Inside KECCAK (Extended Abstract)

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

[1] STMicroelectronics
[2] NXP Semiconductors

Abstract. In this extended abstract we summarize the algebraic properties of the building blocks of KECCAK relevant in cryptanalysis and implementation. KECCAK is a sponge function family that makes use of an underlying set of fixed-length iterated permutations called KECCAK-f. The round function of KECCAK-f consists of 5 step mappings chosen for their ease of implementation and strong long-term diffusion and nonlinearity. We discuss the individual properties of these step mappings such as invertibility, translation-invariance and (when applicable) non-linearity, their interactions and the implications for resistance to attacks such as linear and differential cryptanalysis.

Keywords: Keccak, algebraic properties

1 Introduction

KECCAK is a parameterized sponge function that uses as a building block a set of 7 related permutations of different sizes, called KECCAK-f. There are 7 KECCAK-f permutations, indicated by KECCAK-$f[b]$, where $b = 25 \times 2^\ell$ and ℓ ranges from 0 to 6. KECCAK-$f[b]$ is a permutation over $s \in \mathbb{Z}_2^b$.

The permutation KECCAK-$f[b]$ operates on a state a that is a three-dimensional array of elements of GF(2) denoted by $a[x][y][z]$. The size of the array depends on ℓ and is given by $a[5][5][w]$. It consists of $w = 2^\ell$ *slices*, one for each value of z. A slice contains 25 bits arranged in 5 rows, one for each value of y, and 5 columns, one for each value of x. A state can also be seen as 25 *lanes*, orthogonal to the slices. For the different values of ℓ, the *lane size* w ranges from 1 to 64 in powers of two.

KECCAK-$f[b]$ is an iterated permutation, consisting of a sequence of n_r rounds with $n_\mathrm{r} = 12 + 2\ell$. For KECCAK-$f[1600]$ this results in 24 rounds.

KECCAK is specified in [1] and the properties of its components is described in detail in [2]. In this abstract we give an overview of the algebraic properties of the components of the KECCAK-f round function and refer to [2] for the details. We first list the design principles underlying the KECCAK-f permutations. This is followed by a discussion on the algebraic properties of its components. Finally we explain what this means for the interaction between the different components relevant in cryptanalysis and implementation.

2 Design principles

As a consequence of the underlying design principles, the step mappings have a number of remarkable properties, quite rare for components of symmetric cryptographic primitives. These design principles are the following:

- **Minimalism:** each step mapping has a simple structure and description with operations in GF(2). Each step mapping has a particular purpose and omitting one of them would result in a collapse of security.
- **Symmetry:** the processing of the different bits of the state depends as little as possible on the bit positions. This is reflected in the property of translation-invariance [2, Section 6.1].
- **Transferability of analysis:** analysis of smaller versions of KECCAK-f is relevant for the larger versions. This is reflected in the Matryoshka structure [2, Section 6.2].
- **Efficiency, but only where needed:** the round function is easy to implement on a wide range of platforms and is suited for side-channel resistant implementations. There are no such requirements for the inverse of the round function (as opposed to a block ciphers).

3 The step mappings and their properties

A round consists of five step mappings:

- χ: nonlinearity
- θ: linear mixing for diffusion
- π: bit transposition in the xy direction for *dispersion* (moving nearby bits to remote positions)
- ρ: bit transposition in the z direction for dispersion
- ι: round constants addition (in GF(2)) for destruction of symmetry

In the following subsections we will summarize the algebraic properties of the first four step mappings. For a more detailed treatment we refer to [2, Section 6.3]

3.1 The nonlinear mapping χ

χ is the nonlinear mapping of the round function and operates on the rows independently. It is translation-invariant in any direction and its operation is independent of the lane size w. χ simply consists of adding to each bit the product (AND) of two other statebits, with one of them complemented. It follows that its algebraic degree is 2. Its invertibility is however not trivial and its inverse has algebraic degree 3.

The fact that the algebraic degree is two has consequences for the difference propagation through χ and correlation across χ.

Given a difference pattern a at the input of χ, the output difference patterns form a linear affine variety. This can be represented by an offset and a basis with $w(a)$ elements and hence has $2^{w(a)}$ elements, with $w(a)$ the *restriction weight* of the difference pattern a. For all the output difference patterns, the differential probability (DP) is equal to $2^{-w(a)}$ and is fully determined by the input difference pattern a. A difference propagation (a, b) imposes w linear conditions on state bits at the input of χ.

Given a bit parity at the output of χ defined by a *(linear) mask* u, the masks of the bit parities at the input of χ that have a non-zero correlation with it, form a linear affine variety. This can

be represented by an offset and a basis with $w(u)$ elements and hence has $2^{w(u)}$ elements, with $w(u)$ the *correlation weight* of the mask u. For all the compatible input masks the magnitude of the correlation is equal to $2^{-w(a)/2}$ and is fully determined by the output mask.

