jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2018, VOL. 64, NO. 2, PP.159-166
Manuscript received February 13, 2018; revised March, 2018. DOI: 10.24425/119364

Improving security of lightweith SHA-3 against preimage attacks

Serhii Onopa and Zbigniew Kotulski

***Abstract*—In this article we describe the SHA-3 algorithm and its internal permutation in which potential weaknesses are hidden. The hash algorithm can be used for different purposes, such as pseudo-random bit sequences generator, key wrapping or one pass authentication, especially in weak devices (WSN, IoT, etc.). Analysis of the function showed that successful preimage attacks are possible for low round hashes, protection from which only works with increasing the number of rounds inside the function. When the hash function is used for building lightweight applications, it is necessary to apply a small number of rounds, which requires additional security measures. This article proposes a variant improved hash function protecting against preimage attacks, which occur on SHA-3. We suggest using an additional external randomness sources obtained from a lightweight PRNG or from application of the source data permutation.**

***Keywords*—hash function, SHA-3, Keccak, preimage attack, lightweight cryptography.**

I. Introduction

THE family of functions, called SHA-3 (Secure Hash Algorithm-3) is based on Keccak [1], the algorithm that NIST selected as the winner of the public SHA-3 Cryptographic Hash Algorithm Competition. The SHA-3 family consists of four hash functions and two extendable-output functions. SHA-3 constitutes a structure named the sponge construction; functions with this structure are called sponge functions.

The permutation is specified as an instance of a family of permutations, called KECCAK-f, to provide the flexibility to modify its size and security parameters in the development of any additional modes.

II. SHA-3 Algorithm

The Keccak hash function makes use of the sponge construction, as depicted in Fig. 1. Keccak has two main parameters: r (bitrate) and c (capacity). The sum of those two parameters makes the state size, which Keccak operates on. Different values for bitrate and capacity give trade-offs between speed and security. A higher bitrate gives a faster function at the expense of lower security. In the SHA -3 proposal, the state size is 1600 bits. The function uses two phase processing, see Fig. 1. The initial state is filled with zeros. In the first phase, absorbing, the state is processed by consecutive applications of the permutation Keccak-f. When all message blocks have been processed, the first phase is finished and the second begins, called squeezing phase. The first r bits of the state are returned as part of the output bits, interleaved with applications of the permutation Keccak-f. The second phase is finished after the desired length of the output has been produced, see [1].

Serhii Onopa and Zbigniew Kotulski are with Warsaw University of Technology, Poland (e-mail: zkotulsk@tele.pw.edu.pl, 2322314@gmail.com).

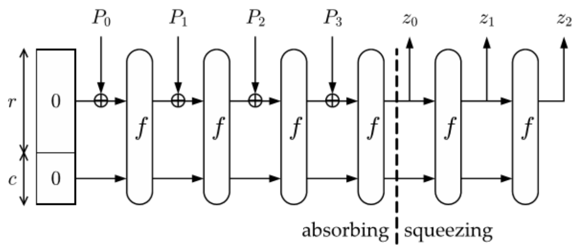

Fig.1. The sponge construction [4]

The value of the parameter c is equal to the hash length multiplied by 2. Keccak can operate on smaller states, for examples $b \in \{25, 50, 100, 200, 400, 800, 1600\}$, where b is the width of the permutation. The Keccak-f permutations are iterated construction consisting of a sequence of almost identical rounds. The number of rounds n_r depends on the permutation width and is given by $n_r = 12 + 12l$, where $2l = b/25$. This gives 24 rounds for Keccak-f[1600]. A Keccak-f round consists of a sequence of invertible steps, each step operating as an array $\boldsymbol{A}$ of 5×5 lanes; the length of lanes is $w \in \{1,2,4,8,16,32,64\}$ $(b = 24w)$. For Keccak-f[1600] with $c = 1024$ and $r = 576$, we have $r + c = 1600$.

In the rest of this section we present a specification of the Keccak round function (following the authors' specification and the FIPS Standard) underlying effect of each map constituting it on state bits transformations. Thus, the round is defined as:

$$Round(\boldsymbol{A}, RC) = \iota(\chi(\pi(\rho(\theta(\boldsymbol{A})))), RC).$$

The pseudo-code of a single round (with a round constant RC) starts in Fig.2, where the first step θ is presented (see [2]).

```
$Round(\boldsymbol{A}, RC)$ {
$\theta\ step$
$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4],$
$forall\ x\ in\ (0\ldots4)$
$D[x] = C[x-1] \oplus rot(C[x+1],1),$
$forall\ x\ in\ (0\ldots4)$
$A[x,y] = A[x,y] \oplus D[x],$
$forall\ (x,y)\ in\ (0\ldots4,0\ldots4)$
```

Fig.2. Pseudo-code of θ step.

