Keccak specifications

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

`http://keccak.noekeon.org/`

Version 2 – September 10, 2009

Keccak (pronounced [kɛtʃak]) is a family of hash functions that are based on the sponge construction [1] and use as a building block a permutation from a set of 7 permutations. In this document, we specify these permutations, the Keccak sponge functions and the parameter values we propose for use in our SHA-3 candidates. We also give conventions for bit and byte numbering, for using the arbitrary-long output mode and for naming parts of the Keccak state. These specifications give all the necessary information to implement the Keccak sponge functions.

For more information, and for the reference code, please refer to the Keccak web page given above.

1 The Keccak-f permutations

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 $s \in \mathbb{Z}_2^b$, where the bits of s are numbered from 0 to $b-1$. We call b the width of the permutation. Our SHA-3 candidates use Keccak-$f[1600]$.

In the sequel, the permutation Keccak-$f[b]$ is described on a state a that is a three-dimensional array of elements of GF(2), namely $a[5][5][w]$, with $w = 2^\ell$. The mapping between the bits of s and those of a is $s[w(5y+x)+z] = a[x][y][z]$. The expression $a[x][y][z]$ with $x, y \in \mathbb{Z}_5$ and $z \in \mathbb{Z}_w$, denotes the bit in position (x, y, z). It follows that indexing starts from zero; expressions in the x and y coordinates should be taken modulo 5 and expressions in the z coordinate modulo w. We may sometimes omit the $[z]$ index, both the $[y][z]$ indices or all three indices, implying that the statement is valid for all values of the omitted indices.

Keccak-$f[b]$ is an iterated permutation, consisting of a sequence of n_r rounds R, indexed with i_r from 0 to $n_\mathrm{r}-1$. A round consists of five steps:

$$\mathrm{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta, \text{ with}$$

$$\begin{array}{rcl}
\theta: & a[x][y][z] & \leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1], \\
\rho: & a[x][y][z] & \leftarrow a[x][y][z-(t+1)(t+2)/2], \\
 & & \quad \text{with } t \text{ satisfying } 0 \le t < 24 \text{ and } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \text{ in } \mathrm{GF}(5)^{2\times 2}, \\
 & & \quad \text{or } t = -1 \text{ if } x = y = 0, \\
\pi: & a[x][y] & \leftarrow a[x'][y'], \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}, \\
\chi: & a[x] & \leftarrow a[x] + (a[x+1]+1)a[x+2], \\
\iota: & a & \leftarrow a + \mathrm{RC}[i_\mathrm{r}].
\end{array}$$

The additions and multiplications between the terms are in GF(2). With the exception of the value of the round constants $\mathrm{RC}[i_\mathrm{r}]$, these rounds are identical. The round constants are given by (with the first index denoting the round number)

$$\mathrm{RC}[i_\mathrm{r}][0][0][2^j - 1] = \mathrm{rc}[j + 7i_\mathrm{r}] \text{ for all } 0 \le j \le \ell,$$

and all other values of $\mathrm{RC}[i_\mathrm{r}][x][y][z]$ are zero. The values $\mathrm{rc}[t] \in \mathrm{GF}(2)$ are defined as the output of a binary linear feedback shift register (LFSR):

$$\mathrm{rc}[t] = \left(x^t \bmod x^8 + x^6 + x^5 + x^4 + 1\right) \bmod x \text{ in } \mathrm{GF}(2)[x].$$

The number of rounds n_r is determined by the width of the permutation, namely,

$$n_\mathrm{r} = 12 + 2\ell.$$

2 The KECCAK sponge functions

We obtain a KECCAK$[r, c, d]$ sponge function, with parameters capacity c, bitrate r and diversifier d, if we apply the sponge construction as described in [1], to KECCAK-$f[r + c]$ and by applying a specific padding to the input. This is specified in Algorithm 1, where $||$ denotes concatenation, $\oplus$ bitwise addition of strings of equal length and $\lfloor s \rfloor_n$ truncation of string s to its first n bits.

