-493.78
946	514.79	-500.55	979	497.54	-493.62
947	514.26	-500.29	980	497.02	-493.46
948	513.73	-500.03	981	496.51	-493.30
949	513.20	-499.78	982	496.00	-493.15
950	512.67	-499.53	983	495.48	-493.00
951	512.14	-499.29	984	494.97	-492.85
952	511.61	-499.05	985	494.46	-492.70
953	511.09	-498.81	986	493.94	-492.56
954	510.56	-498.57	987	493.43	-492.42
955	510.03	-498.34	988	492.92	-492.29
956	509.51	-498.11	989	492.41	-492.15
957	508.98	-497.88	990	491.90	-492.02
958	508.46	-497.65	991	491.39	-491.89
959	507.93	-497.43	992	490.88	-491.76
960	507.41	-497.21	993	490.37	-491.64
961	506.88	-497.00	994	489.87	-491.52
962	506.36	-496.78	995	489.36	-491.40
963	505.84	-496.57	996	488.85	-491.29
964	505.32	-496.37	997	488.35	-491.18
965	504.79	-496.16	998	487.84	-491.07
966	504.27	-495.96	999	487.33	-490.96
967	503.75	-495.77	1000	486.81	-490.86

D Differential paths for KECCAK-f[1600]

Table 6. Example of 3-round differential path for KECCAK-f[1600] with probability 2^{-32}. The first 2 rounds form a best 2-round differential path with probability 2^{-8}.

	F135E26BC449789A	4D789AF135A226BC	5E26BC4D78B5AF13	89AF135E268FC4D7	6BC4D789AF1D35E2	
	F135E26BC449789A	4D789AF135A226BC	5E26BC4D78B5AF13	89AF135E268FC4D7	6BC4D789AF1D35E2	
a_0	F135E26BC449789A	4D789AF135A226BC	5E26BC4D78B5AF13	89AF135E268FC4D7	6BC4D789AF1D35E2	
	F135E26BC4C9789A	4D789AF135AA26BC	5E26BC4D78B5AF13	89AF135E268FC4D7	6BC4D789AF1D35E2	
	F135E26BC449789A	4D789AF135A226BC	5E26BC4D78B5AF13	89AF135E268FC4D7	6BC4D789AF1D35E2	
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	---------------1	----------------	
b_0	----------------	----------------	----------------	----------------	----------------	2^{-4}
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	---------------1	----------------	
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	---------------1	----------------	
a_1	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	---------------1	----------------	
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	----------------	----------------	
b_1	----------------	----------------	----------------	----------------	----------------	2^{-4}
	----------------	----------------	----------------	----------------	-1--------------	
	----------------	--8-------------	----------------	----------------	----------------	
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	----------------	----------------	
a_2	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	----------------	-1--------------	
	----------------	--8-------------	----------------	----------------	----------------	
	----------------	----------------	-------4--------	------------4---	----------------	
	----------2-----	----------------	----------------	----------------	--1-------------	
b_2	----------------	2---------------	-----------4----	---------------1	----------------	2^{-24}
	----------------	----------------	----------------	--------------4-	---2------------	
	--2-------------	---1------------	----------------	----------------	-2--------------	
	----------------	----------------	-------4--------	------------4---	----------------	
	----------2-----	----------------	----------------	----------------	--1-------------	
a_3	----------------	2---------------	-----------4----	---------------1	----------------	
	----------------	----------------	----------------	--------------4-	---2------------	
	--2-------------	---1------------	----------------	----------------	-2--------------	

Table 7. Example of 4-round differential path for KECCAK-f[1600] with probability 2^{-142}. The first 3 rounds forming a 3-round differential path with probability 2^{-36}, is used in the rebound attack as forward path.