The θ map is linear and aimed at diffusion; it is translation-invariant in all directions. Its effect can be described as follows: it adds to each bit $a[x][y][z]$ the bitwise sum of the parities of two columns: that of $a[x-1][\cdot][z]$ and that of $a[x+1][\cdot][z-1]$. A consequence of this is CP-kernel or column parity kernel. Without θ, the Keccak-f round function would not provide diffusion of any significance. Thanks to the interaction with χ (see Fig. 5) each bit at the input of a round potentially affects 31 bits at its output and each bit at the output of a round depends on 31 bits at its input. Note that without the translation of one of the two sheet parities this would only be 25 bits, see [1].

```
$\rho\ step$
$forall\ (x,y)\ in\ (0\ldots 4,0\ldots 4)$
$A[x,y]\ =\ rot(A[x,y],r[x,y]),$
$forall\ (x,y)\ in\ (0\ldots 4,0\ldots 4)$
$B[y,2*x+3*y]\ =\ A[x,y],$
```

Fig.3. Pseudo-code of ρ step

The map ρ (see Fig.3) consists of translations within the lanes aimed at providing inter-slice dispersion. Without it, diffusion between the slices would be very slow. It is translation-invariant in the z-direction. The inverse of ρ is the set of lane translations, where the constants are the same, but the direction is reversed.

```
$\pi\ step$
$forall\ (x,y)\ in\ (0\ldots 4,0\ldots 4)$
$B[y,2*x+3*y]\ =\ A[x,y]$
```

Fig.4. Pseudo-code of π step

The map π (see Fig.4) is a transposition of the lanes that provides dispersion aimed at long-term diffusion. Without it, Keccak-f would exhibit periodic trails of low weight. π operates in a linear way on the coordinates (x,y): the lane in position (x,y) goes to position $(x,y)M^T$, with $M = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$ being a 2×2 matrix with elements in $GF(5)$. It follows that the lane in the origin $(0,0)$ does not change position. As π operates on the slices independently, it is translation-invariant in the z-direction. The inverse of π is defined by M^{-1}. Many matrices could be used for π. In fact, the invertible $2\ \times 2$ matrices with elements in $GF(5)$ with the matrix multiplication form a group with 480 elements containing elements of order 1, 2, 3, 4, 5, 6, 8, 10, 12, 20 and 24. Each of these matrices defines a permutation on the 6 axes, and equivalently, on the 6 directions. Thanks to its linearity, the 5 positions on an axis are mapped to 5 positions on an axis (not necessarily the same). Similarly, the 5 positions that are on a line parallel to an axis, are mapped to 5 positions on a line parallel to the axis, see [1].

```
$\chi\ step$
$forall\ (x,y)\ in\ (0\ldots 4,0\ldots 4)$
$A[x,y]\ =\ B[x,y]\ \oplus\ ((\neg B[x+1,y])\wedge\ B[x+2,y])$
```

Fig.5. Pseudo-code of χ step

χ (see Fig.5) is the only nonlinear map in Keccak-f. Without it, the Keccak-f round function would be linear. It constitutes the parallel application of $5w$ S-boxes operating on 5-bit rows. χ is translation-invariant in all directions and has algebraic degree two. This has consequences for its differential propagation and correlation properties, see [1], [2], [3].

```
$\iota\ step$
$A[0,0]\ =\ A[0,0]\ \oplus\ RC$
$return\ A$ }
```

Fig.6. Pseudo-code of ι step

The map ι (see Fig.6) consists of the addition of round constants and is aimed at disrupting symmetry. Without it, the round function would be translation-invariant in the z direction and all rounds of Keccak-f would be equal making it subject to attacks exploiting symmetry, such as slide attacks. The number of active bit positions of the round constants, i.e., the bit positions in which the round constant can differ from 0, is $l+1$. As l increases, the round constants add more and more asymmetry. The bits of the round constants are different from round to round and are taken as the output of a maximum-length LFSR. The constants are only added in a single lane of the state. Because of this, the disruption diffuses through θ and χ to all lanes of the state after a single round, see [1].

All the operations on the indices are done modulo 5. $A[x,y]$ denotes a lane in that state and A denotes the complete permutations state array. The constants $r[x,y]$ are the rotations offsets, where RC are the round constants. $rot(W,m)$ is the usual bitwise rotation operation, moving bit at position i into position $i+m$ in the lane W. θ is a linear operation that provides diffusion to the hash state. ρ mixes bits of a lane using the rotation and π permutes lanes. χ is the only non-linear operation, ι calculates XOR of the round constant with the first lane, see [2].

