On alignment in Keccak

Guido Bertoni[1], Joan Daemen[1], Michaël Peeters[2], and Gilles Van Assche[1]

[1] STMicroelectronics
[2] NXP Semiconductors

Abstract. In this paper we investigate the ability to predict the propagation of truncated differences and linear masks in cryptographic primitives. We speak of strong alignment if this propagation is predictable and of weak alignment if the propagation is hard to predict. We show the relevance of alignment with respect to some types of cryptanalysis including the rebound attack. We give insight in the alignment in the NIST SHA-3 hash function contest finalist Keccak by reporting on a number of experiments we conducted. It appears that the propagation of differences or linear masks does not respect the row boundaries, hence Keccak has weak alignment.

Keywords: Keccak, alignment, truncated differentials, rebound attack

1 Introduction

In this paper we study a property of cryptographic round functions that plays an important role in its sensitivity to truncated differential cryptanalysis and rebound attacks: *alignment*. More particularly, we investigate alignment for the NIST SHA-3 hash function contest finalist Keccak [2].

Truncated differential cryptanalysis (DC) was introduced in [11] as a powerful variant of classical DC for attacking block ciphers. Techniques from truncated DC play an important role in rebound attacks, a powerful method for attacking hash function components [13]. In truncated DC one divides the function input, output and intermediate computation values in sub-blocks, typically of equal size (e.g. bytes). Whereas in classical DC one studies propagation of differences that are fully specified, in truncated DC one only specifies whether a sub-block is active (has a difference) or is passive (has no difference). In case of sub-blocks of n bits, the range of possible values of an active sub-block of a difference covers all $2^n - 1$ non-zero sub-block values. A truncated difference is fully determined by the *activity pattern* indicating for each sub-block whether it is active or not.

We first introduce alignment by contrasting activity pattern propagation in Keccak with that in Rijndael(AES) [4,16]. Both designs attempt to avoid narrow trails, but their approach in terms of alignment differ significantly. This is followed by a discussion on the relevance of alignment for cryptanalysis. Finally, we report on our investigation on alignment in Keccak. The outcome of our experiments suggest that the round function of Keccak-f has in general weak alignment. For the cases where there is no weak alignment, this is due to saturation where the vast majority of differences/masks propagate to activity patterns that cover the complete state.

2 Alignment in Rijndael

We assume that most readers are aware of the general structure of Rijndael and refer those that are not to [4,16] for a description. The round function of Rijndael consists of four steps. Each of these steps treats a *state* as an array of bytes. Therefore it makes sense to take bytes for the sub-blocks. Activity patterns have the same shape as a Rijndael state, where for each byte position the (binary) activity is specified.

Consider an activity pattern with a single active byte at the input of a round and see how it propagates through the four steps of a Rijndael round: `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey`. The step `SubBytes` applies an invertible S-box to the state bytes separately and hence leaves the activity pattern invariant. The byte transpositions `ShiftRows` moves the active byte to another position in the same row (if in row 0, it leaves its position intact). The mixing layer `MixColumns` operates on the columns independently and thanks to the fact that it has branch number 5, converts the column with a single active byte to a column with 4 active bytes. Finally, the step `AddRoundKey` leaves the activity pattern invariant. So an activity pattern with a single active byte at the input of the round function fully determines the activity pattern at the output of the round function. Similarly, one can show that any difference at the output of the round function with a single active byte fully determines the activity pattern of the difference at the input.

We express the fact that this difference propagation nicely follows the byte boundaries by saying that the Rijndael round function has *strong alignment* with respect to bytes.

Four rounds of AES (Rijndael restricted to a block size of 16 bytes) can be represented as a first layer of four 32-bit super S-boxes [5], a diffusion mapping and a second layer of super S-boxes. For this representation one can take as sub-blocks 32-bit blocks and again a single active block at the input implies that all four blocks are active at the output, and vice versa.

In fact, these strong alignment properties are a direct consequence of the very design choices that have enabled the crystal clear proofs on lower bounds for the weight of four-round trails (both differential and linear) in Rijndael [4].

The examples given represent truncated differentials with probability 1, i.e., that are deterministic. One can associate a differential probability (DP) with a truncated differential (a, b) over a function. This is simply for all input pairs that have an activity pattern a, the proportion of output pairs with activity pattern b. Similarly, one can associate a DP with a truncated differential trail. The weight of a truncated differential is minus the binary log of its DP. The weight of a truncated differential trail is the sum of the weights of its single-round differentials.

In the steps of the Rijndael round function, a truncated differential (a, b) over `SubBytes` has DP equal to 1 if $a = b$ and 0 otherwise. The same goes for the round key addition `AddRoundKey`. For the byte transposition `ShiftRows` we have a DP equal to 1 if $b = \texttt{ShiftRows}(a)$ and 0 otherwise. So for these three steps, activity pattern propagation is deterministic. The mixing layer `MixColumns` is an exception to this and has propagation probabilities ranging from 1 for an input with at most one active byte per column down to 255^{-n} with n the Rijndael block size in bytes, for an input with all bytes active and an output with exactly one active byte per column. Moreover, the DP of a truncated differential depends on the *direction*. For example, a differential from a single active byte to four active bytes has DP equal to 1 in the forward direction. In the backward direction, it has DP 255^{-3}. This is linked to the fact that the number of pairs with four active bytes is 255^3 times larger than the number of pairs with a single active byte.

As opposed to single-round classical differentials, the DP is not determined by the nonlinear step `SubBytes` but rather by the mixing layer `MixColumns`. In truncated single-round differentials the nonlinear step `SubBytes` merely defines the state partitioning in sub-block and the activity propagation probabilities are fully determined by the linear layer $\texttt{AddRoundKey} \circ \texttt{MixColumns} \circ \texttt{ShiftRows}$.

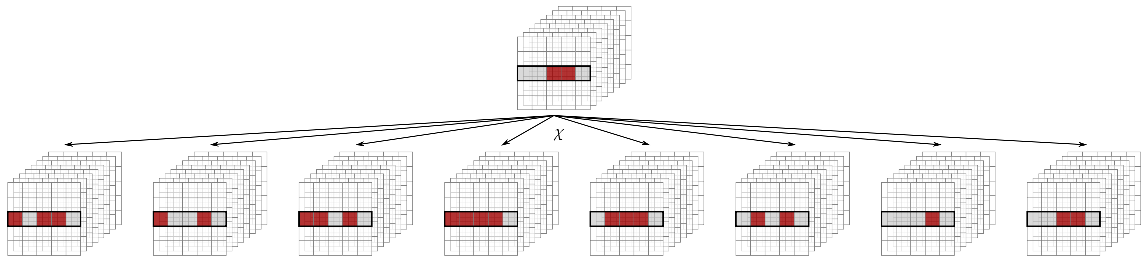

Fig. 1. Propagation of a difference in a row of Keccak-f[200] at the input of χ. The set of possible output differences is an affine space. The row-activity pattern is preserved by χ. The state is represented with bits as squares, with positions x along the horizontal axis, from $x = -2$ (left) to $x = 2$ (right), with positions y along the vertical axis, from $x = -2$ (bottom) to $x = 2$ (top), and with positions z in projective perspective, from $z = 0$ (foreground) to $z = 2^\ell - 1$ (background). A red square means an active bit.

