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, Swizerland
[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.

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

[*] Part of the work was done while the author was visiting Nanyang Technological University, supported by the NTU NAP Startup Grant M58110000. This work has also partially been supported by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

[†] Part of the work was done while the author was visiting Tsinghua University, supported by the National Natural Science Foundation of China under grant No. 61133013 and No. 60931160442.

[‡] The author is supported by the Lee Kuan Yew Postdoctoral Fellowship 2011 and the Singapore National Research Foundation Fellowship 2012.

[§] The author is supported by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03, the Singapore Ministry of Education under Research Grant T206B2204 and by the NTU NAP Startup Grant M58110000.

most important primitives in cryptography. 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.

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. Zero-sum distinguishers [2] can reach an important number of rounds, but generally with a very high complexity. For example, the latest results [9] provide zero-sum partitions distinguishers for the full 24-round 1600-bit internal permutation with a complexity of 2^{1575}. 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 rounds using SAT-solvers have been demonstrated [18]. Finally, Bernstein 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 [17] 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 [21, 13]. One reason for that is that KECCAK has *weak alignment* [6]. This is why we call our attack "unaligned rebound attack". 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 [23] and managed to obtain the currently best known practical collision attack for up to two rounds.

2 The KECCAK Hash Function Family

KECCAK [7, 8] 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 $b = r + c$.

2.1 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.

$\boldsymbol{\theta}$ mapping. This linear mapping intends to provide diffusion for the state and is defined for every x, y and z by: $\theta : a[x][y][z] \leftarrow a[x][y][z] + \bigoplus_{y'=0}^{4} a[x-1][y'][z] + \bigoplus_{y'=0}^{4} a[x+1][y'][z-1]$.

$\boldsymbol{\rho}$ 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.

$\boldsymbol{\pi}$ mapping. This linear mapping intends to provide diffusion in the state through transposition of the lanes.

$\boldsymbol{\chi}$ 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. This mapping adds constants derived from an LFSR to the lane $a[0][0][\cdot]$. These constants are different in every round i. This mapping aims at destroying the symmetry introduced by the identical nature of the remaining mappings in a round.

3 Finding Differential Paths for KECCAK-f

We first study how to find "good" differential paths for all KECCAK variants. In this section, we describe our differential finding algorithms.

3.1 Special properties of θ and χ

It is noted by the Keccak designers [7, Section 2.4.3] that when every column of the state sums to 0, θ acts as identity. The set of such states is called *column parity kernel* (*CPK*). 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 acts as identity on differences in the CPK. 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} [7, 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 is a row-wise operation and can also be viewed as a 5-bit Sbox. Similar to the analysis of other Sboxes, we build its differential distribution table (DDT). We remark that when a single difference is present, χ acts as identity with best probability 2^{-2}, while input differences with more active bits tend 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}).

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 CPK (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 , it finds the best slice-wise mappings with help of a table of size 2^{25}.

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.
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 CPK. Since it is impossible to enumerate all combinations, we further restrict to a subset of the CPK 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}. An example of 4 round path is given in the full version of the paper [10]. We provide also in Table 1 some of 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 the full version of the paper. Paths in bold are new results we found with the method presented in this paper.

b	best differential path probability (differential complexity of the rounds)		
	3 rds	4 rds	5 rds
400	2^{-24} (8 - 8 - 8)	$\mathbf{2^{-84}}$ (16 - 14 - 12 - 42)	2^{-216} (16 - 32 - 40 - 32 - 96)
800	$\mathbf{2^{-32}}$ (4 - 4 - 24)	$\mathbf{2^{-109}}$ (12 - 12 - 12 - 73)	2^{-432} (32 - 64 - 80 - 64 - 192)
1600	$\mathbf{2^{-32}}$ (4 - 4 - 24)	$\mathbf{2^{-142}}$ (12 - 12 - 12 - 106)	$\mathbf{2^{-709}}$ (16 - 16 - 16 - 114 - 547) A better path (2^{-510}) was found independently [20]

4 Simple Distinguishers for Reduced Keccak-*f*

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. 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.

Note that some more involved freedom degree methods (such as message modification [24]) might even allow to also control some of the conditions of the original differential path, thus further reducing the complexity.

4.2 Getting More Rounds

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 [11] 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}).