III. Applications Based on the Duplex Construction

Cryptographic hash functions are fundamental components in different information security applications, such as digital signature generation and verification, key derivation, and pseudorandom bit generators. In this section we briefly present known applications, where Keccak-like functions, especially in the lightweight form, have been applied or could be applied for security solutions. The duplex construction can be used as a pseudo-random bit sequence generator [5], key wrapping tool [6] or one pass authenticated encryption algorithms [6]. Key wrapping can provide the assurance of integrity of data and the confidentiality of cryptographic keys or other data. Authenticated encryption used a duplex function by including a secret key in the input. If the duplex function behaves like a random oracle, the keyed duplex function behaves as a random function to anyone not knowing the key but having access to the duplex function.

One more important possible application of SHA-3 is hash cash, what in the future bitcoin might be applied. Because the SHA-1 is already broken and SHA-2 is of a similar design, that there was a problem with collisions for the older hash functions and SHA-3 fixes this. For hashcash-SHA-3 is that, there is some debate [7] within the NIST comments process on the proposal of weakening SHA-3's resistance to preimage attacks down to 128-bit (vs the full hash size, as with previous hashes). The aim is a small performance gain with the rationale that some hash-pluggable algorithms do not rely on full-length preimage resistance, which we propose to improve.

There are also cryptanalytic risks. A practical issue with switching to hashcash-SHA-3 is that it would invalidate all existing ASIC mining hardware and so is a change that would unlikely to be made.

One likely side-effect, however, would be that it would introduce more memory or pre-computation tradeoffs, which could make ASICs unprofitable or give advantages to people with large resources to do the pre-computations. Pre-computation advantages would be a motivation to replace the hash with SHA-3.

IV. Attacks on the Algorithm and Properties Exploited in Attacks

The six SHA-3 functions are designed to provide special properties, such as resistance to collisions, preimage, and second preimage attacks [8], [9], [10].

Criteria are laid by developers for permutation is to have no properties that can be exploited in a shortcut attack when being used in the sponge construction: bit-oriented structure; bitwise logical operations and fixed rotations, symmetry, parallelism, round degree 2, matryoshka structure, eggs in another basket, see [1].

Now consider the properties used in the attacks, which are described detail in the paper [11]. The preimage attack presented in [11] exploits two properties of θ[12]. The first one is the CP-kernel or the column parity kernel. CP-kernel allows to stay in very low Hamming weight difference state through the first round. The second property is the observation that either all 5 bits in each column are left unchanged or all 5 are flipped (the authors used in the 4-round preimage attack), see [1].

```
$for\ x = 0\ to\ 4\ do$
$C[x] = a[x,0]$
$for\ y = 1\ to\ 4\ do$
$C[x] = C[x] \oplus a[x,y]$
$end\ for$
$end\ for$
$for\ x = 0\ to\ 4\ do$
$D[x] = C[x-1] \oplus ROT(C[x+1],1)$
$for\ y = 0\ to\ 4\ do$
$A[x,y] = a[x,y] \oplus D[x]$
$end\ for$
$end\ for$
```

Fig.7. Properties of θ: transformation from the cube to a rectangle

Without θ, the Keccak-f round function would not provide diffusion of any significance. Its effect can be described as follows (using transformation of data presented in Fig.7): it adds to each bit $a[x][y][z]$ the bitwise sum of the parities of two columns: that of $a[x-1][\cdot][z]$ and that of $a[x+1][\cdot][z-1]$. A consequence of this is CP-kernel or column parity kernel.

The following describes the difference between the Keccak-f[1600] permutation and a purely random permutation. Further analyzing we trace XOR differences between corresponding bits from two states. 'Active' bit is a bit with difference 1 and 'inactive' with difference 0, see [11]. We need paths, which work with probability 1. We choose differential path starts with 4 active bits such as two of them are in a column and the other two active are in other column, for two reasons: the difference state is in the CP-kernel and after first θ there are still only 4 active bits; we need a family of differential paths from which we can construct other paths by changing the columns with active bits, see [11].

First three steps (θ, ρ, π) of a round are linear and a calculation, how differences change after these steps is simple. At the 4th step (χ step) we must calculate the non-linear function $y = x_1 \oplus x_2 x_3$. If x_2 or x_3 is an active or unknown bit, the output difference is unknown. Hence, this bit is marked as ‘unknown’. The last step is ι; it calculated XOR of the first lane with a round constant and does not affect non-linearity. According to this description, in Fig.8 is illustrated the differential path. Instead, 5×5 matrix of lanes there are 25 rows, each representing a single lane (64 bits). Unknown bits start to appear after χ in the 1st round. They are spread through the state in the 2nd round and their number is substantial in the end of the 2nd round. In the 3rd round χ changes useful bits to unknown bits. There are more differential paths like the one presented in Fig. 8. A family of such distinguishers is used in the preimage attack, see [11].