3 Alignment in Keccak

3.1 Short description of Keccak

We give here a short description of Keccak, relevant in the context of this paper. For more details and background we refer the reader to [2]. Keccak is a *sponge function family* [2] that makes use of a set of underlying b-bit permutations called Keccak-f. There are in total 7 such permutations with different widths b: 25, 50, 100, 200, 400, 800 and 1600. Each of the Keccak-f permutations is iterated: it consists of the iterated application of a simple round function. The only difference between the rounds is the application of a round constant.

The round function operates on a state in the form of a three-dimensional bit array with dimensions $5 \times 5 \times 2^\ell$. It can be seen as 2^ℓ *slices* of each 5 *rows* and 5 *columns* or as a two-dimensional array of 5×5 *lanes* of each 2^ℓ bits. A bit position can be specified by three coordinates x, y and z where the former two range from 0 to 4 and the latter from 0 to $2^\ell - 1$. The position of a row is determined by two coordinates y and z, that of a column by x and z and that of a lane by x and y. The position of a slice is determined by the z-coordinate only.

The round function of Keccak-f consists of 5 invertible steps:

θ Linear mixing layer that adds to each bit of the state in position (x, y, z) the parity of bits in columns $(x, z + 1)$ and $(x - 1, z - 1)$.

ρ Bit transposition that shifts the bits in each lane along the z-axis with a particular offset $c_\rho(x, y)$ that depends on the lane position: it shifts bit in position (x, y, z) to position $(x, y, z + c_\rho(x, y))$.

π Bit transposition that moves the lanes around within the 5×5 array: it moves bit in position (x, y, z) to position (X, Y, z) with $(X, Y)^\mathrm{T} = \mathrm{M}(x, y)^\mathrm{T}$ with M a fixed 2×2 matrix.

χ Nonlinear layer that operates on the rows separately. It complements bit in position (x, y, z) if the bits in positions $(x + 1, y, z)$ and $(x + 2, y, z)$ exhibit the pattern 01.

ι Addition of round constant to the lane in position $(0, 0)$.

In this description x, y and z indices must be taken modulo 5, 5 and 2^ℓ respectively, and addition and multiplication of bits is in GF(2). With the exception of ι, all steps are translation-invariant in the direction of the z-axis.

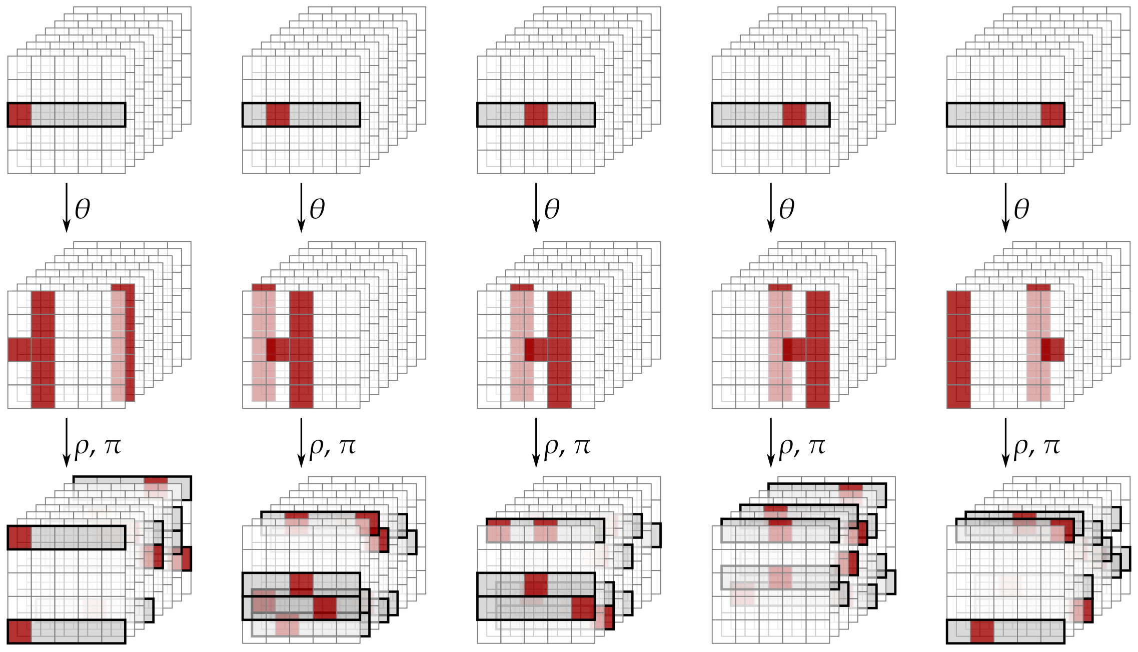

Fig. 2. On top, from left to right, the basis elements e_i of differences in row $(y, z) = (0, 0)$ of Keccak-$f[200]$; in the middle, the effect of θ; and at the bottom, the corresponding values $L(e_i)$. The active rows are grayed.

3.2 Alignment along the row boundaries

The nonlinear step χ can be seen as the parallel application of 5-bit S-boxes on the rows of the state. Therefore it seems natural to consider the rows as sub-blocks. A row-activity pattern consists of $5 \times 2^\ell$ bits. The steps χ and ι leave a row-activity patterns invariant as they operate on the rows independently and are invertible (see also Figure 1). So we just have to study the effect of θ, ρ and π on the activity patterns. Note that these are all three linear functions and two of them are mere bit transpositions. We denote $\pi \circ \rho \circ \theta$ by L: the effect of a round on an activity pattern is that of L.

Consider now an activity pattern at the input of a round with a single active row in position $(y, z) = (0, 0)$ and see to which activity patterns this may propagate at the output. This activity pattern covers a set of 31 difference patterns. In the remainder of this section we will denote this set by A and use A^* for A augmented with the all-zero pattern. The set A^* can be described as a vector space over GF(2) with a basis $\langle e_i \rangle$ of 5 elements. The elements of A^* can then be expressed as $\sum_i a_i e_i$ with $a_i \in \text{GF}(2)$. In the simplest possible basis, e_i is the vector with the bit in position $(i, 0, 0)$ equal to 1 and all other bits equal to 0.

Thanks to its linearity, we can now describe the effect of L on A^* as its effect on the basis elements e_i. Applying θ to a base element consists of setting 10 bits in two additional columns to active, leading to a state with 11 active bits in total. The step ρ shifts these 11 active bits to different slices and finally π moves these active bits around within those slices. This is illustrated in Figure 2.

For large values of 2^ℓ we expect the active bits of $L(e_i)$ to be in 11 different rows. Additionally, as there are many rows, we expect there to be little overlap of active rows between these base vectors. It is therefore likely that the elements $L(e_i)$ generate a space where no two elements have the same activity pattern. In other words, the elements of A^* would lead to 32 different activity patterns. Omitting the zero vector gives 31.

If $\ell = 0$ there is only a single slice and ρ is the identity function. Due to the fact that after θ all 5 rows in this single slice are active and that π is a mere bit transposition, we expect most elements of $L(A)$ to have no passive rows. This implies that the 31 nonzero elements of $L(A)$ would have the same activity pattern.

