First Analysis of KECCAK

Jean-Philippe Aumasson[1] and Dmitry Khovratovich[2]

[1] FHNW, Switzerland
[2] University of Luxembourg

Abstract. We apply known automated cryptanalytic tools to the KECCAK-f[1600] permutation, using a triangulation tool to solve the CICO problem, and cube testers to detect some structure in the algebraic description of the reduced KECCAK-f[1600]. The applicability of our tools was notably limited by the strength of the inverse permutation.

Unless otherwise stated, we consider the KECCAK permutation used in the KECCAK submission to SHA-3, that is, the function called KECCAK-f[1600] in [2].

1 Solving the CICO problem

1.1 Preliminaries

Assume we try to detect non-randomness in a function f with n-bit input and m-bit output. Consider the following problem: find a solution to

$$f(x) = y \tag{1}$$

such that first q bits of x and y are zero, $q \leq \min(n, m)$. Brute-force search, which works for any f, requires about 2^q computations of f. This bound remains the same even if $n = m$ and f is invertible.

One expects that for a "good" hash transformation, this problem should have the same workload. Although non-trivial solutions do not imply collision and preimage weaknesses, they are a first sign of non-ideal behavior.

Such a weakness has been previously discovered in a reduced version of MD6 [3]: for 26 rounds of the compression function four bits can be fixed, two bits for 30 rounds, and one bit for 33 rounds. It has been recently suggested that so many rounds can be broken due to slow diffusion in MD6.

A more general problem has been proposed by the designers of KECCAK [2]. Assume that $n = m$ and define $X \subseteq \{0,1\}^n$ as a set of possible inputs and $Y \subseteq \{0,1\}^n$ as a set of possible outputs. Then find a solution to Eq. (1) with $(x, y) \in X \times Y$; [2, §§4.2.4] calls this the *CICO problem* (Constrained-Input Constrained-Output).

1.2 Triangulation algorithm

The triangulation algorithm was proposed in [4] as a tool for solving systems of non-linear equations (see Appendix A), which appear in the differential attacks. Given the constraints on the internal variables, the algorithm outputs a special set of variables, called *free variables*. Those variables can be assigned randomly; and this assignment together with pre-fixed variables completely and efficiently determines the whole execution. The fewer variables are fixed, the better the algorithm works.

Consider the application of the triangulation algorithm to the analysis of hash functions. Evidently, the initial value and the message completely determine all the execution of the transformation. Following the framework of the Gaussian elimination process, such variables are called *free variables* since they can be assigned randomly and independently. While it is trivial to find a set of free variables when there are no constraints, it becomes harder if some variables are pre-fixed and are positioned far from one another. The fixed bits of the input and the output (CICO) are an example of such case.

Underlying ideas. The triangulation algorithm iteratively searches for a variable involved in only one equation, and that can be expressed as a function of the other variables involved in that equation. If such a variable is found this implies that it can be determined in the last step when all the other variables are known. Then the equation and the variable are removed from the consideration (in Gaussian elimination terminology, they are put on the diagonal), and the process goes on.

Applications. So far, the algorithm properties are not carefully investigated. However, based on empirical observations, we conjecture that it stops working if the distance between fixed variables is twice the number of rounds needed for the full diffusion. If the diffusion is different in the backward direction, the bound may change. Thus the efficiency of the triangulation algorithm applied to KECCAK is determined by its diffusion properties.

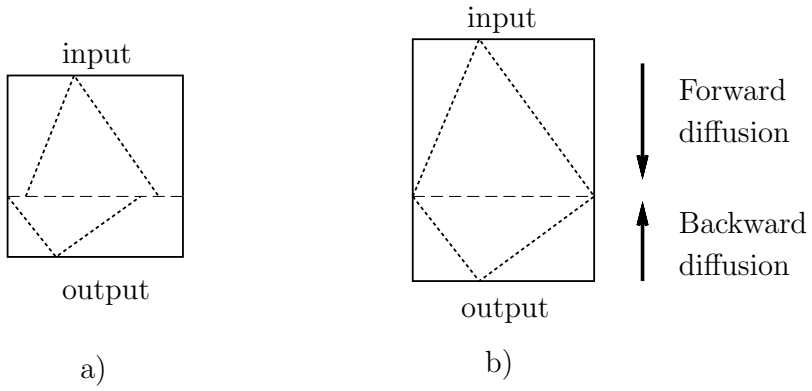

Fig. 1. Triangulation algorithm: a) Works b) Full diffusion, stops.