Algorithm 1 KECCAK$[r, c, d]$

```
Input $M \in \mathbb{Z}_2^*$
Output $Z \in \mathbb{Z}_2^*$
$P = \text{pad}(M, 8)||\text{enc}(d, 8)||\text{enc}(r/8, 8)$
$P = \text{pad}(P, r)$
Let $P = P_0||P_1||\ldots||P_{|P|-1}$ with $P_i \in \mathbb{Z}_2^r$
$s = 0^{r+c}$
for $i = 0$ to $|P| - 1$ do
  $s = s \oplus (P_i||0^c)$
  $s = $ KECCAK-$f[r + c](s)$
end for
$Z =$ empty string
while output is requested do
  $Z = Z||\lfloor s \rfloor_r$
  $s = $ KECCAK-$f[r + c](s)$
end while
```

It makes use of the following functions:

- pad(M, n) returns a bit string obtained by appending to the bit string M a single 1 and the smallest number of zeroes such that the length is a multiple of n.
- enc(x, n) returns a string of n bits coding the integer x, from the least significant bit to the most significant bit, i.e., it returns the string $x_0, x_1, \ldots, x_{n-1}$ when $x = \sum_{i=0}^{n-1} x_i 2^i$.

This specifies KECCAK$[r, c, d]$ for any combination of $r > 0$ that is a multiple of 8 and c such that $r + c$ is a width supported by the KECCAK-f permutations and any integer value of d in the range $[0, 2^8 - 1]$.

The default value for d is 0 and the default values for r and c are 1024 and 576 respectively. Hence we have:

- KECCAK$[r, c] \triangleq$ KECCAK$[r, c, d = 0]$,
- KECCAK$[] \triangleq$ KECCAK$[r = 1024, c = 576]$.

Initially, the state has value 0^b, called the *root state*. The root state has a fixed value and shall never be considered as an input. This is crucial for the security of the sponge construction.

3 Security claim for the Keccak sponge functions

To simplify our claim, we first define the notion of *shortcut attack*:

Definition 1. *An attack on a specific sponge function is a* shortcut attack *if the expected success probability is higher when mounted on that specific function than it would be on a random oracle for a given attack workload. Here the workload of an attack on a sponge function shall be expressed in terms of the number of calls to the underlying function f, while the cost of an attack on a random oracle is the sum of the workload of all queries sent to the random oracle by the attacker. The workload depends on the padding and on the bitrate.*

Definition 2. *In the case of* Keccak$[r, c, d]$*, the workload of a single query with input length l and requesting n output bits is given by*

$$\left\lfloor \frac{8\lfloor \frac{l}{8} \rfloor + 24}{r} \right\rfloor + \left\lceil \frac{n}{r} \right\rceil \text{ calls.}$$

For each of the supported parameter values, we make a *flat sponge claim* [1].

Claim 1. *The expected success probability of any shortcut attack against* Keccak$[r, c, d]$ *with a workload equivalent to N calls to* Keccak-$f[r + c]$ *or its inverse shall be smaller than or equal to*

$$1 - \exp\left(-N(N+1)2^{-(c+1)}\right) .$$

We exclude here weaknesses *due to the mere fact that* Keccak-$f[r + c]$ *can be described compactly and can be efficiently executed, e.g., the so-called random oracle implementation impossibility.*

Note that the claimed capacity is equal to the capacity used by the sponge construction.

4 The Keccak candidates for SHA-3

Our candidates are:

1. SHA3-224: $\lfloor$Keccak$[r = 1152, c = 448, d = 28]\rfloor_{224}$
2. SHA3-256: $\lfloor$Keccak$[r = 1088, c = 512, d = 32]\rfloor_{256}$
3. SHA3-384: $\lfloor$Keccak$[r = 832, c = 768, d = 48]\rfloor_{384}$
4. SHA3-512: $\lfloor$Keccak$[r = 576, c = 1024, d = 64]\rfloor_{512}$

However, we recommend using Keccak[] (with default parameters), where the output is truncated by the user at any desired output length, and up to the security claim of Section 3. See also Section 5.2.