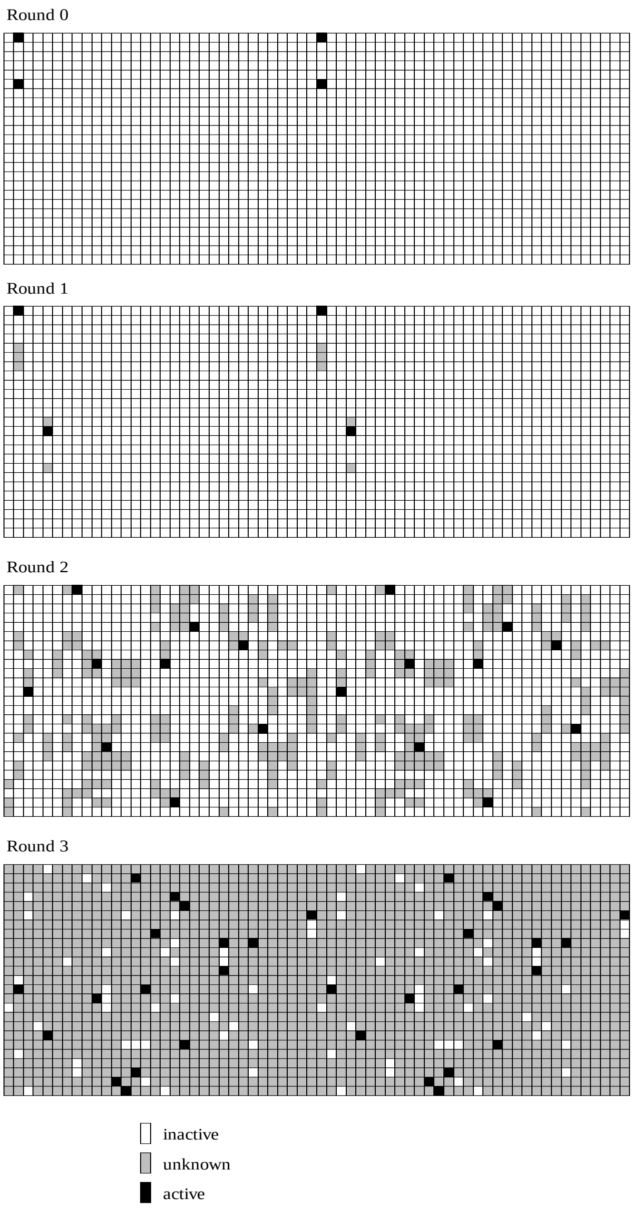

Fig.8. Differential path used in the preimage attack.

V. Additional Protection Against Attack

As it is mentioned, in the attacks presented above the properties of θ have been exploited. In this paper we propose to make additional changes after the operation θ as a countermeasure to make impossible knowing all differences in each column. The improved full round function is being constructed with adding logical operation XOR with some binary block after θ step, according to CP-kernel in θ permutation. In the proposed operation, we can use a sequence of pseudo-random binary blocks obtained from an external source, see Fig. 9. In this section we consider three possible solutions.

Fig. 9 illustrates Algorithm 1 for the case, where we use the addition operation after θ; specifically, we use operation XOR. In this scheme symbols $P_0, P_1, \dots P_n$ are used to denote the output binary blocks (of appropriate length) of some external Pseudo-Random Number Generator (PRNG) or HMAC. Application of such an additional operation protects the hash algorithm against attacks in a manner described in the papers [13] and [14]. Such a solution is the simplest one but to satisfy our expectations, it requires application of an external PRNG of good quality and of low computational overhead.

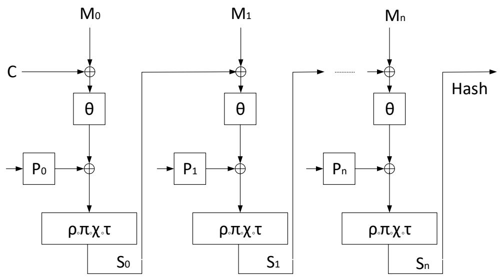

Fig.9. Algorithm 1: using external PRNG.

In the next Algorithm 2, illustrated in Fig. 10, we use a similar approach, which is used to remove the vulnerability. Here we additionally insert an internal operation, the purpose of which is pseudo-random selecting a range of data and then changing the data after the operation θ via XOR operation. In Algorithm 1 we generate a sequence of texts $P_0, P_1, \dots P_n$ from the sequence of messages $M_0||M_1|| \dots || M_n$ by a permutation: a fixed one or pseudo-random, obtained from an external source of randomness (some PRNG). In this case we save a calculation overhead, because at each pass of the algorithm we need only random permutation of n numbers instead of n binary blocks of the length depending on the size of the blocks of SHA-3.