5 The Rebound Attack on Keccak

The rebound attack is a freedom degrees utilization technique that was first proposed by Mendel et al. in [17] as an analysis of round-reduced Grøstl and Whirlpool. It was then improved in [16, 15, 11, 22] to analyze AES and AES-like permutations and also ARX ciphers [14].

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.

The SuperSbox technique [15, 11] 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 [6].

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.

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

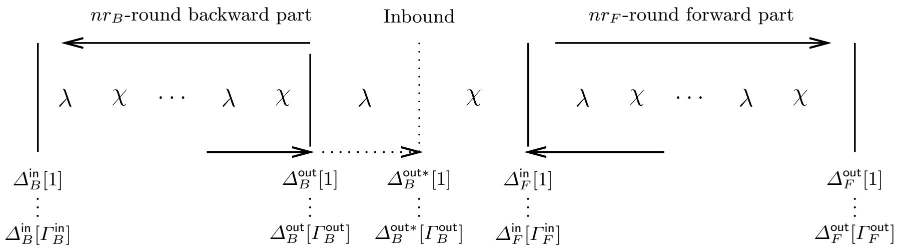
nr_B-round backward part
Inbound
nr_F-round forward part
λ χ $\cdots$ λ χ
λ χ
λ χ $\cdots$ λ χ
$\Delta_B^{\mathsf{in}}[1]$
$\Delta_B^{\mathsf{in}}[\Gamma_B^{\mathsf{in}}]$
$\Delta_B^{\mathsf{out}}[1]$
$\Delta_B^{\mathsf{out}}[\Gamma_B^{\mathsf{out}}]$
$\Delta_B^{\mathsf{out}*}[1]$
$\Delta_B^{\mathsf{out}*}[\Gamma_B^{\mathsf{out}}]$
$\Delta_F^{\mathsf{in}}[1]$
$\Delta_F^{\mathsf{in}}[\Gamma_F^{\mathsf{in}}]$
$\Delta_F^{\mathsf{out}}[1]$
$\Delta_F^{\mathsf{out}}[\Gamma_F^{\mathsf{out}}]$

Fig. 1. Rebound attack on Keccak

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 to 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. We call Γ_B^{mid} (resp. Γ_F^{mid}) the minimum number of differences in the backward (resp. forward) part.

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

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

with

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

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.

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.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*

$$b_{\mathsf{bucket}}(n,s) := \sum_{i=\lceil n/5 \rceil}^{s} (-1)^{s-i} \binom{s}{i} \binom{5i}{n} \quad \text{if} \quad s \leq n \leq 5s \tag{3}$$

and 0 *otherwise.*

The proof of this lemma is given in the full version of the paper.

Using (3), we can derive the probability $p_{\mathsf{bucket}}(n, s)$ that every bucket contains at least one ball when n balls are thrown into s buckets with capacity 5 and the expected number of active buckets when n balls are thrown into s buckets. We can now relate this lemma to the more general pattern-match problem. 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. We analyze this behavior in more details in the full version of the paper.

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 2. *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*

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

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 use from now on only forward and backward paths such that *all Sboxes are active in 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, since this will have an impact on the derivation of p_{match} and N_{match} (this will be handled in the next two sections).

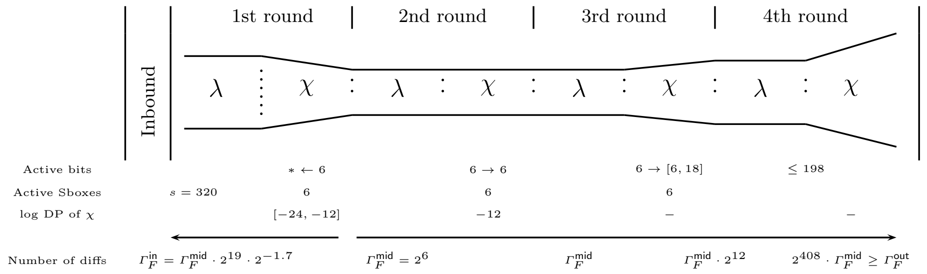
1st round
2nd round
3rd round
4th round
Inbound
λ χ λ χ λ χ λ χ
Active bits
$* \leftarrow 6$
$6 \rightarrow 6$
$6 \rightarrow [6, 18]$
≤ 198
Active Sboxes
$s = 320$
6
6
6
log DP of χ
$[-24, -12]$
-12
–
–
Number of diffs
$\Gamma_F^{\mathsf{in}} = \Gamma_F^{\mathsf{mid}} \cdot 2^{19} \cdot 2^{-1.7}$
$\Gamma_F^{\mathsf{mid}} = 2^6$
Γ_F^{mid}
$\Gamma_F^{\mathsf{mid}} \cdot 2^{12}$
$2^{408} \cdot \Gamma_F^{\mathsf{mid}} \geq \Gamma_F^{\mathsf{out}}$