1.3 Properties of KECCAK

The KECCAK round consists of five transformations, of which only two provide diffusion (χ and θ), and only one is non-linear (χ). Let us briefly describe their diffusion properties.

The θ operation. The θ operation is an efficient *linear* transformation. It just XORs 11 bits into one. Therefore, every bit affects eleven more by θ. Surprisingly, the inversion of θ, though not so efficient, has much better diffusion properties. For example, a change in one bit in the whole state results in a change in about a half of the state. This significantly reduces the efficiency of the triangulation algorithm, since only one and a half rounds are needed for the full diffusion in the backward direction.

The χ operation. The χ operation is the only *nonlinear* part of KECCAK. It is an invertible transformation over Z_2^5 with good diffusion properties. The triangulation algorithm can process it both as a set of independent equations and as a single invertible transformation. We figured out that the exact form of the equations is not important for the performance of the algorithm, while the nonlinearity does matter.

Comparison with MD6. We have seen the inverse KECCAK permutation achieves full diffusion in one and a half round. For comparison, the MD6 compression function needs about 15 rounds to achieve full diffusion. The speed of diffusion is the same in both directions. The diffusion in KECCAK appears to be significantly better: after three rounds every bit affects any other one, and the diffusion in the backward direction is even faster.

1.4 Results

Using the SAGE computer algebra software, the designers of KECCAK made some experiments [2, §§5.9.4] to solve the CICO problem for KECCAK-f[25], which is the KECCAK permutation with one-bit words, instead of 64-bit. Fixing 16 bits in the input and nine in the output, they were able to solve the problem for two rounds of KECCAK-f[25], but for three and more rounds, the program ran out of memory.

Using our triangulation tool [4], we could find solutions of Eq.(1) for KECCAK-f[1600] with three rounds. Versions with more rounds suffer from the fast backward diffusion in θ. We think, that if it had the same diffusion properties as in the forward direction, at least six rounds could be attacked.

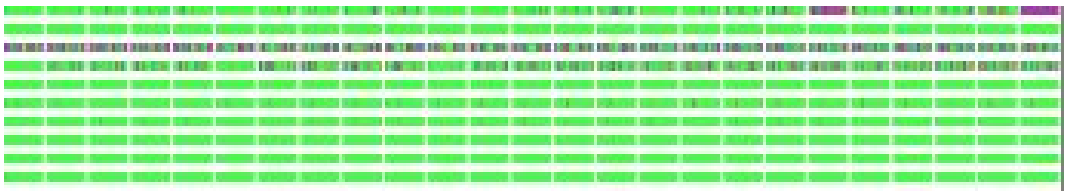

Fig. 2. Solution for three rounds. Free variables (red and blue) among the other variables (green).

2 Cube Testers

In the documentation of KECCAK, Table 5.4 reports upper bounds on the degree of monomials in the algebraic normal form (ANF) of the permutation, as function of the number of rounds. One can thus predict the existence of cube testers [1], given these bounds; for example, on four rounds monomials exist up to degree nine, hence a cube tester with cubes of degree larger than nine will give a null superpoly, allowing one to distinguish the permutation from a random permutation within about 2^{10} queries. The documentation even suggests that for up to three rounds, even low-degree cubes can be used to build cube testers on KECCAK-f[1600].

Besides the cube testers predicted by the algebraic analysis in [2], we could not find cubes yielding significantly better distinguishers on the KECCAK permutation. Our search included the use of an evolutionary algorithm, which is well suited for search in space of unknown structure, and we considered all the 1600 inputs bits as possible cube indices.

Yet, we observed a peculiar algebraic structure in the KECCAK permutation. In particular, we observed that the KECCAK permutation becomes significantly stronger after four rounds, since:

- with three rounds, cube testers with only *quadratic* cubes can be used as distinguishers (testing the balance of their superpoly);
- with four rounds, one needs cubes of degree at least nine.