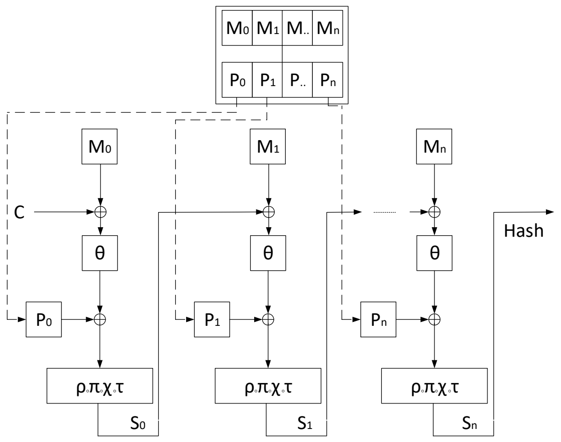

Fig. 10. Algorithm 2: Improved hash function with the addition of a pseudo randomness chosen from a data block

Our last proposition is Algorithm 3, illustrated in Fig. 11. In this algorithm the original SHA-3 round is modified with application of a pseudo-random generator like those used in the Petra-2 hash function, see [15]. This approach also makes impossible the attack of the preimage type, proposed and considered earlier in papers [11], [13], [16], [17]. Perta-2 is a Merkle-Damgard type hash function using in its compression function a simple PRNG. The aim of this generator is to calculate a permutation of 16 numbers: $1,2,\dots,16$, which are stored in table $Z = \{i_1, i_2, \dots, i_{16}\}$, see [11], which defines indexes of mixed subblocks of the internal state data block. Thus, in our Algorithm 3, we propose in each round iteration of SHA-3, after θ operation, XOR a pseudo-random binary block obtained from the input data block by permutation of its subblocks according to the table Z. An advantage of such a solution, like in Algorithm 2, is that it has low computational overhead for randomness generation. Another advantage is that it does not need any externat PRNG.

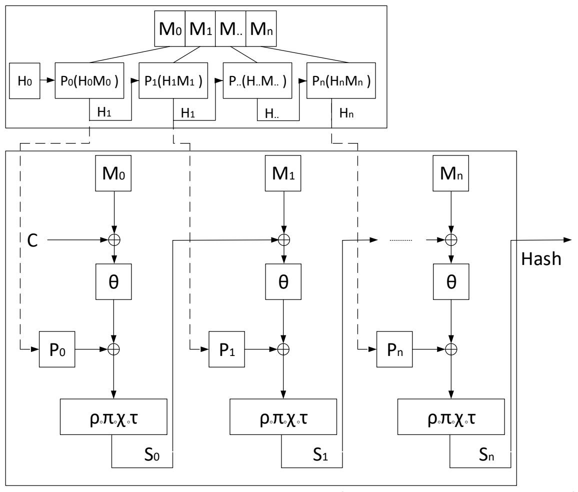

Fig. 11. Algorithm 3: Improved hash function with application of the Petra-2-like PRNG.

Validation of the proposed modified algorithms requires checking at least their computational effectiveness and verifying randomness of their outputs. Statistical properties of hash outputs can be checked out with NIST suite of tests [18]. However, in this paper we restrict our analysis to analyzing possible statistical dependencies of the output of pure SHA-3 rounds and the rounds modified with adding a binary sequence after the operation θ. Numerical results of such an analysis is presented in Section VI.

VI. Experimental Results and Analysis

Based on the Algorithms 1, 2 and 3 described in this article and on the scheme, in which it was modified the SHA-3 standard algorithm, several generated sets of data have been tested and analyzed. The idea of tests is the following. We introduced the block of data after the θ step to XOR it with θ output. The block P_i has been changing from 0 to 15. The purpose of calculations was to show, how the obtained hash value (or, more precisely, the round output) changes for three different hash inputs, changing P_i blocks and for growing number of rounds. First, we used the Hamming weight to measure a “randomness” of the output data changing with P_i changes. Next, we calculated the Hamming distance of input and output data to estimate changes of "correlation" between data blocks after hash transformation, for growing number of rounds. The full number of rounds that have been analyzed is 24 but, in this section, we show only the most representative results.

As the hash input block, we used three variants of the buffer: a sequence of zeros, named "empty" in further analysis, a binary representation of four numbers "0", "1", "2" and "3", named "0123", and a binary representation of four letters "w", "o", "r" and "d", named "word". Below in Fig.12a, Fig.12b, Fig.12c, Fig.12d, Fig.12e an evolution of the Hamming weight of the hash changed in the manner described above, for the buffer "empty" and for different number of rounds: 1, 2, 3, 5, and 24. In Fig.13a, Fig.13b, Fig.13c, Fig.13d, Fig.13e analogous changes of the Hamming weight are presented for the buffer "0123" and for different number of rounds: 1, 2, 3, 5, and 24. Finally, we also investigated the buffer "word" in the identical experiment, but the graphs obtained do not contain specific trends or other important dependencies and therefore they are not shown in plots.