Fig. 2. The forward trails we are using. The distance between the two lines reflects the number of differences.

The forward paths. For the forward paths set (see Fig. 2), 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 2 first rounds of the 4-round differential path we found (see full version) which have a total success probability 2^{-24} and present 6 active Sboxes during the χ layer of the first round. 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 $2^{-24} \cdot 2^{-2 \cdot 6} = 2^{-36}$ (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.[4] 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 $nr_F = 2$ Keccak-f[1600] rounds, all with DP higher or equal to $p_F = 2^{-36}$. 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 $\Gamma_F^{\mathsf{mid}} = 2^6$. The *total amount of input differences* is $\Gamma_F^{\mathsf{in}} := \Gamma_F^{\mathsf{mid}} \cdot 2^{17.3} = 2^{23.3}$ and we have to generate in total $n_F = \tau_F \cdot \Gamma_F^{\mathsf{in}} = 2^{25}$ forward differential paths. We discuss the amount of output differences in Section 5.8, since we extend there the path with two free additional rounds.

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

[4] 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.

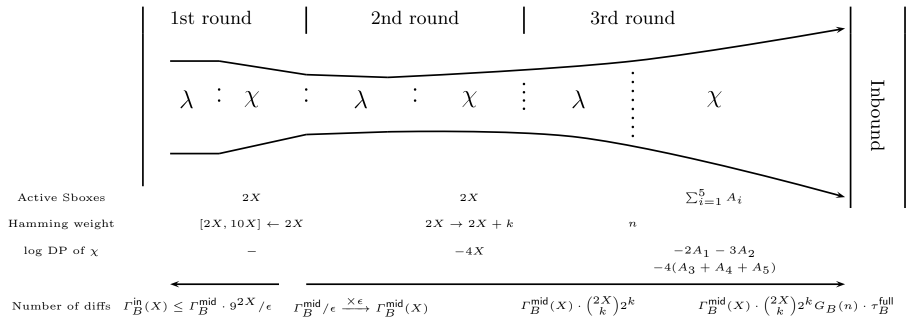
1st round
2nd round
3rd round
Inbound
λ
χ
λ
χ
λ
χ
Active Sboxes
$2X$
$2X$
$\sum_{i=1}^{5} A_i$
Hamming weight
$[2X, 10X] \leftarrow 2X$
$2X \rightarrow 2X + k$
n
log DP of χ
$-$
$-4X$
$-2A_1 - 3A_2$
$-4(A_3 + A_4 + A_5)$
Number of diffs
$\Gamma_B^{\mathsf{in}}(X) \leq \Gamma_B^{\mathsf{mid}} \cdot 9^{2X}/\epsilon$
$\Gamma_B^{\mathsf{mid}}/\epsilon \xrightarrow{\times\epsilon} \Gamma_B^{\mathsf{mid}}(X)$
$\Gamma_B^{\mathsf{mid}}(X) \cdot \binom{2X}{k} 2^k$
$\Gamma_B^{\mathsf{mid}}(X) \cdot \binom{2X}{k} 2^k G_B(n) \cdot \tau_B^{\mathsf{full}}$

Fig. 3. The backward trails we are using. The distance between the two arrows reflects the number of differences.

rebound-like attack. Thus, concerning the backward paths set, we build another type of 2-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 paths 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. This happens with a probability about $\tau_B^{\mathsf{full}} := b_{\mathsf{bucket}}(800, 320)/\binom{5 \cdot 320}{n} = 2^{-15.9}$ 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 $N_{\mathsf{match}} = 2^{486}$ and we previously showed that $p_F = 2^{-36}$. Hence, $p_B^{\min} = 2^{36-486} = 2^{-450}$.