For values of 2^ℓ in between it is harder to predict what will happen, so we determined the distributions of $L(A)$ experimentally. Thanks to the fact that θ, ρ and π are translation-invariant in the direction of the z-axis [2], the results for a row in position $(y, 0)$ are also valid for all rows in position (y, z) with $z \neq 0$. We list the number of possible activity patterns at the output in Table 1.

lane size 2^ℓ	$y = 0$	$y = 1$	$y = 2$	$y = 3$	$y = 4$
1	1	1	1	1	1
2	11	10	11	9	12
4	26	28	29	27	28
8	31	31	31	31	31
16	31	31	31	31	31
32	31	31	31	31	31
64	31	31	31	31	31

Table 1. Number of possible row activity patterns at output of a Keccak-f round for input differences with a single active row

Clearly, for width values above 4, the 31 input differences with a single active row lead to 31 different output activity patterns. As opposed to Rijndael, the difference propagation does not respect the row boundaries and we speak of *weak alignment*. For the widths 1 and 2, we observe a saturation effect due to the high average diffusion of θ and the limited number of rows in the state. It turns out that L maps most input differences with a single active row to differences with all rows active.

4 The relevance of alignment

In this section we explain how alignment plays an important role in the sensitivity of a round function to rebound attacks [13], the grouping of multiple trails in differentials and correlations [5] and the existence of plateau-like trails [6].

4.1 Rebound attacks

In this section we try to capture the general idea of rebound attacks relevant in the context of this paper. We do not have the ambition to provide its full bibliography and limit our citations to some interesting papers to illustrate our points and refer the interested reader to the ECRYPT II SHA-3 Zoo [7] for an excellent source of references.

Rebound attacks are a technique of finding pairs that follow truncated differential trails (also called *right pairs of characteristics*) that typically have few active bits at the input and at the output. It applies to use cases where the attacker has full knowledge of, and partial control over, the input to the function. This function is typically a compression function of a hash function, an iterated permutation or a block cipher. Rebound attacks are often used as certificational distinguishers for compression functions and block ciphers [13,12,8,15,10]. However, in the case of hash functions they are sometimes used to construct actual attacks [14,9].

In the simplest version of the rebound attack, pairs are constructed in two phases. In a first phase, called *inbound*, available degrees of freedom are exploited to efficiently construct candidates that satisfy conditions imposed by the middle part of the truncated trail, that is typically heavy, i.e., imposes many conditions. In a second phase, called *outbound* phase, candidate pairs are computed forwards and backwards until a pair is found that follows the full truncated trail. Many variants and optimizations have been published in recent years, such as rebound attacks that combine multiple inbound phases.

To the best of our knowledge, all published rebound attacks making use of truncated differential trails are on cryptographic primitives with strong alignment. Cryptographic primitives with strong alignment tend to have strong upper bounds for the DP of differential trails. For example in Rijndael there are no four-round differential trails with probability above 2^{-150}. It follows that for a fixed key, the expected number of pairs following such a trail is 2^{-23}. However, due to the strong alignment, many such trails cluster in truncated differentials followed by many pairs. When there is weak alignment, differential trails tend to diverge and it is highly unlikely that exploitable truncated differentials can be constructed.

4.2 Clustering of differential trails

Alignment plays a role in the way differential trails cluster in differentials. The differential probability (DP) of differentials over multiple rounds of a block cipher and iterated permutations is often (silently) approximated by that of single differential trails. If differential trails tend to cluster in differentials, this may result in a strong underestimation, hence a reliable estimation of the DP of differentials should take into account this clustering effect. For a high-level description of differentials, trails and their DP in iterated permutations we refer the interested reader to [1, Section "Differential Cryptanalysis"].

Consider the case of two-round trails and differentials. A two-round trail is specified by the differences b'_i at the input of three consecutive rounds: (b'_0, b'_1, b'_2). A differential is specified by the initial difference b'_0 and a final difference b'_2 only. The DP of the differential (b'_0, b'_2) is the sum of the differential probabilities of all matching differential trails (b'_0, b'_1, b'_2):

$$\mathrm{DP}(b'_0, b'_2) = \sum_{b'_1} \mathrm{DP}(b'_0, b'_1, b'_2) \ .$$

Clustering of trails occurs if for a given differential (b'_0, b'_2) when there are many values of b'_1 with $\mathrm{DP}(b'_0, b'_1, b'_2) > 0$. For $\mathrm{DP}(b'_0, b'_1, b'_2)$ to be non-zero, both single-round differentials (b'_0, b'_1) and (b'_1, b'_2) must have a non-zero DP. A Keccak-f round is composed of a linear step L followed by a nonlinear step χ. If we denote by a'_i the differences at the input of χ of round i, we have $a'_{i+1} = L(b'_i)$:

$$b'_0 \overset{L}{\Rightarrow} a'_1 \overset{\chi}{\Rightarrow} b'_1 \overset{L}{\Rightarrow} a'_2 \overset{\chi}{\Rightarrow} b'_2 \qquad (1)$$

We will denote the row activity pattern of a difference a' by $\mathrm{act}(a')$. For the differential (b'_0, b'_1) this requires $\mathrm{act}(b'_1) = \mathrm{act}(a'_1)$. For the differential (b'_1, b'_2) this requires $\mathrm{act}(L(b'_1)) = \mathrm{act}(a'_2)$. Hence, for all contributing trails, b'_1 must satisfy both requirements. If for the differences b'_1 with activity pattern $\mathrm{act}(a'_1)$ the activity patterns $\mathrm{act}(L(b'_1))$ are all different, the two-round differential (b'_0, b'_2) has at most a single differential trail with non-zero DP. It follows that weak alignment prevents the clustering of differential trails in two-round differentials.

Strong alignment on the other hand does not necessarily lead to massive clustering of differential trails in two-round differentials. Here the properties of the S-boxes also play a role and investigating it requires dedicated analysis. This was done for two-round differentials in Rijndael[5].

4.3 Clustering of linear trails

Similarly, alignment plays a role in the way linear trails cluster in correlations. Correlations over multiple rounds of a block cipher or an iterated permutation are often (silently) approximated by that of single linear trails. This is illustrated by the usage of the term *linear approximation* for both concepts in many papers. The truth is that the value of a multi-round correlation in an iterated permutation is the sum of the correlation contributions of all compatible linear trails [3]. These correlation contributions have a sign and when they are added the value of the correlation may indeed be smaller than that of the trails contributing to it. If linear trails tend to cluster in multi-round correlations, some correlations will be higher than expected and some will be lower. Thus the correlation values may diverge significantly from the ones predicted by considering single linear trails and a reliable estimation of the multi-round correlations should therefore take into account this clustering effect. For a high-level description of correlations and linear trails in iterated permutations we refer the interested reader to [1, Section "Linear Cryptanalysis"].