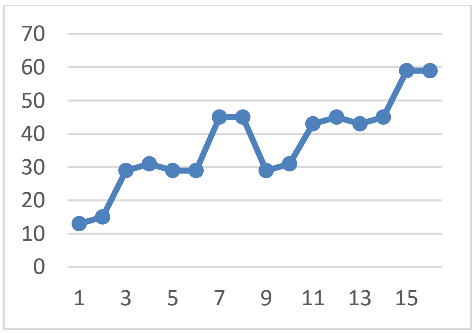

Fig. 12a. Hamming weight. 1 round. Buffer – "empty".

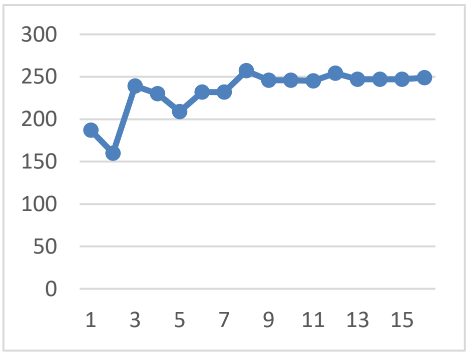

Fig. 12b. Hamming weight. 2 rounds. Buffer – "empty".

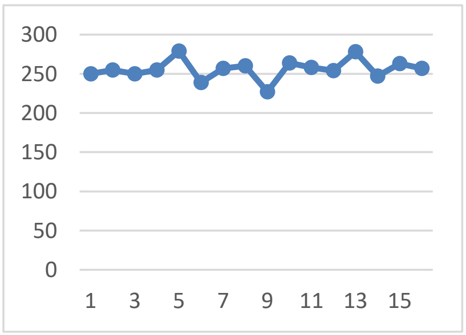

Fig. 12c. Hamming weight. 3 rounds. Buffer – "empty".

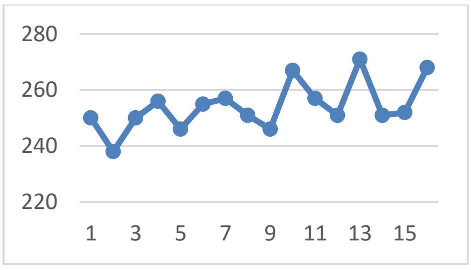

Fig. 12d. Hamming weight. 5 rounds. Buffer – "empty".

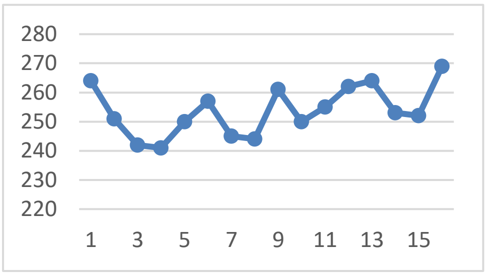

Fig. 12e. Hamming weight. 24 rounds. Buffer – "empty".

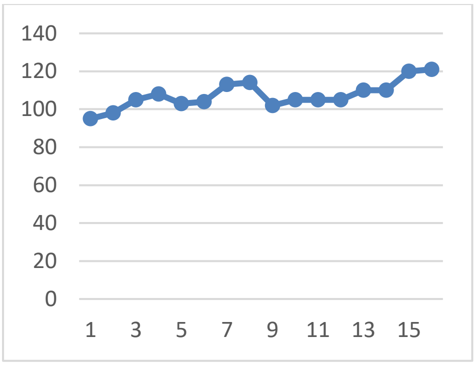

Fig. 13a. Hamming weight. 1 round. Buffer – "0123".

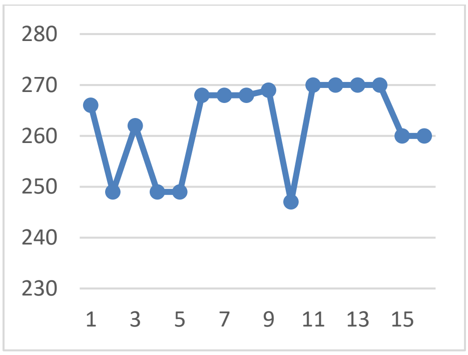

Fig. 13b. Hamming weight. 2 rounds. Buffer – "0123".

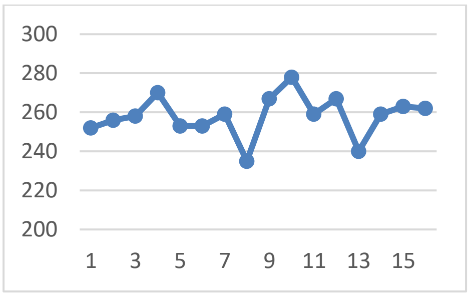

Fig. 13c. Hamming weight. 3 rounds. Buffer – "0123".