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 $p_{\mathsf{match}} = 2^{-491.47}$, that $\lceil \frac{1}{p_F \cdot p_B \cdot N_{\mathsf{match}}} \rceil = 1$ and that $\tau_B = 2^{-15.9}$. We also know that $\Gamma_F^{\mathsf{in}} = 2^{23.3}$. Hence, $n_B^{\min} = 2^{491.47+15.9-23.3} = 2^{484.07}$.

We show now how we generated these paths. Fig. 3 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. Note also that all active columns are in the CPK 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

$$\epsilon \coloneqq p_{\mathsf{dist}}(2X, 0, 0, 0, 0) \ , \tag{5}$$

where p_{dist} is given by Lemma 2.[5] 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 $\Gamma_B^{\mathsf{mid}}(X) \coloneqq \binom{5}{2}^X \cdot \binom{s}{X} \cdot \epsilon$.

Looking at the DDT from χ, 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 $n \approx u - (u \cdot (u - 1))/1600$ 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

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

and will not collide. Hence, we verify the equation for n. 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 2.

Lemma 3. *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*

$$G_B(n) \geq \sum_{A_5=0}^{\lfloor \frac{n}{5} \rfloor} \sum_{A_4=0}^{N_4} \sum_{A_3=0}^{N_3} \sum_{A_2=0}^{N_2} F(X, A_1, \dots, A_5) \cdot 2^{2A_1+3A_2+3.58A_3+4(A_4+A_5)} , \quad (6)$$

where $N_4 := \lfloor (n - 5A_5)/4 \rfloor$, $N_3 := \lfloor (n - 5A_5 - 4A_4)/3 \rfloor$,
$N_2 := \lfloor (n - 5A_5 - 4A_4 - 3A_3)/2 \rfloor$, $A_1 := n - 5A_5 - 4A_4 - 3A_3 - 2A_2$, and

$$F(X, A_1, \dots, A_5) := \begin{cases} p_{\mathsf{dist}}(A_1, \dots, A_5) & \text{if } 2^{-4X-2A_1-3A_2-4(A_3+A_4+A_5)} \geq p_B^{\min} \\ 0 & \text{else.} \end{cases} \quad (7)$$

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

The proof is given is the full version of the paper.

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

$$\tau_B^{\mathsf{DP}} := \sum_{A_5=0}^{\lfloor n/5 \rfloor} \sum_{A_4=0}^{\lfloor (n-5A_5)/4 \rfloor} \sum_{A_3=0}^{\lfloor (n-5A_5-4A_4)/3 \rfloor} \sum_{A_2=0}^{\lfloor (n-5A_5-4A_4-3A_3)/2 \rfloor} F(X, A_1, A_2, A_3, A_4, A_5) \quad (8)$$

with $F(\dots)$ defined in (7) 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

$$\varGamma_B^{\mathsf{out}} = \epsilon \cdot \binom{5}{2}^X \cdot \binom{s}{X} \cdot \binom{2X}{k} \cdot 2^k \cdot G_B(n) \cdot \tau_B^{\mathsf{full}} \quad (9)$$

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

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.

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**. Note that *this effect is only strong regarding the input of the Sbox*, 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 $\sum_{i=1}^{5} c_i = s$ since we forced that all Sboxes are active during a match. Moreover, for a Hamming weight value w, we have $\sum_{i=1}^{5} i \cdot c_i = w$. The set of divisions c_i verifying the above mentioned equations is denoted by C_w. The number of possible $5s$-bit vectors satisfying $(c_1, \ldots, c_5)$ (i.e., c_1 Sboxes with 1 active bit, c_2 with 2 etc.) is denoted $b_c(c_1, \ldots, c_5)$ and

$$b_c(c_1, \ldots, c_5) = \frac{s!}{c_1! c_2! \ldots c_5!} \cdot 5^{c_1 + c_4} \cdot 10^{c_2 + c_3} \; . \tag{10}$$

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*

$$\sum_{w=s}^{5s} \Pr[\mathsf{Hw}_{\mathsf{tot}} = w | \mathsf{full}] \sum_{(c_1, \ldots, c_5) \in C_w} \frac{b_c(c_1, \ldots, c_5)}{b_{\mathsf{bucket}}(w, s)} \prod_{i=1}^{5} \left(\sum_{y \in \{0,1\}^5} \sum_{\substack{v \in \{0,1\}^5 : \\ \mathsf{Hw}(v) = i}} \frac{P_{\mathsf{out}}(y) \mathbb{1}_{\mathsf{DDT}[v][y]}}{\binom{5}{i}} \right)^{c_i} \tag{11}$$