A two-round trail is specified by the masks v_i at the output of three consecutive rounds: (v_0, v_1, v_2). A two-round correlation is specified by an output mask v_0 and an input mask v_2 only. The correlation of (v_0, v_2) is the sum of the (signed) correlation contributions of all matching linear trails (v_0, v_1, v_2):

$$C(v_0, v_2) = \sum_{v_1} C(v_0, v_1, v_2) \; .$$

Clustering of trails occurs if for a given correlation (v_0, v_2) there are many values of v_1 with $C(v_0, v_1, v_2) \neq 0$. For $C(v_0, v_1, v_2)$ to be non-zero, both single-round correlations $C(v_0, v_1)$ and $C(v_1, v_2)$ must be non-zero. If we denote by u_i the mask at the input of χ of round i, the trail can be depicted as follows:

$$v_2 \overset{L}{\Rightarrow} u_1 \overset{\chi}{\Rightarrow} v_1 \overset{L}{\Rightarrow} u_0 \overset{\chi}{\Rightarrow} v_0 \tag{2}$$

we have $v_{i+1} = L^{\mathrm{T}}(u_i)$, with the T suffix denoting the transpose, or equivalently $u_i = L^{-\mathrm{T}}(v_{i+1})$ with the $-$T suffix denoting the inverse of the transpose. This follows directly from the expression of parities using transposes (denoting L as a matrix multiplication):

$$\langle u, a \rangle = u^{\mathrm{T}} a = u^{\mathrm{T}} L L^{-1} a = (L^{\mathrm{T}} u)^{\mathrm{T}} L^{-1} a = \langle L^{\mathrm{T}} u, L^{-1} a \rangle.$$

For the correlation $C(v_0, v_1)$ this results in the requirement $\mathrm{act}(L^{-\mathrm{T}}(v_1)) = \mathrm{act}(u_0)$. For the correlation (v_1, v_2) this results in the requirement $\mathrm{act}(v_1) = \mathrm{act}(u_1)$. Hence, for all contributing trails, v_1 must satisfy both requirements. If for the masks u_0 with activity pattern $\mathrm{act}(v_0)$, the activity patterns $\mathrm{act}(L^{\mathrm{T}}(u_0))$ are all different, the two-round correlation $C(v_0, v_2)$ has at most a single linear trail with non-zero correlation. It follows that weak alignment prevents the clustering of linear trails in two-round correlations.

4.4 Plateau trails

A differential over an S-box imposes conditions on the bits of members of pairs that satisfy it. Each condition consists of an equation over GF(2) that the bits must satisfy. The pairs that follow a trail satisfy the conditions imposed by all its active S-boxes. Consider now a two-round trail (b'_0, b'_1, b'_2). The active S-boxes in the differential (a'_1, b'_1) over the first non-linear layer impose conditions on b_1 and the active S-boxes in the differential (a'_2, b'_2) impose conditions on a_2. The equations in the bits of a_2 can simply be *propagated* to b_1 by substitution using $a_2 = L(b_1) + p$ where p is a round key or round constants, and a_2 (resp. b_1) is the absolute value corresponding to a'_2 (resp. b'_1).

Assume now the equations in a_2 are affine and can be expressed as $u^{\mathrm{T}} a_2 = c$. If we express the linear step L as the multiplication by a matrix M, the substitution looks like this: $u^{\mathrm{T}}(Mb_1 + p) = c$. If we denote $M^{\mathrm{T}}u$ by v and $Mp + c$ by d, this gives $v^{\mathrm{T}} b_1 = d$. Clearly, u and v are linear masks and the propagation of u to v is governed by L^{T}. In the presence of strong alignment, masks u in a single S-box propagate to masks v with the same activity pattern. This may lead to the situation that the number of equations becomes larger than the number of bit positions occurring in the masks, leading to an overdetermined set of linear equations. For certain values of the round key (or round constant) it will have solutions and for others it will not. In the latter case, the number of pairs following the trail (or equivalently, its DP) is larger than 2^w with w the weight of the trail. Moreover, if the equations are affine, the number of solutions is a power of two. This effect was called *plateau trails* and investigated for the case of Rijndael in [6].

For plateau trails, the average of the DP over all round key values is 2^{-w} with w the weight of the trail. However, the DP is 2^{w-n} for an affine subspace of the round key space containing a fraction 2^{-n} of the keys and 0 for keys outside that subspace. It turns out that in Rijndael the vast majority of trails are plateau trails with $n > 0$. For those trails, 2^{-w} is not a reliable estimate for the DP for fixed keys.

An iterated permutation such as Keccak-f has no round keys and a hypothetical plateau trail would have either no pairs or more pairs than suggested by 2^{-w}. When considering trails with weight above the width of the permutation, clearly 2^{-w} cannot be correct as the DP must be an integer multiple of 2^{1-b} due to the fact that the total number of pairs is 2^{b-1}. Trails with only a few pairs appear to be of little use to the cryptanalyst. However, when the plateau trail effect occurs for low-weight trails, this may be exploitable due to the fact that finding pairs following the trail would consume less degrees of freedom than its weight w suggests.

Let us now take a look at two-round trails in Keccak-f. Each equation (not necessarily linear) due to the active S-boxes in the first nonlinear layer involve bits of a single row of b_1 only. Each equation due to the active S-boxes in the second nonlinear layer can be expressed by a linear mask that is active in a single row. When L^{T} has weak alignment, the propagation of the equation masks from a_2 to b_1 results in masks that have distinct row activity patterns. If all these masks have at least two active rows, it is very unlikely that the combined set of equations is overdetermined thanks to the mere number of bits involved. It follows that the row-alignment of L^{T} plays an important role in the occurrence of plateau-like trails.

5 A search optimization technique

When investigating these distributions, we noticed an interesting property of bases and used it to optimize our programs. This property is not specific for Keccak and can be applied for propagation of vector spaces through any linear function L and for any sub-block partitioning.

Lemma 1. *Let $\langle e_i \rangle$ be a basis for a set A^* containing an element e_j for which there exists a sub-block position in which $L(e_j)$ is active and $\forall i \neq j$ $L(e_i)$ is passive. Then for any pair of elements $a = \sum_i a_i e_i$ and $b = \sum_i b_i e_i$ with $a_j \neq b_j$, $L(a)$ and $L(b)$ have different sub-block activity patterns.*

Proof. Say $a_j = 1$ and $b_j = 0$. Then $L(a)$ has an active sub-block at the position where among the base vectors only e_j is active and $L(b)$ has a passive sub-block at that position. It follows that $L(a)$ and $L(b)$ have different slice activity patterns. □

We call e_j an activity-splitting base element for a given L and type of sub-blocks. It partitions the elements of $L(A^*)$ in two equally-sized subsets with non-overlapping activity patterns. If for a given vector space A^* a basis can be constructed with n activity-splitting elements, then the elements of $L(A^*)$ are distributed over at least 2^n different activity patterns. If n is the dimension of A^*, all activity patterns are different.

A technique for constructing activity-splitting base elements from an existing base is the following. Let E_a be the set of base elements for which $L(e_i)$ has an active sub-block in a given position. If for all base elements in E_a the value of $L(e_i)$ restricted to this sub-block is the same, then an activity-splitting element can be generated: it suffices to choose one of the base elements in E_a and adapt the basis by XORing this base element to the other base elements in E_a. The chosen base element is now an activity-splitting vector. This can be repeated for all sub-block positions.

6 Investigations on alignment in Keccak

In Section 3.2 we already reported on the activity patterns at row level of differences at the output of a Keccak-f round due to a single-row input difference at its input. In this section we report on a number of additional experiments we did to gain a better understanding on alignment in the Keccak-f round function.

First, we also investigated slice activity patterns. The lane transposition step π operates on the slices independently. Moreover, as the nonlinear step χ operates on the rows independently, it also operates on the slices independently. So it seems natural to consider the slices as sub-blocks and investigate the distribution of the activity patterns resulting from the $2^{25} - 1$ differences or masks with a single active slice. A slice-activity pattern consists of a lane of 2^ℓ bits.