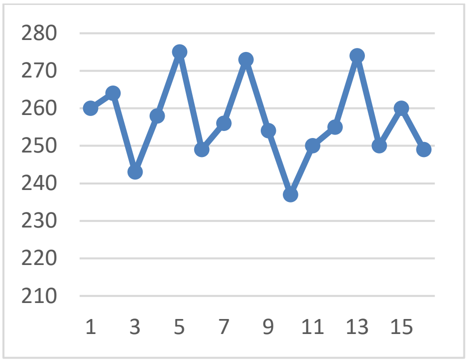
Fig. 13d. Hamming weight. 5 rounds. Buffer – “0123”.

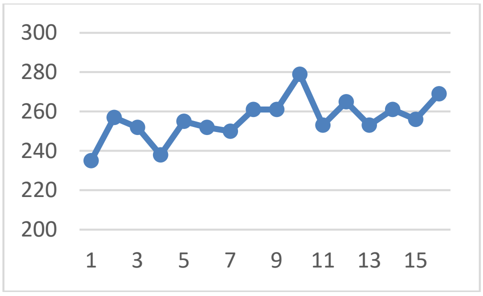
Fig. 13e. Hamming weight. 24 rounds. Buffer – “0123”.

The next two series of figures contain results presenting changes of the Hamming distance for our texts. In Fig.14a, Fig.14b, Fig.14c, Fig.14d, Fig.14e, an evolution of the Hamming distance between the original SHA-3 round and the hash changed in the manner described above, for the buffer “empty” and for different number of rounds: 1, 2, 3, 5, and 24. The next series of figures, Fig.15a, Fig.15b, Fig.15c, Fig.15d, Fig.15e, present the analogous Hamming distance for the buffer “0123” for different number of rounds: 1, 2, 3, 5, and 24. Similarly to the above described, we also investigated the Hamming distance for the buffer “word” in an identical experiment, but the graphs obtained do not contain significant tendencies or dependencies and therefore they are not shown.

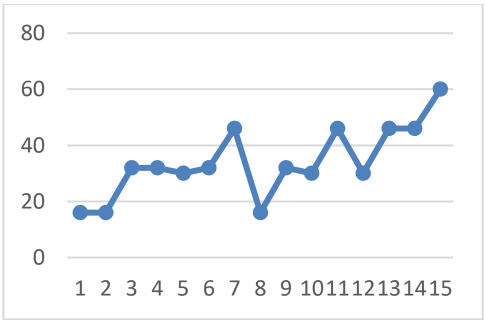
Fig. 14a. Hamming distance. 1 round. Buffer – “empty”.

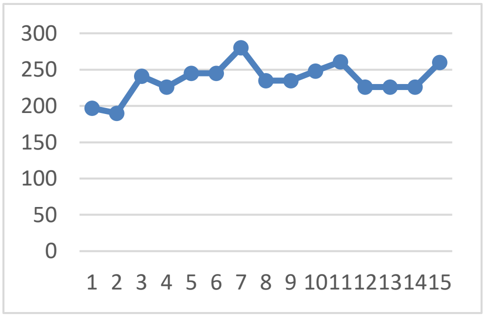
Fig. 14b. Hamming distance. 2 rounds. Buffer – “empty”

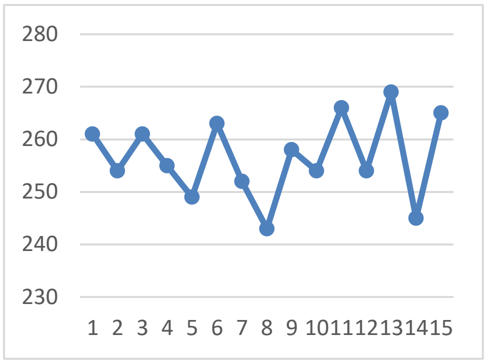
Fig. 14c. Hamming distance. 3 rounds. Buffer – “empty”.

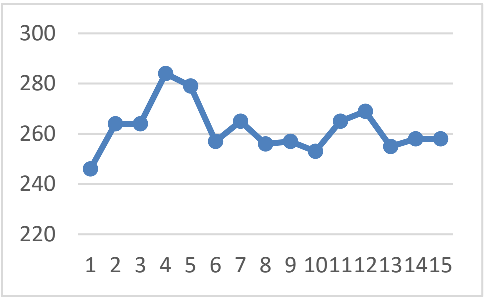
Fig. 14d. Hamming distance. 5 rounds. Buffer – “empty”.

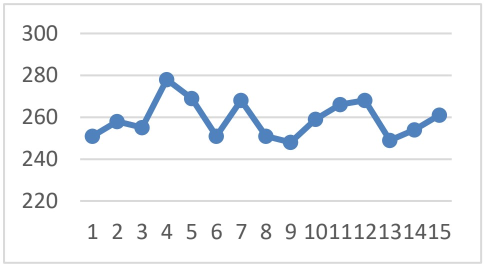
Fig. 14e. Hamming distance. 24 rounds. Buffer – “empty”.