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, $\Pr[\mathsf{Hw}_{\mathsf{tot}} = w|\mathsf{full}]$ *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 (10), $b_{\mathsf{bucket}}(w,s)$ 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.*[6]

We leave the proof to the complete version of the paper. However, we define the following intermediate result: $p_{\mathsf{match}}(w) := \Pr[\mathsf{match}|\mathsf{Hw}_{\mathsf{tot}} = w, \mathsf{full}]$ which can be written as

$$\sum_{(c_1,\dots,c_5)\in C_w} \Pr[\mathsf{match}|(c_1,\dots,c_5), \mathsf{Hw}_{\mathsf{tot}} = w, \mathsf{full}] \cdot \Pr[(c_1,\dots,c_5)|\mathsf{Hw}_{\mathsf{tot}} = w, \mathsf{full}] \ . \quad (12)$$

The measured distributions along with some intermediate values will be given in the extended version of the paper.

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 $N_w := \mathbb{E}[N|\mathsf{match}, \mathsf{Hw}_{\mathsf{tot}} = w, \mathsf{full}]$ values from a match, with*

$$N_w = \frac{1}{p_{\mathsf{match}}(w)} \sum_{(c_1,\dots,c_5)\in C_w} \prod_{i=1}^{5} Z^{c_i} \cdot \frac{b_c(c_1,\dots,c_5)}{b_{\mathsf{bucket}}(w,s)} \ , \quad (13)$$

[6] Note that $\Pr[\mathsf{Hw}_{\mathsf{tot}} = w|\mathsf{full}]$ 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.

with

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

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* (12)*,* $b_c(\dots)$ *is given by* (10)*,* $b_{\mathsf{bucket}}(w, s)$ *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.*

The proof is given in the complete version of the paper.

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

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

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. 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 look then for the maximum Hamming weight $w_{\max}$ we can afford, i.e., 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 need to update (11). 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}) will be given in the extended version of the paper.

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, $\Pr[\mathsf{Hw}_{\mathsf{tot}} = w|\mathsf{full}]$. 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 $\Pr[\mathsf{Hw}_{\mathsf{tot}} = w|\mathsf{full}]$ by taking some random backward paths with limited complexity. Using the worst case cost of these paths, we were able to select $w_{\max}$ 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 $\Pr[\mathsf{Hw}_{\mathsf{tot}} = w|\mathsf{full}]$ 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}.

When $(X,k) = (8,8)$, we have $\epsilon = 0.736$ and $\Gamma_B^{\mathsf{mid}} = \binom{5}{2}^X \cdot \binom{s}{X} \cdot \epsilon = 2^{77.26}$. If we filter all paths that have a DP smaller than 2^{-450}, i.e., we set $p_B = 2^{-450}$, we get for $(X,k) = (8,8)$ at least $\epsilon \cdot \binom{5}{2}^X \cdot \binom{s}{X} \cdot \binom{2X}{k} \cdot 2^k \cdot G_B(n) \cdot \tau_B^{\mathsf{full}} = 2^{493.88-15.9} = 2^{477.98}$ distinct differences using (9) 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 $\tau_B^{\mathsf{DP}} \approx 1 - 10^{-10}$ of the paths, following (8), i.e., we have almost no filtering on the DP. Hence, we filter the backward paths with a ratio $\tau_B = \tau_B^{\mathsf{full}} \cdot \tau_B^{\mathsf{DP}} \approx 2^{-15.9} \cdot (1 - 10^{-10}) = 2^{-15.9}$. We have also $p_B = 2^{-450}$ and $p_F = 2^{-36}$. Therefore, we need $N_{\mathsf{match}} \geq 2^{486}$. Computations show that we have to set $w_{\max} = 1000$. This leads to $p_{\mathsf{match}} = 2^{-491.47}$. This implies that the minimum total number of backward paths we need to generate is $n_B^{\min} = 1/(p_{\mathsf{match}} \cdot \Gamma_F^{\mathsf{in}} \cdot \tau_B) = 1/(p_{\mathsf{match}} \cdot \Gamma_F^{\mathsf{in}} \cdot \tau_B^{\mathsf{full}}) = 2^{484.07}$. All these paths apply on $nr_B = 2$ Keccak-f[1600] rounds, all with DP higher or equal to $p_B^{\min} = 2^{36-486} = 2^{-450}$.