Fig. 3 shows the number of significantly biased output bits on three rounds, as a function of the position of the two-bit cube tested (taking a cube formed of two contiguous bits, for each bit offset, e.g. bits 0 and 1, bits 1 and 2, bits 2 and 3, etc.). The best cube is formed of bits 642 and 643, leading to imbalanced superpolys for 695 out of the 1600 components of the permutation. For comparison, the worst cube, bits 319

and 320, shows bias for 377 output bits. Fig. 3 suggests that the strength of quadratic cubes on three rounds is rather heterogeneous, for the domain is split into 19 regions wide of at least 64 bits, such that each two consecutive regions have significantly different strength, with respect to our methods.

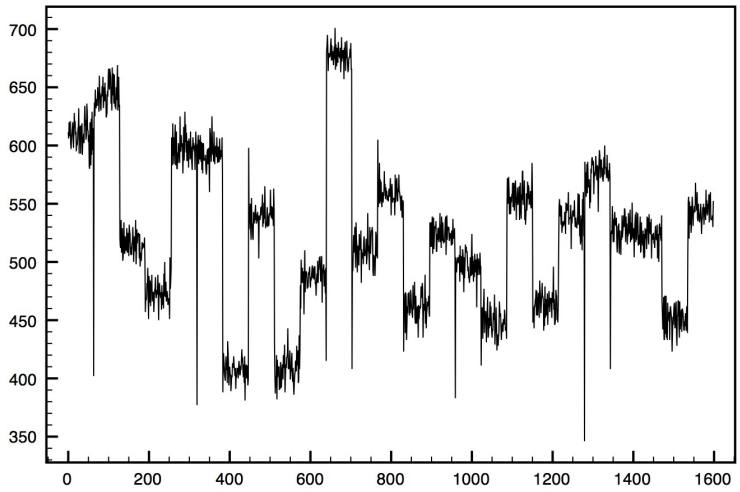

Fig. 3. Distribution of the number of biases on three rounds of the KECCAK permutation, as a function of the position of the quadratic cube tested (with x-coordinates in *bits*).

Fig. 4 shows the number of significantly biased output bits on three rounds, as a function of the position of the three-bit cube tested (taking a cube formed of three contiguous bits, for each bit offset, e.g. bits 0, 1, and 1, bits 1, 2, and 3, etc.). The curve has a similar shape as for two-bit cubes, and for both curves the region around 700 has the most biased components. However, a pairwise comparison of the regions for each curve does not give the same results: for example, the third region is slightly "better" than the fourth on Fig. 3, while it is slightly "lesser" on Fig. 4.

Fig. 5 shows the number of output bits with a *constant* superpoly on four rounds, as a function of the position of the nine-bit cube tested (taking cubes with bits 0–8, 8–17, etc.). The shape with peaks to 1600 suggests that the cubes tested either have a constant superpoly for all the 1600 components (that is, they can be used for building a distinguisher), or have no constant superpoly for any of the 1600 components (as expected for an ideal function).

Appendix B gives the curves for three rounds and cubes of four and five bits.

Our observations show that the KECCAK permutation reduced to three and four rounds, besides its reduced degree, does not have an ideal algebraic structure, and that the 1600-bit input can be divided into regions of various algebraic strength. In particular, we observed on four rounds an "all-or-nothing" behavior of our cube testers.

One may thus conjecture that such structure can be observed as long as the algebraic degree of the KECCAK permutation is not maximal, that is, on up to ten rounds.