Second, we have treated the special case of the so-called *kernel*. The mixing layer θ and its transpose θ^{T} have the property that they behave like the identity for inputs that have an even number of active bits in each column. The Keccak designers have called the set of such states the (*column parity*) *kernel* [2]. For in-kernel input differences, L behaves as a bit transposition and so the round function has weak diffusion (see also Figure 3). For that reason we also report on the results for the in-kernel sub-space of the states with a single active slice. This subspace has dimension 20. One can easily construct a basis for this subspace in the following way. Each base element has one active column and there are 4 base elements per column. Each of four base elements for column i have two active bits: one in position $(x, y, z) = (i, 0, 0)$ and one in position $(x, y, z) = (i, j, 0)$, where j ranges from 1 to 4.

We applied sets of differences and (linear) masks at one side of the round and investigated the distribution of the sub-block activity patterns at the other side of the round. More particularly, we covered all 24 combinations of the following choices:

– Propagation type: differences (DC) or masks (LC)
– Propagation direction: the round function or its inverse

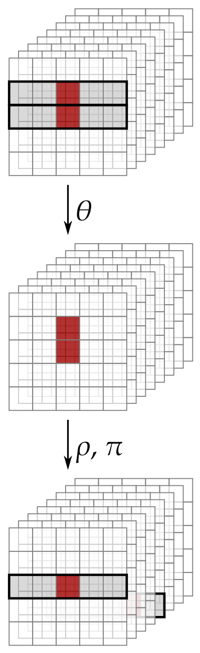

Fig. 3. Example of propagation of a difference lying in the column parity kernel of Keccak-f[200].

– Applied range: single-row, single-slice, in-kernel single-slice
– Sub-blocks determining resulting activity: rows and slices

We report on all these except the case of slice-activity patterns of single-row differences or masks. This would result in 4 more tables and we feel they would bring very little additional insight.

6.1 Distribution characteristics

In Table 1 the numbers in rows with permutation width 2^ℓ equal to 2 and 4, just counting the cardinality of the activity patterns may lead to incorrect conclusions if not interpreted with care. For example, the number 11 for $2^\ell = 2$ and for row $y = 0$ means that the 31 elements $L(A)$ are distributed over 11 activity patterns. One may expect there to be about 3 elements per activity pattern. However, it turns out that 21 of the elements have the same activity pattern and the remaining 10 each have a different activity pattern. To better capture this we introduce an additional quantifier, the *activity entropy*. It expresses the uncertainty on the activity pattern of $L(a)$ for a randomly chosen from A.

Definition 1. *Let* $\Pr(z|A)$ *be the probability that* $L(a)$ *has activity pattern* z *if* a *is chosen randomly from* A*. Then the activity entropy for a given set* A *and a mapping* L *with respect to a given sub-block partitioning is given by*

$$h = -\sum_z \Pr(z|A) \log_2 \Pr(z|A) .$$

The activity entropy ranges from 0 to $\log_2 \#A$. A value of 0 means that the input activity pattern fully determines the output activity pattern, the maximum value occurs when the elements of $L(A)$ have all different activity patterns. Low entropy values imply strong alignment and high entropy values weak alignment.

Additionally, in our experiments we noticed a saturation effect: if there is an activity pattern to which most input differences are mapped by L, it is the activity pattern with all rows or slices active. To capture this, we also list the average number of active sub-blocks in $L(a)$ over all input patterns $a \in A$ and denote it by $\overline{w}$. So for each distribution, we list the total number of activity patterns, the *cardinality* N, the entropy h and the average sub-block weight $\overline{w}$.

6.2 Difference propagation through the round function

We list the results on the distribution of the activity patterns at the output of a Keccak-f round due to differences with a single active row at its input in Table 2. Clearly, the low entropies for lane sizes 1 and 2 and the corresponding high average row weights indicate that most difference patterns map to the activity pattern with all active rows (the total number of rows for lane size 2^ℓ is $2^\ell 5$).

	$y=0$			$y=1$			$y=2$			$y=3$			$y=4$		
2^ℓ	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00
2	11	1.97	9.35	10	1.78	9.35	11	1.97	9.32	9	1.72	9.35	12	2.16	9.32
4	26	4.60	15.54	28	4.73	15.29	29	4.82	15.41	27	4.63	15.41	28	4.73	15.35
8	31	4.95	19.22	31	4.95	19.09	31	4.95	19.22	31	4.95	19.35	31	4.95	19.22
16	31	4.95	23.09	31	4.95	23.09	31	4.95	23.09	31	4.95	23.09	31	4.95	23.09
32	31	4.95	25.29	31	4.95	25.29	31	4.95	25.29	31	4.95	25.29	31	4.95	25.29
64	31	4.95	25.54	31	4.95	25.54	31	4.95	25.54	31	4.95	25.54	31	4.95	25.54

Table 2. Output row-activity distribution for single-row differences at round input

Table 3 lists the results on the distribution of the slice- and row-activity patterns for all input differences with a single active slice $z = 0$ and its in-kernel subspace. These results can be understood by considering the effect of the steps of L. First θ computes the column parities and complements some columns in slice $z = 0$ and slice $z = 1$. On the average, these two slices contain about 25 active bits. Subsequently ρ moves these bits to different slices and finally π moves the bits around within the slices. For the in-kernel subspace, θ behaves as the identity, so at the input of ρ is a state with on the average 12.5 active bits. ρ then moves these bits to different slices.

For slice-activity and considering all single-slice input differences, the average number of active slices is very high and the entropy remains very low up to lane size 8. This is due to saturation: for most input differences, ρ moves active bits to all slices. Limiting ourselves to the in-kernel differences, this saturation effect stops at lane size 4. Starting from lane size 32, the $2^{20} - 1$ in-kernel differences have all different output activity patterns. This is a direct consequence of the fact that the 25 ρ offset values are different for lane sizes 32 and 64. Over all single-slice input differences, the entropy does not reach its maximum value 25 and the average number of active slices reaches 22.5.

For row-activity the saturation effect is less powerful thanks to the fact that there are five times more rows than slices in a state. Considering all single-slice input differences, alignment becomes very weak at lane size 16 and entropy reaches its maximum at lane size 32. Limited to the in-kernel states, full entropy is reached at lane size 16.

	Resulting slice activity distribution						Resulting row activity distribution					
	full single-slice set			in-kernel subset			full single-slice set			in-kernel subset		
2^ℓ	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	1.00	1	0.00	1.00	31	1.00	4.84	26	1.00	4.84
2	3	0.0002	1.99	3	0.005	1.99	994	1.94	9.69	994	6.77	8.12
4	15	0.04	3.99	15	0.41	3.94	88743	11.86	15.50	50551	14.20	10.25
8	247	0.98	7.85	247	4.14	7.06	2004479	18.94	19.19	499711	18.60	11.50
16	50622	7.86	13.93	49999	14.18	10.25	16613375	23.68	22.37	1048575	20.00	12.00
32	5611775	19.66	20.25	1048575	20.00	12.50	33554431	25.00	24.50	1048575	20.00	12.50
64	12599295	22.87	22.50	1048575	20.00	12.50	33554431	25.00	24.75	1048575	20.00	12.50

Table 3. Output activity distributions for single-slice differences at round input

6.3 Difference propagation through the inverse round

When considering propagation of differences from the output of the round to the input of the round, propagation goes through $L^{-1} = \theta^{-1} \circ \rho^{-1} \circ \pi^{-1}$. The inverses of the two bit transpositions ρ and π do not differ much from their forward counterparts: ρ^{-1} still moves bit along the z axis within the lanes and π^{-1} still moves bits within the slices. θ^{-1} however behaves very different from θ. When applied to a difference with a single active bit (or a single column with odd parity), it flips the bits in about half of the columns of the state. See Figure 4 for an illustration.