To summarize, we have that the number of backward output differences is $\Gamma_B^{\mathsf{out}} = n_B^{\min} \cdot \tau_B = 2^{484.07-15.9} = 2^{468.17}$ and that the number of forward input differences is $\Gamma_F^{\mathsf{in}} = 2^{23.3}$. 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 $1/(p_B \cdot p_F) \leq 2^{450+48} = 2^{486}$ which is lower or 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 $C = 2^{491.47}$.

This model was verified on the Keccak-f[100] internal permutation. By applying this attack on it, we found a 4-round result together with solution pairs. This gives a 6-round distinguisher with complexity $2^{28.76}$.

5.8 The distinguisher

We will use exactly the same type of limited-birthday distinguishers as in Section 4.

Relaxing the Forward Paths. We analyze now the impact of this two additional paths 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 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 $\Gamma_F^{\mathsf{out}} \leq \Gamma_F^{\mathsf{mid}} \cdot 2^{12} \cdot 4^{198} = 2^{18} \cdot 2^{396} = 2^{414}$.

Relaxing the Backward Paths. 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}.

We have $\Gamma_B^{\mathsf{in}} \leq \Gamma_B^{\mathsf{mid}}(8) \cdot 9^{2 \cdot 8}/\epsilon = 2^{77.7+50.7} = 2^{128.4}$ and $\Gamma_F^{\mathsf{out}} \leq 2^{414}$. The generic complexity of the distinguisher is, hence, greater than $2^{1057.6}$. This is much greater than the complexity of the rebound attack $C = 2^{491.47}$.

6 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-$f[1600]$.

b	best differential distinguishers complexity							
	1 rd	2 rds	3 rds	4 rds	5 rds	6 rds	7 rds	8 rds
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}	$2^{491.47}$

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 CPK considerations and better χ mapping. This new method provides some of the best 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 $2^{491.47}$. 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. Using the techniques presented in [19] could help reducing the complexity of it.

Acknowledgements. The authors would like to thank the anonymous referees for their helpful comments. We are extremely grateful to Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche for their remarks on the first drafts of this paper. Finally, 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. On alignment in Keccak. ECRYPT II Hash Workshop, 2011.
7. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The KECCAK Reference. Submission to NIST (Round 3), 2011.

8. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The KECCAK SHA-3 Submission. Submission to NIST (Round 3), 2011.
9. Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-Order Differential Properties of Keccak and *Luffa*. In Antoine Joux, editor, *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 252–269. Springer, 2011.
10. Alexandre Duc, Jian Guo, Thomas Peyrin, and Lei Wei. Unaligned Rebound Attack - Application to Keccak. Cryptology ePrint Archive, Report 2011/420, 2011. `http://eprint.iacr.org/`.
11. Henri Gilbert and Thomas Peyrin. Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In Hong and Iwata [12], pages 365–383.
12. 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.
13. Dmitry Khovratovich, María Naya-Plasencia, Andrea Röck, and Martin Schläffer. Cryptanalysis of *Luffa* 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.
14. Dmitry Khovratovich, Ivica Nikolic, and Christian Rechberger. Rotational Rebound Attacks on Reduced Skein. In Abe [1], pages 1–19.
15. 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.
16. 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.
17. 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.
18. 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.
19. María Naya-Plasencia. How to Improve Rebound Attacks. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 188–205. Springer, 2011.
20. María Naya-Plasencia, Andrea Röck, and Willi Meier. Practical analysis of reduced-round keccak. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *INDOCRYPT*, volume 7107 of *Lecture Notes in Computer Science*, pages 236–254. Springer, 2011.
21. Vincent Rijmen, Deniz Toz, and Kerem Varici. Rebound Attack on Reduced-Round Versions of JH. In Hong and Iwata [12], pages 286–303.
22. 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.
23. Keccak team. Keccak Crunchy Crypto Collision and Pre-image Contest, 2011. See `http://keccak.noekeon.org/crunchy_contest.html`.
24. 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.