5 Conventions

5.1 Bit and byte numbering

In this section, we detail the mapping between the bits of the input to the Keccak sponge functions, and their representation in the SHA-3 API defined by NIST [2]. The bits of the input message M are numbered from $i = 0$ to $i = |M| - 1$. When the bits are gathered in bytes, it implies the equivalent numbering $i = i_{\text{bit}} + 8i_{\text{byte}}$ with $0 \leq i_{\text{bit}} < 8$.

In our internal convention, $i_{\text{bit}} = 0$ indicates the least significant bit (LSB) of a byte while $i_{\text{bit}} = 7$ indicates the most significant one (MSB). The NIST convention [2] is different: The bits of a byte are numbered from 0 for the MSB to 7 for the LSB. To be compatible with the API convention outside and with our convention inside, the following *formal* bit-reordering is performed on the input bit string M before it is processed:

- For all bytes that contain 8 bits, position $i_{\text{bit}}+8i_{\text{byte}}$ is mapped to position $7-i_{\text{bit}}+8i_{\text{byte}}$.
- For the last byte if it contains $p < 8$ bits, position $i_{\text{bit}} + 8i_{\text{byte}}$ is mapped to position $(p-1) - i_{\text{bit}} + 8i_{\text{byte}}$.

This mapping is bijective and does not affect the security.

In practice, the above operation cancels with the change of convention, so there is nothing to do, except:

- For the last byte if it contains $p < 8$ bits, the bits are shifted by $8-p$ positions towards the LSB (i.e., to the "right").

5.2 Extending the API for arbitrarily-long output

The Keccak[] sponge function can produce an arbitrarily-long bit string. To enable this, we have made the following additions to the API [2].

In the `Init` function, the `hashbitlen` parameter can be set to 224, 256, 384 or 512 to select the parameters as specified in Section 4, with a fixed output length. If the `hashbitlen` parameter is set to 0, the arbitrarily-long output mode is activated.

To use the arbitrarily-long output mode, the following sequence of calls needs to be made.

- First, `Init` is called with `hashbitlen=0`. This selects the Keccak[] sponge function.
- The input data is processed as usual with the `Update` function.
- The `Finalize` function indicates that the whole input message has been fetched (i.e., it switches the sponge to the squeezing phase [1]). This function has to be called, but the `hashval` parameter is ignored, as no output is generated yet. This parameter can safely be set to 0 (the null pointer).
- The output is then generated by calling the `Squeeze` function as many times as desired.

```
HashReturn Squeeze(hashState *state, BitSequence *output, DataLength outputLength);
```

- **Parameters:**
  - `state`: a structure that holds the `hashState` information
  - `output`: the storage for output to be returned
  - `outputLength`: the length, in bits, of the output to produce; must be a multiple of 8
- **Returns:**
  - `SUCCESS` – on success

5.3 Parts of the state

In this subsection, we wish to define names of parts of the Keccak-f state, as illustrated in Figure 1. This section is not, as such, needed to implement the Keccak sponge functions. Nevertheless, it may help use a common terminology when analyzing or describing properties of Keccak-f.
The one-dimensional parts are:

- A *row* is a set of 5 bits with constant y and z coordinates.
- A *column* is a set of 5 bits with constant x and z coordinates.
- A *lane* is a set of w bits with constant x and y coordinates.

The two-dimensional parts are:

- A *sheet* is a set of $5w$ bits with constant x coordinate.
- A *plane* is a set of $5w$ bits with constant y coordinate.
- A *slice* is a set of 25 bits with constant z coordinate.

6 Change log

- The number of rounds of Keccak-f has changed from $12 + \ell$ to $12 + 2\ell$.
- The capacity and bitrate of the four fixed-output-length candidates has been changed. Now for each of them the capacity is equal to twice the output length.

References

[1] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, *Sponge functions*, Ecrypt Hash Workshop 2007, May 2007, also available as public comment to NIST from `http://www.csrc.nist.gov/pki/HashWorkshop/Public_Comments/2007_May.html`.

[2] NIST, *ANSI C cryptographic API profile for SHA-3 candidate algorithm submissions, revision 5*, February 2008, available from `http://csrc.nist.gov/groups/ST/hash/sha-3/Submission_Reqs/crypto_API.html`.