Table 4 lists characteristics for the distribution of row selection patterns at the round input due to a difference at its output with a single active row. First π^{-1} moves the active bits to different rows and then ρ^{-1} to different slices. Then θ^{-1} adds a number of active columns. Clearly, the average number of active rows at the output indicates that most inputs result in output activity patterns with almost all rows active. This results in saturation for all lane sizes: a small number of differences at the output and low entropies.

	$y = 0$			$y = 1$			$y = 2$			$y = 3$			$y = 4$		
2^ℓ	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00
2	4	0.61	9.74	4	0.61	9.74	3	0.40	9.90	3	0.40	9.90	4	0.61	9.74
4	4	0.61	19.61	4	0.61	19.54	4	0.61	19.90	3	0.40	19.67	5	0.81	18.83
8	10	1.78	37.67	4	0.61	39.03	5	0.81	39.45	4	0.74	39.06	5	0.94	37.90
16	5	0.94	75.96	5	0.81	77.83	6	1.14	76.16	6	1.01	76.61	6	1.01	76.67
32	6	1.01	153.38	6	1.01	153.71	3	0.40	155.42	5	0.94	149.16	6	1.01	156.29
64	6	1.01	304.25	5	0.81	310.83	4	0.61	310.55	4	0.74	303.41	5	0.81	310.83

Table 4. Input row-activity distribution for single-row differences at round output

Table 5 lists the results on the distribution of the slice- and row-activity patterns at the round input for all output differences with a single active slice $z = 0$ and its in-kernel subspace. The observed behaviour is very similar to that for the case of single-row differences: high saturation for all lane sizes. There is no significant difference between the distributions for all single-slice differences and for the in-kernel ones. This is no surprise as in-kernel single-slice differences are mapped to differences outside the kernel by the bit transpositions before arriving at the input of θ^{-1}.

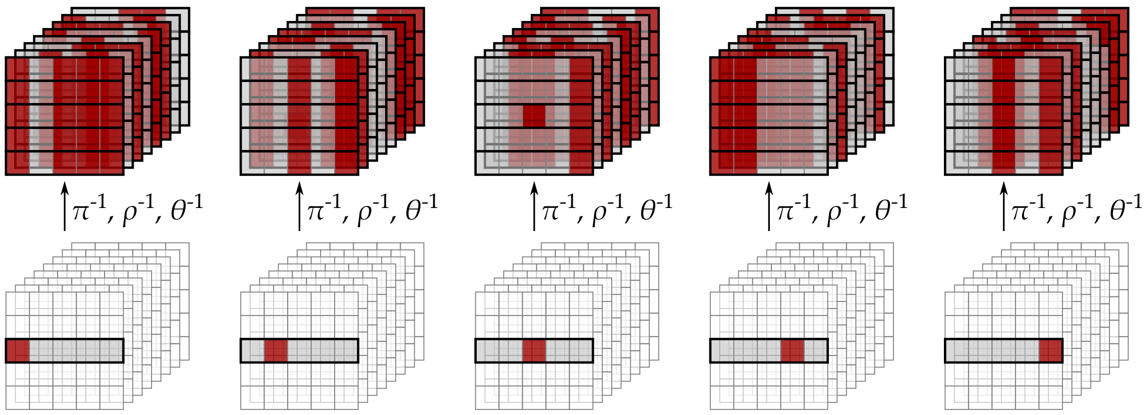
$\pi^{-1}, \rho^{-1}, \theta^{-1}$
$\pi^{-1}, \rho^{-1}, \theta^{-1}$
$\pi^{-1}, \rho^{-1}, \theta^{-1}$
$\pi^{-1}, \rho^{-1}, \theta^{-1}$
$\pi^{-1}, \rho^{-1}, \theta^{-1}$

Fig. 4. At the bottom, from left to right, the basis elements e_i of differences in row $(y, z) = (0, 0)$ of Keccak-f[200]; and on top, the corresponding values $L^{-1}(e_i)$.

	Resulting slice activity distribution						Resulting row activity distribution					
	full single-slice set			in-kernel subset			full single-slice set			in-kernel subset		
2^ℓ	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	1.00	1	0.00	1.00	31	1.00	4.84	31	1.69	4.68
2	3	0.0002	1.99	3	0.0003	1.99	675	1.96	9.68	335	1.86	9.68
4	15	0.02	3.99	11	0.02	3.99	11454	3.29	19.37	4174	3.28	19.37
8	98	0.27	7.97	66	0.27	7.97	21939	4.35	38.75	8126	4.34	38.75
16	1188	1.13	15.82	572	1.13	15.82	29892	5.38	77.50	12049	5.37	77.50
32	8856	2.83	31.26	2686	2.81	31.26	46230	6.06	154.65	11153	5.97	154.65
64	17413	4.29	61.40	6323	4.28	61.40	40359	6.12	305.78	14518	6.09	305.78

Table 5. Input activity distributions for single-slice differences at round output

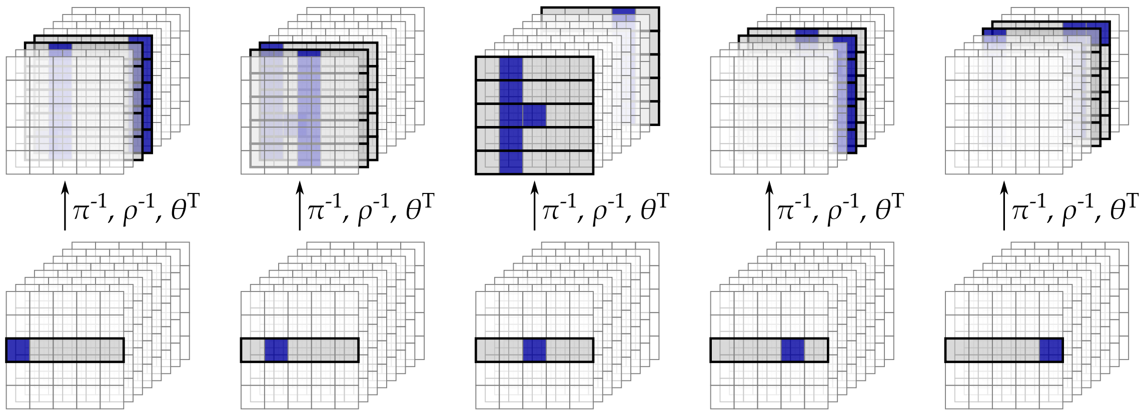
$\pi^{-1}, \rho^{-1}, \theta^{\mathrm{T}}$
$\pi^{-1}, \rho^{-1}, \theta^{\mathrm{T}}$
$\pi^{-1}, \rho^{-1}, \theta^{\mathrm{T}}$
$\pi^{-1}, \rho^{-1}, \theta^{\mathrm{T}}$
$\pi^{-1}, \rho^{-1}, \theta^{\mathrm{T}}$

Fig. 5. At the bottom, from left to right, the basis elements e_i of masks in row $(y, z) = (0, 0)$ of Keccak-f[200]; and on top, the corresponding values $L^{\mathrm{T}}(e_i)$. The active mask bits are depicted in blue.