	D78BE9AF44D1AF44	E26A4BC4B5E3C4B5	9AF-D135F88935F8	BC4D2D79A6B579A6	135AFE262F16262F	
	D78BE9AF44D1AF44	E26A-BC4B5E3C4B5	9AF-D135F89935F8	BC4D2D79A6B579A6	135AFE262F16262F	
a_0	D78BE9AF44D1AF44	E26ACBC4B5E3C4B5	9AF-D135F88935F8	BC4D2D78A6B579A6	135AFE262F16262F	
	D78BE9AF44D1AF44	E26A4BC4B5E3C4B5	9AF-D135F88935F8	BC4D2D79A6B579A6	135AFE262F16262F	
	D78BE9AF44D1AF44	E26A4BC4B5E3C4B5	9AF-D135F88935F8	BC4D2D79A6B578A6	135EFE262F16262F	
	----------------	---------4------	----------------	----------------	---------------1	
	----------------	----------------	----------------	----------------	----------------	
b_0	----------------	---------4------	-2--------------	----------------	----------------	2^{-12}
	----------------	----------------	-2--------------	----------------	---------------1	
	----------------	----------------	----------------	----------------	----------------	
	----------------	---------4------	----------------	----------------	---------------1	
	----------------	----------------	----------------	----------------	----------------	
a_1	----------------	---------4------	-2--------------	----------------	----------------	
	----------------	----------------	-2--------------	----------------	---------------1	
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	------1---------	----------------	----------------	
	----------------	----------------	----------------	----------------	----------------	
b_1	---------8------	----------------	----------------	-------------1--	----------------	2^{-12}
	---------8------	----------------	------1---------	-------------1--	----------------	
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	------1---------	----------------	----------------	
	----------------	----------------	----------------	----------------	----------------	
a_2	---------8------	----------------	----------------	-------------1--	----------------	
	---------8------	----------------	------1---------	-------------1--	----------------	
	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	----------------	--------2-------	----------------	
	----------------	----------------	--------4-------	----------------	----------------	
b_2	----------------	----------------	-------2--------	----------------	----------------	2^{-12}
	----------------	----------------	----------------	---8------------	----------------	
	-------4--------	----------------	----------------	--------------1-	----------------	
	----------------	----------------	----------------	--------2-------	----------------	
	----------------	----------------	--------4-------	----------------	----------------	
a_3	----------------	----------------	-------2--------	----------------	----------------	
	----------------	----------------	----------------	---8------------	----------------	
	-------4--------	----------------	----------------	--------------1-	----------------	
	----------------	-------------8--	---1----8---12--	--48---------1--	---2-8-----4---2	
	26--------------	--82-----1----8-	----------------	------------1---	---2-----8-----4	
b_3	-------1--------	-4-----------8--	-48-------------	-8---82-----1---	--1-------------	2^{-106}
	41------8---4---	----------------	-----2----------	----2-----1----8	1--------24-----	
	---4----1------8	---------12-----	-----8---4---41-	----------------	-------2--------	
	----------------	-------------8--	---1----8---12--	--48---------1--	---2-8-----4---2	
	26--------------	--82-----1----8-	----------------	------------1---	---2-----8-----4	
a_4	-------1--------	-4-----------8--	-48-------------	-8---82-----1---	--1-------------	
	41------8---4---	----------------	-----2----------	----2-----1----8	1--------24-----	
	---4----1------8	---------12-----	-----8---4---41-	----------------	-------2--------	

Table 8. Example of 5-round differential path for KECCAK-f[1600] with probability 2^{-709}.

	4----2A6981735E2	C----314BD2F789A	4-----9913DC26BC	C----9C6D77FAF13	C----48BE251C4D7	
	4----2A6981735E2	C----314BD2F789A	4-----9913DC26BC	4----9C6D77FAF13	C----48FE251C4D7	
a_0	4----2A69A1735E2	C----314BD2B789A	4-----9913DC26BC	C----9C6D77FAF13	C----48BE251C4D7	
	4----2A6989735E2	C----314BD27789A	4-----9913DC26BC	C----9C6D77FAF13	C----48BE251C4D7	
	4----2A6981735E2	C----314BD2F789A	4-----D913DC26BC	C----1C6D77FAF13	C----48BE251C4D7	
	----------------	----------------	----------------	----------------	----------------	
	----------------	--4-------------	--------1-------	---------------1	-------8--------	
b_0	----------------	----------------	----------------	----------------	----------------	2^{-16}
	----------------	----------------	--------1-------	----------------	-------8--------	
	----------------	--4-------------	----------------	---------------1	----------------	
	----------------	----------------	----------------	----------------	----------------	
	----------------	--4-------------	--------1-------	---------------1	-------8--------	
a_1	----------------	----------------	----------------	----------------	----------------	
	----------------	----------------	--------1-------	----------------	-------8--------	
	----------------	--4-------------	----------------	---------------1	----------------	
	----------------	-------4--------	----------------	----------------	----------------	
	----------------	--8-------------	----------------	----------------	----------------	
b_1	----------------	-------4--------	----------------	-----8----------	----------------	2^{-16}
	----------------	----------------	----------------	-----8----------	-1--------------	
	----------------	--8-------------	----------------	----------------	-1--------------	
	----------------	-------4--------	----------------	----------------	----------------	
	----------------	--8-------------	----------------	----------------	----------------	
a_2	----------------	-------4--------	----------------	-----8----------	----------------	
	----------------	----------------	----------------	-----8----------	-1--------------	
	----------------	--8-------------	----------------	----------------	-1--------------	
	----------------	-------8--------	----------------	---------------1	--------------4-	
	----------------	----------------	----------------	----------------	----------------	
b_2	-------8--------	----------------	--------------1-	---------------1	----------------	2^{-16}
	----------------	----------------	----1-----------	----------------	----------------	
	----------------	----------------	----------------	----------------	-2--------------	
	-------8--------	-------8--------	----------------	---------------1	--------------4-	
	----------------	----------------	----------------	----------------	----------------	
a_3	-------8--------	----------------	--------------1-	---------------1	----------------	
	----------------	----------------	----1-----------	----------------	----------------	
	----------------	----------------	----------------	----------------	-2--------------	
	-2----18------4-	---2-----2------	----8-------4---	--------12--8--2	--------------8-	
	-------91-4--1--	----------------	1-----C------2--	---4-----4------	-------1--------	
b_3	----4-1-------4-	-----2----------	-------122-8--2-	----------------	--4------1---8--	2^{-114}
	-------2--------	-----4--2----1--	--8---------8---	-8-4------------	9--4--1---------	
	-------2--------	48-2---8--------	----------------	----8--4----2---	----8---------8-	
	-2----18------4-	---2------------	----8----2--C---	--------12--8--2	--------------8-	
	-------91-4--1--	----------------	1-----C1-----2--	---4----14------	----------------	
a_4	----4-11------4-	-----2--2-------	-------122-8--2-	----------------	--4---1--1---8--	
	-------2-----1--	-8---4--2----1--	1-8---------8---	-8-4------------	9--4--1---------	
	-------2--------	48-2---8--------	----8-----------	-------4----2---	----8---------8-	
	12448C-8-1---A4-	-844--3--418-5-5	4A-9228---3--97-	18-441-A4425-1A-	---1-9A-282-83--	
	-3--8D2232--9--C	--4268-A--2-C--8	82266--8-8--52--	1-A8-16--8B--E-A	8A--1-C--18-282-	
b_4	-6-483--A-A1-88-	---1B-43--5-C11-	8-441-A4424-12-1	-48-14268-A--29C	3-48-4--28--4112	2^{-547}
	2134-5-41-6--4--	14--B--1-448C191	-1-E-1C14211---C	-3---6--E-822-4-	212--9--C422-872	
	14--218--38-6-41	1-9--68-6-1B-429	C-54-1-62-4--A13	--16--2-891824-2	2C-9-621414211-1	
	1--48C-8-5---24-	--41--3--4-8-425	42-963----11-8D-	-8--4--244-5--A-	---1-18-2C2-814-	
	-32-8D2832--1--4	--4--8-2----8--8	---66-48----5--8	1-A81----83-26-E	884-18E--18-382-	
a_5	-6---32-A-A1188-	-4C1B-41--1-C18-	8-44--86424-5285	-28-12262--1-29C	3-48-44-88-----2	
	213-24-4142-4484	14--B6C1445BC195	-2-E--C12211--4C	-234-2-4A4A22-2-	----1D-1C4628862	
	141426-4-3816E4-	1-C--6A24-43-438	C-45-3-76-4--A11	181E--2-C89A-4-2	38-9--2161C231--	