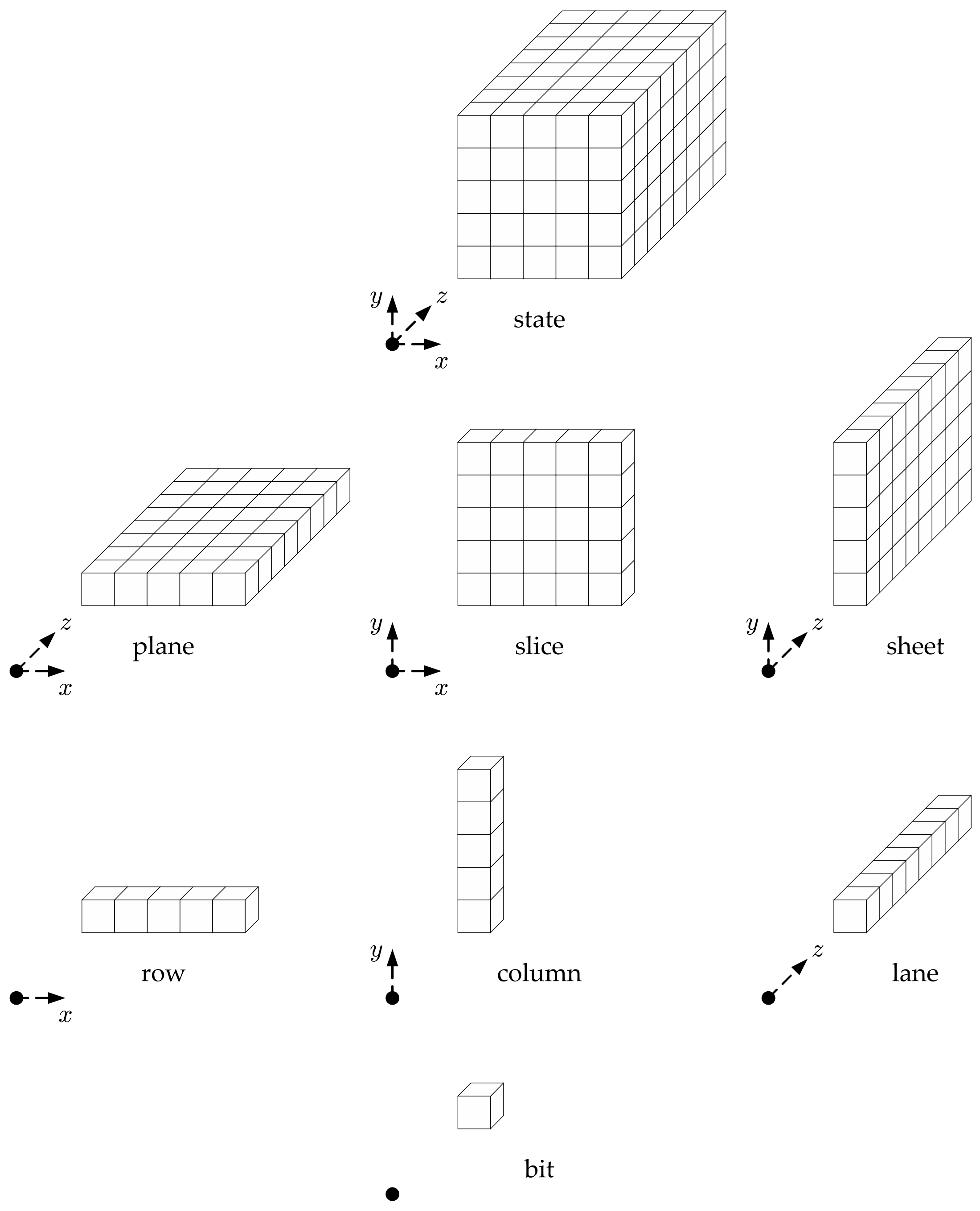

Figure 1: Naming conventions for parts of the KECCAK-f state