6.4 Mask tracking through the round function

For tracking (linear) masks through the round function we adopt the convention taken in [2]. In this convention, propagation of a mask from the output of a function to its input is called *direct*. We introduce the term tracking here that is in our opinion more intuitive. For direct tracking, we have $L^{\mathrm{T}} = \theta^{\mathrm{T}} \circ \rho^{\mathrm{T}} \circ \pi^{\mathrm{T}}$. For bit transpositions it turns out that the transpose is equal to the inverse. So we have $L^{\mathrm{T}} = \theta^{\mathrm{T}} \circ \rho^{-1} \circ \pi^{-1}$. The transpose of θ is somewhat harder to obtain and we refer the interested reader to [2] for its derivation. What is important for understanding our results, is that the behaviour of θ^{T} is similar to that of θ: when applied to a mask with a single active bit (or a single column with odd parity), it flips the bits in two columns. See Figure 5 for an illustration.

Table 6 lists the characteristics of the distribution of row selection patterns at the round input due to a linear mask at its output with a single active row. We can track the masks through L. In this scenario, the bit transpositions are applied first, followed by the mixing operation θ^{T}. This scenario leads to distributions that are very similar to that of difference propagation from input to output of the round of Table 2. The only difference is that the saturation effect reaches until width 4.

	$y = 0$			$y = 1$			$y = 2$			$y = 3$			$y = 4$		
2^{ℓ}	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00
2	10	2.15	9.09	10	2.15	9.09	4	0.61	9.70	4	0.61	9.70	10	2.15	9.09
4	11	2.71	16.25	10	2.78	15.48	10	2.88	15.03	6	2.18	15.80	16	3.75	14.45
8	29	4.82	20.90	10	2.78	15.48	21	4.26	18.32	27	4.69	21.93	19	4.08	19.48
16	31	4.95	24.51	15	3.72	20.00	31	4.95	23.22	31	4.95	24.51	15	3.72	23.22
32	31	4.95	24.51	31	4.95	24.51	31	4.95	25.80	31	4.95	25.80	31	4.95	25.80
64	31	4.95	24.51	31	4.95	25.80	31	4.95	25.80	31	4.95	25.80	31	4.95	25.80

Table 6. Input row-activity distribution for single-row masks at round output

Table 7 lists the results on the distribution of the slice- and row-activity patterns for all input masks with a single active slice $z = 0$ and its in-kernel subspace. First of all, there is no significant difference between the distributions for all single-slice differences and for the in-kernel ones. This is no surprise as similar to the case of propagation of differences through the inverse round function, in-kernel single-slice masks are mapped to masks outside the kernel by the bit transpositions before arriving at the input of θ^{T}. The bit transpositions are applied first, moving the active bits in a single slice to different slices. At the input of θ there are a number of active bits that are relatively isolated when the lane size grows. The effect of θ is that for each of these bits two columns are complemented. This explains the numbers for large lane size values: each of these bit results in about two active slices, or 11 active rows after θ^{T}. When the lane size decreases, this leads to saturation.

	Resulting slice activity distribution						Resulting row activity distribution					
	full single-slice set			in-kernel subset			full single-slice set			in-kernel subset		
2^ℓ	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	1.00	1	0.00	1.00	31	1.00	4.84	31	1.69	4.69
2	3	0.0002	1.99	3	0.0004	1.99	667	1.96	9.69	337	1.86	9.69
4	15	0.04	3.99	11	0.04	3.99	9298	3.52	19.28	3618	3.50	19.28
8	114	0.97	7.86	88	0.97	7.86	8854	5.40	37.72	4351	5.38	37.72
16	8089	7.66	13.94	5476	7.64	13.94	22910	9.37	68.37	11699	9.30	68.38
32	4953311	21.00	20.25	632861	18.91	20.25	5644799	21.30	100.25	682391	19.09	100.25
64	22020095	24.20	22.50	1048575	20.00	22.50	22020095	24.20	112.50	1048575	20.00	112.50

Table 7. Input activity distributions for single-slice masks at round output

6.5 Mask tracking through the inverse round function

When considering mask tracking from the round input to the round output, the function to consider is $L^{-\mathrm{T}} = \pi \circ \rho \circ \theta^{-\mathrm{T}}$. The mixing layer $\theta^{-\mathrm{T}}$ behaves qualitatively like θ^{-1}, as depicted in Figure 6.

Table 8 lists characteristics for the distribution of row selection patterns at the round output due to a linear mask at its input with a single active row. In this scenario, the mixing transformation is applied first, followed by the bit transpositions. The fact that $\theta^{-\mathrm{T}}$ behaves very much like θ^{-1} clearly appears in the average number of active rows that is close to the total number of rows that is similar to that for the case of activity patterns at the round input for a single active row at the output of Table 4. However, the cardinality N and the entropy are very different. This can be understood as follows: $\theta^{-\mathrm{T}}$ makes about half of the columns active. Then ρ moves the active bits along the z-axis, followed by π maps the bits within the slices. This may lead to some rows without active bits here and there, and different from input to output. If the total number of rows is large enough, this leads to different activity patterns.

Table 9 lists the results on the distribution of the slice- and row-activity patterns for all input masks with a single active slice $z = 0$ and its in-kernel subspace. First of all, there is a huge difference between the distributions for the full set of single-slice masks and for the in-kernel ones. This is due to the fact that the bit transpositions come after the mixing layer. The behaviour of the kernel is actually identical to the case of difference propagation through the round (Table 3) as in both cases we study the propagation through $\pi \circ \rho$ (see also Figure 7). For the full single-slice set, we notice a very strong saturation for all lane sizes. This is due to the fact that $\theta^{-\mathrm{T}}$ has the tendency to fill the state up with active columns resulting in states with very high Hamming weight. The bit transposition then

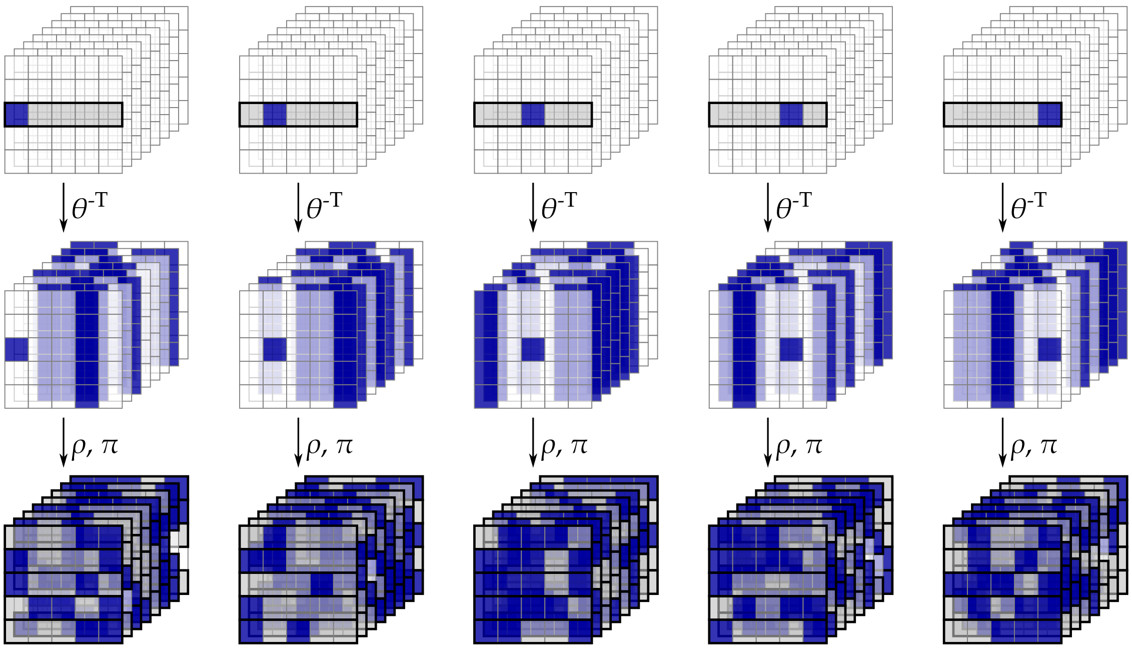

Fig. 6. On top, from left to right, the basis elements e_i of masks in row $(y,z) = (0,0)$ of KECCAK-f[200]; in the middle, the effect of $\theta^{-\mathrm{T}}$; and at the bottom, the corresponding values $L^{-\mathrm{T}}(e_i)$.