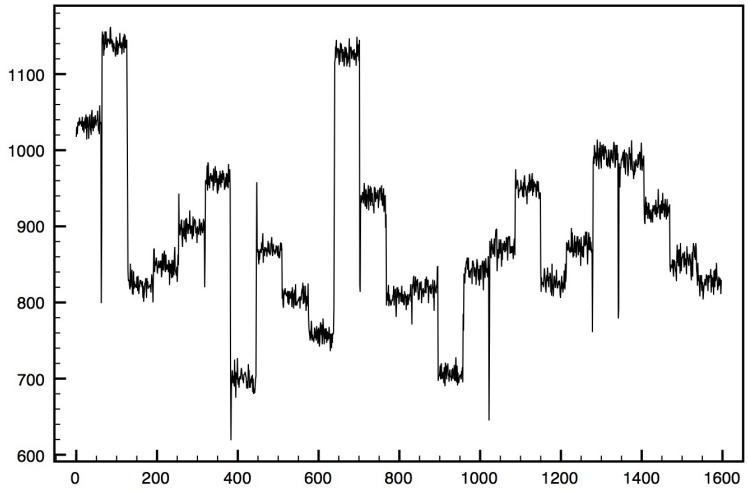

Fig. 4. Distribution of the number of biases on three rounds of the Keccak permutation, as a function of the position of the cubic cube tested (with x-coordinates in *bits*).

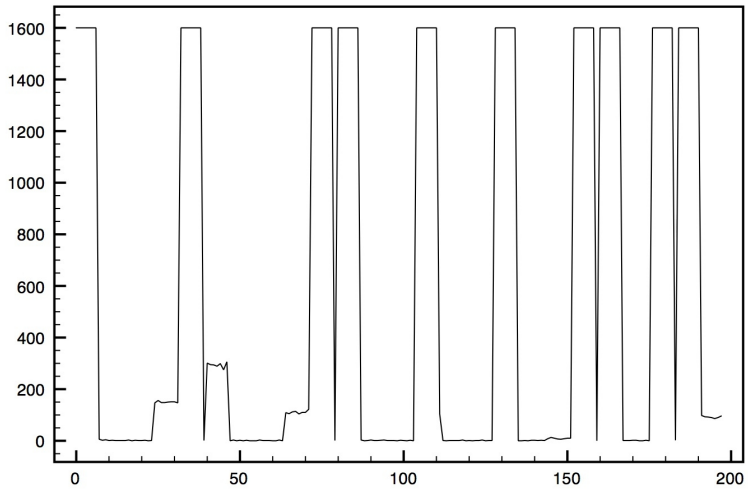

Fig. 5. Distribution of constant superpoly's on four rounds of the Keccak permutation, as a function of the position of the nine-bit cube tested (with x-coordinates in *bytes*).

3 Differentials

We first found, by simple linear algebra, an input difference that yields a difference of Hamming weight one after θ:

```
789AF135  DE26BC4D  26BC4D78  09AF135E  AF135E26
EBC4D789  C4D789AF  7135E26B  35E26BC4  CD789AF1
789AF134  DE26BC4D  26BC4D78  09AF135E  AF135E26
EBC4D789  C4D789AF  7135E26B  35E26BC4  CD789AF1
789AF134  DE26BC4D  26BC4D78  09AF135E  AF135E26
EBC4D789  C4D789AF  7135E26B  35E26BC4  CD789AF1
789AF134  DE26BC4D  26BC4D78  09AF135E  AF135E26
EBC4D789  C4D789AF  7135E26B  35E26BC4  CD789AF1
789AF134  DE26BC4D  26BC4D78  09AF135E  AF135E26
EBC4D789  C4D789AF  7135E26B  35E26BC4  CD789AF1
```

This difference is of Hamming weight 851, which is consistent with the claims in [2, §§5.3.2] that "applying θ^{-1} to a difference pattern with a single active bit results in a difference pattern with about half of the bits active."

On three rounds, we observe that this input difference yields differences on 257 bits with probability one, for a random 1600-bit input. Furthermore, most of the bits are strongly biased, as Fig. 6 shows. On four rounds, our difference also yields strong biases, as shown by Fig. 7.

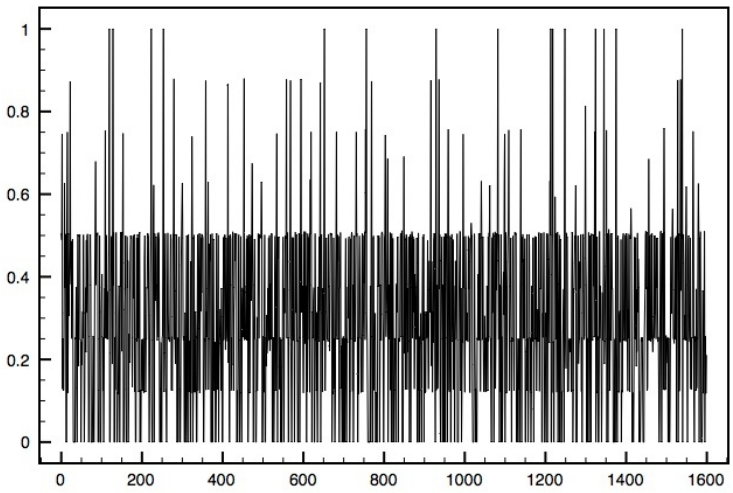

Fig. 6. Probability distribution of the output for three rounds.