3.2 The linear mixing layer θ

θ is the linear mixing layer of the round function. It is translation-invariant in any direction and operates independently of the lane size w if $w > 1$. θ simply consists of the addition to each bit of the state the parity of two columns. Every output bit of θ is the bitwise sum of 11 input bits. Computing the inverse of θ is a non-trivial exercise and requires calculations in ideals over polynomial rings in three variables. It turns out that every input bit of θ is the bitwise sum of about half of its output bits. The inverse of θ depends on the lane size w.

For the subset of states with an even number of 1-bits in each column, θ is the identity mapping. We call this subset the (*column parity*) *kernel*. In general, the column parity (i.e., a two-dimensional array containing the parities of all columns) of state a fully determines the column parity of $\theta(a)$. The column parity imposes lower bounds on the total number of bits equal to 1 (active bits) in a and $\theta(a)$ jointly: a parity-specific branch number.

3.3 The transposition of lanes π

π is one of the two dispersion layers and its purpose is to change the relative positions of bits in the slices. It is translation-invariant in the direction of the z-axis and operates independently of the lane size w. It moves the bits within a slice to other positions where the new position (X, Y) is computed from the old position (x, y) by a multiplication with a 2×2 matrix M with elements in GF(5). The inverse of π is defined by the inverse matrix.

The multiplication by M is a linear mapping with a number of interesting properties. It leaves the origin $(0, 0)$ invariant and moves the other 24 positions around in a single cycle. The relative position of two points before π fully determines their relative position after π. As such it respects the 6 possible partitions of the 25 (x, y) positions according to lines parallel to lines through the origin. As such, the 5 positions in a row are mapped to a column, a falling 2-slope, a rising 2-slope, a falling 1-slope, a rising 1-slope and back to a row again in 6 iterations.

3.4 The intra-lane translation ρ

ρ is one of the two dispersion layers and its purpose is to move bits in the same slice to different slices. It is translation-invariant in the direction of the z-axis and its operation depends on the lane size w. It just (cyclically) translates the bits within the lane in the direction of the z-axis, by a different offset for each lane. The translation offsets are the same for all values of w, but as the translation are cyclic, in practice their values are reduced modulo w.

The translation offsets are defined by the sequence $i(i+2)/2$. For any lane size 2^ℓ the sequence $i(i+2)/2 \bmod 2^\ell$ has period $2^{\ell+1}$ and runs through all integers smaller than 2^ℓ.

4 Interaction between the step mappings

4.1 Differential and linear cryptanalysis

The fact that χ is the only nonlinear step mapping in a round, makes that the difference patterns at the input of a round defines a linear affine variety of possible difference patterns at the input of

the next round. Similarly, a mask at the output of a round defines a linear affine variety of possible masks at the output of the previous rounds.

These linear affine varieties can be seen as the product a linear affine variety in the kernel and a linear affine variety ortogonal to the kernel. As such, difference propagations with small branch number can be identified both in the case of differential and linear cryptanalysis. this can be used in proving lower bounds for the weight of linear and differential trails.

All step mappings are translation-invariant in the direction of the z-axis. This implies that for any differential or linear trail, there are w equivalent ones that are just z-translated versions. On the other hand, as a consequence of the Matryoshka structure of Keccak-f, any trail for a given lane size w also defines one for w with twice the weight [2, Section 6.5.2].

Exploiting the kernel allows finding low weight trails for 2 and 3 rounds, but not beyond [2, Sections 6.5.3 to 6.5.6].

4.2 Implementation

Keccak-f is well suited to be implemented in software with bitwise logical operations and cyclic shifts. These cyclic shifts implement the translations in the direction of the z-axis and require the word size of the CPU to be equal to the lane size w. However, the high level of symmetry and the translation-invariance in the direction of the z-axis allow the use of cyclic shifts also when the CPU word length divides the lane size. This can be done with the technique of bit interleaving. In its simplest form, a lane is stored in two words: one word containing the bits with even z coordinates and one the bits with odd z coordinates [2, Section 9.2.2].

The fact that the round function of Keccak-f only has degree 2 makes it very well suited for implementations that provide resistance against differential power analysis. This low degree makes it suitable for implementing secret sharing methods, both in software and in hardware [2, Sections 9.3.4 and 9.4.4].

References

1. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, Keccak *specifications, version 2*, NIST SHA-3 Submission, September 2009, `http://keccak.noekeon.org/`.
2. _____, Keccak *sponge function family main document*, NIST SHA-3 Submission (updated), June 2010, `http://keccak.noekeon.org/`.