E Example of valid pairs for Keccak-f[100]

Table 9. Example of 4-round differential path for Keccak-f[100] and valid pairs verifying this path. Each 4-bit hex number represents the lane value with LSB for the first slice.

Position	Difference						Value 1						Value 2					Complexity
	-	-	-	2	4		2	9	4	E	7		2	9	4	C	3	
	-	-	-	-	-		4	5	2	D	9		4	5	2	D	9	
a_0	-	-	-	-	-		-	F	7	7	8		-	F	7	7	8	2^8
	-	-	-	2	4		3	6	8	5	4		3	6	8	7	-	
	-	-	-	-	-		E	B	3	3	F		E	B	3	3	F	
	-	-	-	4	-		4	B	C	8	3		4	B	C	C	3	
	2	-	-	-	-		F	D	2	D	3		D	D	2	D	3	
a_1	-	-	-	4	-		2	3	6	5	B		2	3	6	1	B	2^8
	2	-	-	-	-		1	5	6	-	6		3	5	6	-	6	
	-	-	-	-	-		3	6	F	C	1		3	6	F	C	1	
	-	-	-	-	-		6	5	6	5	A		6	5	6	5	A	
	4	-	-	4	-		1	-	1	9	8		5	-	1	D	8	
a_2	8	8	8	-	-		C	3	B	D	F		4	B	3	D	F	rebound
	2	2	-	-	2		4	B	A	C	F		6	9	A	C	D	
	-	4	4	4	-		C	8	D	2	D		C	C	9	6	D	
	F	6	5	B	-		B	5	8	3	5		4	3	D	8	5	
	7	E	1	B	1		D	2	C	E	E		A	C	D	5	F	
a_3	D	6	1	B	2		3	A	3	6	A		E	C	2	D	8	2^{11}
	7	E	1	B	-		-	4	B	9	2		7	A	A	2	2	
	F	6	9	B	-		2	E	2	2	F		D	8	B	9	F	
	-	8	-	-	-		-	5	2	4	4		-	D	2	4	4	
	-	-	-	1	-		C	5	A	9	A		C	5	A	8	A	
a_4	-	-	-	-	-		F	2	2	3	-		F	2	2	3	-	
	-	8	-	-	-		D	9	1	E	7		D	1	1	E	7	
	-	-	-	1	-		D	8	3	A	1		D	8	3	B	1	