When looking at the inverse permutation, it seems more difficult to find such biased distributions, for the transformation now starts with the nonlinear operator χ, and benefits of the diffusion of θ^{-1}, faster than that of θ, as highlighted in §1.3.

From the existence of differentials with probability one for the inverse permutation, one may be able to describe *impossible differentials*, with a miss-in-the-middle strategy. However, we could not find probability-one differentials for more than one round of the inverse permutation. Thus, we could not find impossible differentials on five (or more) rounds of the KECCAK permutation. As for the CICO problem, if the backward

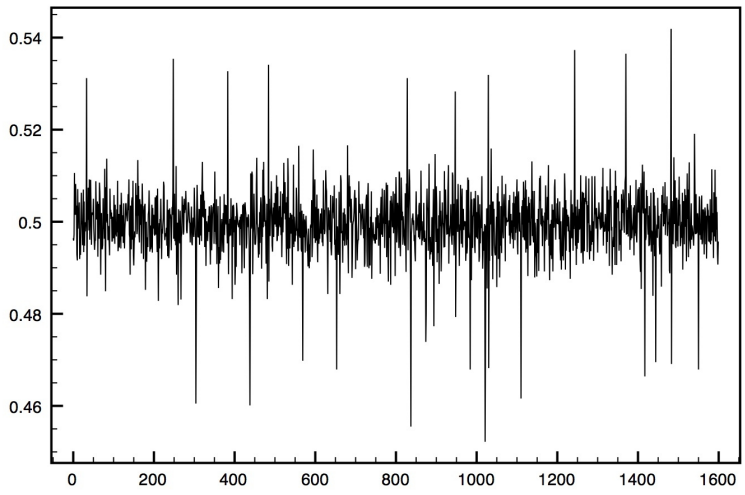

Fig. 7. Probability distribution of the output for four rounds.

transformation had the same diffusion properties as in the forward direction, then five of six rounds may have been attackable.

4 Conclusion

We applied known automated cryptanalysis tools (triangulation tool, and cube testers) to the core permutation of the Keccak hash function. Our results are consistent with the claims in the documentation of Keccak, and suggest that Keccak enjoys a fast diffusion, and a fast growth of nonlinearity. In particular, our observations with cube testers were consistent with the algebraic analysis of the designers, and highlighted some particular structure on three and four rounds. Finally, the inverse Keccak permutation appears to be significantly stronger, with respect to our tools.

The Keccak documentation claims that 13 rounds are sufficient to prevent any structural distinguishers, and nine rounds against collisions. Our observations suggest that these claims are reasonable, and that the proposed number of rounds in the Keccak submission (18) provides a comfortable security margin.

References

1. Jean-Philippe Aumasson, Itai Dinur, Willi Meier, and Adi Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In *FSE 2009*.
2. Guido Bertoni, Joan Daemen, Michaeël Peeters, and Gilles Van Assche. Keccak sponge function family main document. Submission to NIST (updated), 2009. Version 1.1.
3. Dmitry Khovratovich. Nonrandomness of the 33-round MD6. Rump session of FSE 2009.
4. Dmitry Khovratovich, Alex Biryukov, and Ivica Nikolić. Speeding up collision search for byte-oriented hash functions. In *CT-RSA 2009*.

A Triangulation tool

We make our triangulation tool publicly available, as a Windows executable binary file, at

https://cryptolux.uni.lu/mediawiki/uploads/0/03/Keccak-tool.zip

The archive contains instructions for using the program. Fig. 8 shows a screenshot of the main window.