	$y=0$			$y=1$			$y=2$			$y=3$			$y=4$		
2^ℓ	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00	1	0.00	5.00
2	5	0.94	9.80	5	0.94	9.77	6	1.01	9.83	5	0.94	9.77	5	0.81	9.83
4	10	2.01	19.54	11	1.97	19.54	11	2.31	19.41	11	2.10	19.51	11	2.10	19.54
8	20	3.66	38.64	18	3.23	38.77	19	3.50	38.48	20	3.66	38.54	19	3.50	38.67
16	26	4.56	76.61	28	4.76	76.70	26	4.63	76.74	28	4.73	76.48	28	4.73	76.70
32	29	4.82	153.58	31	4.95	153.67	30	4.88	153.64	30	4.88	153.54	31	4.95	153.51
64	29	4.82	308.00	31	4.95	308.00	29	4.82	308.00	30	4.88	307.93	31	4.95	307.96

Table 8. Output row-activity distribution for single-row masks at round input

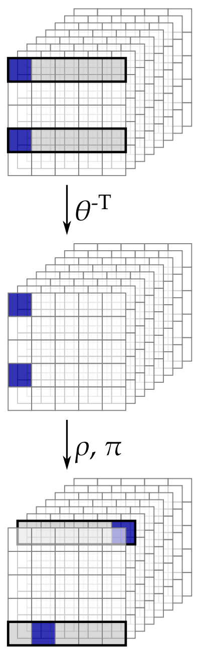

Fig. 7. Example of propagation of a mask lying in the column parity kernel of Keccak-f[200].

move these bits around, but the number of active bits remains large. The large cardinality N for the slice activity patterns is almost completely due to the kernel, while for the row activities, N is about twice as large for the set all masks than for the kernel. After some investigations, it turned out that this is due to the existence of a class of masks with a single active slice that lead to only two active slices after $\theta^{-\mathrm{T}}$. This class has the same size as the subset of masks in the kernel and consists of all single-slice masks for which all five columns have odd parity. This class only significantly affects the distributions of the row activity with high lane size. For the other cases, this effect is drowned in the saturation.

	Resulting slice activity distribution						Resulting row activity distribution					
	full single-slice set			in-kernel subset			full single-slice set			in-kernel subset		
2^ℓ	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	1.00	1	0.00	1.00	31	1.00	4.84	26	1.00	4.84
2	3	0.0002	1.99	3	0.005	1.99	994	1.95	9.69	994	6.77	8.13
4	15	0.02	3.99	15	0.41	3.94	50653	3.34	19.15	50551	14.20	10.25
8	247	0.24	7.97	247	4.14	7.06	500304	5.37	37.72	499711	18.60	11.50
16	49999	0.64	15.82	49999	14.19	10.25	1179743	7.08	74.44	1048575	20.00	12.00
32	1048640	1.21	31.27	1048575	20.00	12.50	2097215	7.02	148.94	1048575	20.00	12.50
64	1064960	1.46	61.40	1048575	20.00	12.50	2097195	6.66	298.50	1048575	20.00	12.50

Table 9. Output activity distributions for single-slice masks at round input

7 Conclusions

In this paper we have introduced the concept of alignment. It appears that especially for non-keyed primitives such as hash functions, strong alignment is a weapon in the hands of the attacker. While it appears that in the design of many cryptographic primitives strong alignment is the consequence of a design choice, this is not the case in Keccak. The outcome of our experiments suggest that the round function of Keccak-f has in general weak alignment. For the cases where there is no weak alignment, this is due to saturation where the vast majority of differences/masks propagate to activity patterns that cover the complete state.

References

1. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, *Cryptographic sponge functions*, January 2011, `http://sponge.noekeon.org/`.
2. ———, *The* Keccak *reference*, January 2011, `http://keccak.noekeon.org/`.
3. J. Daemen, *Cipher and hash function design strategies based on linear and differential cryptanalysis, PhD thesis*, K.U.Leuven, 1995.
4. J. Daemen and V. Rijmen, *The design of Rijndael — AES, the advanced encryption standard*, Springer-Verlag, 2002.
5. ———, *Understanding two-round differentials in AES*, SCN (Roberto De Prisco and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 4116, Springer, 2006, pp. 78–94.
6. ———, *Plateau characteristics and AES*, IET Information Security **1** (2007), no. 1, 11–17.
7. ECRYPT Network of excellence, *The SHA-3 Zoo*, 2011, `http://ehash.iaik.tugraz.at/index.php/The_SHA-3_Zoo`.
8. H. Gilbert and T. Peyrin, *Super-Sbox cryptanalysis: Improved attacks for AES-like permutations*, FSE (Seokhie Hong and Tetsu Iwata, eds.), Lecture Notes in Computer Science, vol. 6147, Springer, 2010, pp. 365–383.
9. K. Ideguchi, E. Tischhauser, and B. Preneel, *Improved collision attacks on the reduced-round Grøstl hash function*, ISC (Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, eds.), Lecture Notes in Computer Science, vol. 6531, Springer, 2010, pp. 1–16.
10. D. Khovratovich, M. Naya-Plasencia, A. Röck, and M. Schläffer, *Cryptanalysis of Luffa v2 components*, Selected Areas in Cryptography (A. Biryukov, G. Gong, and D. R. Stinson, eds.), Lecture Notes in Computer Science, vol. 6544, Springer, 2010, pp. 388–409.
11. L. R. Knudsen, *Truncated and higher order differentials*, Fast Software Encryption 1994 (B. Preneel, ed.), Lecture Notes in Computer Science, vol. 1008, Springer, 1994, pp. 196–211.
12. K. Matusiewicz, M. Naya-Plasencia, I. Nikolic, Y. Sasaki, and M. Schläffer, *Rebound attack on the full Lane compression function*, Asiacrypt (M. Matsui, ed.), Lecture Notes in Computer Science, vol. 5912, Springer, 2009, pp. 106–125.
13. F. Mendel, C. Rechberger, M. Schläffer, and S. Thomsen, *The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl*, FSE (Orr Dunkelman, ed.), Lecture Notes in Computer Science, vol. 5665, Springer, 2009, pp. 260–276.
14. ———, *Rebound attacks on the reduced Grøstl hash function*, CT-RSA (Josef Pieprzyk, ed.), Lecture Notes in Computer Science, vol. 5985, Springer, 2010, pp. 350–365.
15. M. Naya-Plasencia, *Scrutinizing rebound attacks: new algorithms for improving the complexities*, Cryptology ePrint Archive, Report 2010/607, 2010, `http://eprint.iacr.org/`.
16. NIST, *Federal information processing standard 197, advanced encryption standard (AES)*, November 2001.