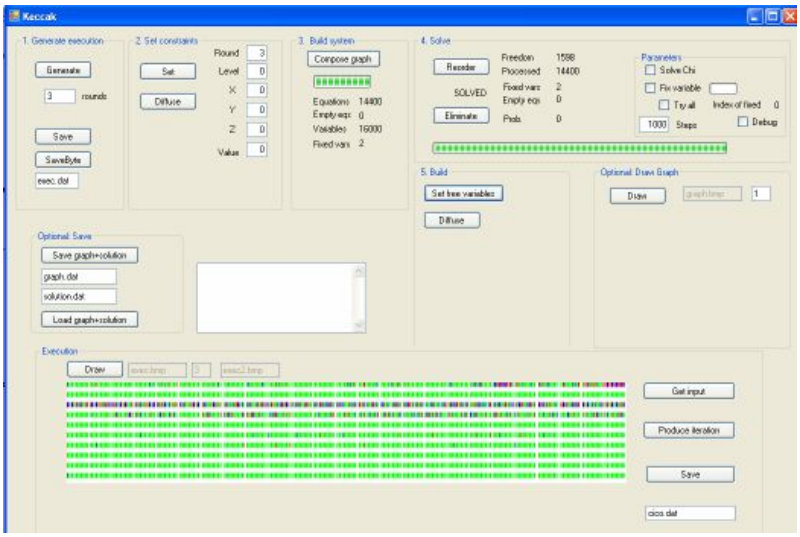

Fig. 8. Main window of our triangulation tool, as used on KECCAK.

B Results with cube testers

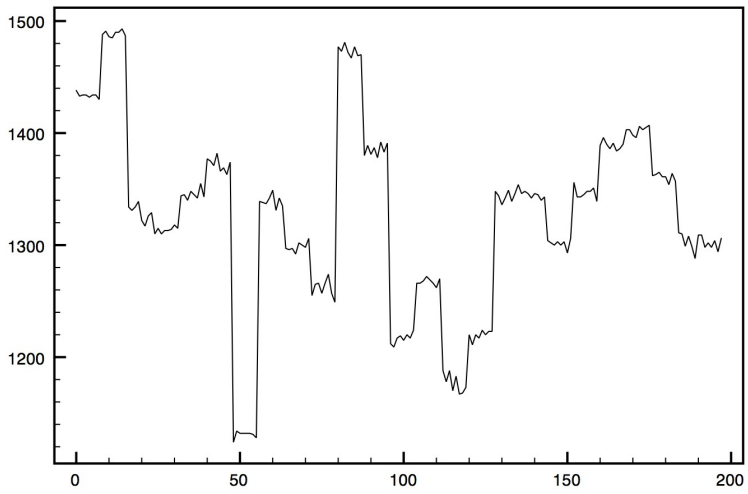

Fig. 9. Distribution of the number of biases on three rounds of the KECCAK permutation, as a function of the position of the four-bit cube tested (with x-coordinates in *bytes*).

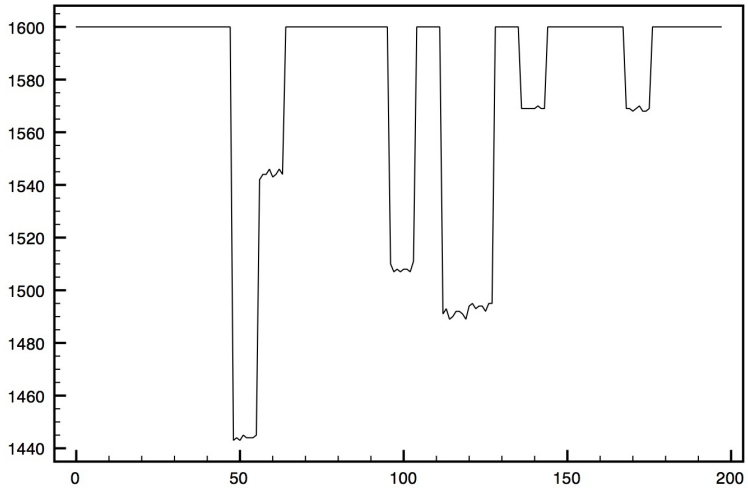

Fig. 10. Distribution of the number of biases on three rounds of the Keccak permutation, as a function of the position of the five-bit cube tested (with x-coordinates in *bytes*).