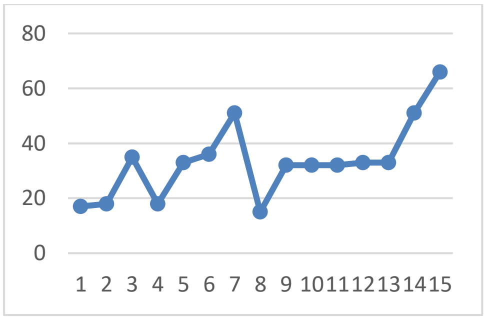
Fig. 15a. Hamming distance. 1 round. Buffer – “0123”.

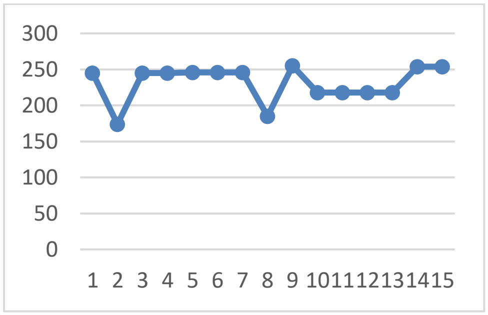
Fig. 15b. Hamming distance. 2 rounds. Buffer – “0123”.

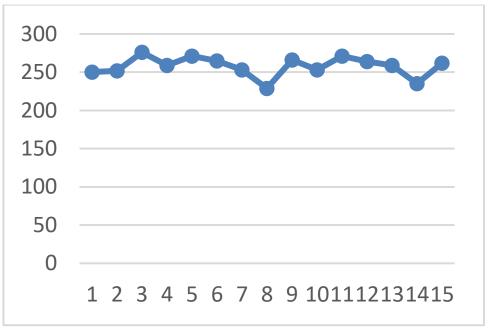

Fig. 15c. Hamming distance. 3 rounds. Buffer – “0123”.

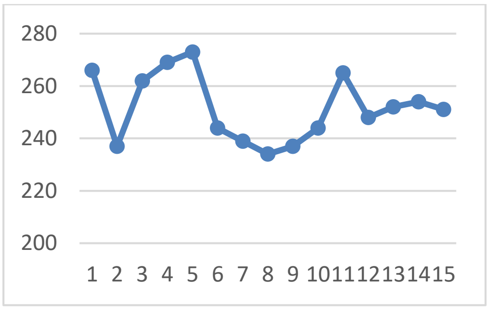

Fig. 15d. Hamming distance. 5 rounds. Buffer – “0123”.

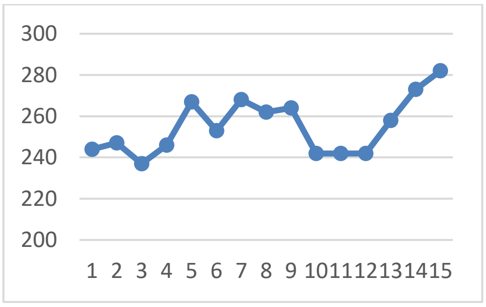

Fig. 15e. Hamming distance. 24 rounds. Buffer – “0123”.

In each experiment we considered 512-bit blocks, so such binary blocks can be considered as “random” if the number of “0” bits and “1” bits is approximately equal or do not differ too much. So, such an equivalence number of bits is 256. Results presented in the plots could be summarized as:

- For large number of rounds (e.g., 24) the hash output is pseudorandom even if the input buffer is very specific, independently if we modify the round function with a pseudorandom inclusion.
- For a small number of rounds adding extra randomness after the operation θ can have significant influence for some specific input blocks. It has small effect if the input block is pseudorandom (this results from out experiment with the buffer “word”).
- There is a border number of iterations of a round (it could be 3 or 4) where the effect of included internal randomness rapidly decreases.

Summary and Conclusions

Based on SHA-3 standard and the permutation described in it, is proposed a new type of the construction, where to the original SHA-3 structure an additional randomness source is included. A new approach should provide additional strength for preimage and collision attacks. This scheme is adding a pseudo-random data block after diffusion operation in the round permutation and reduces the ability to predict variables after the first round. In this work we analyzed SHA-3, which is based on Keccak, the algorithm that NIST selected as a winner of the public SHA-3 Cryptographic Hash Algorithm Competition [19]. Next, we described parameters of the Keccak hash function, which make it possible to manage speed and security of the sponge construction. These parameters affect the size of permutation, which is the basis of the sponge construction. The function uses two phase processing: absorbing and squeezing phase. We referred to the pseudo-code of a single round of Keccak, which contains this permutation. Security of the sponge construction also depends of the number of rounds. One round consists of five operations: $\theta, \rho, \pi, \chi, \iota$. This pseudo-code describes in detail the effect of these operations on the properties of the sponge construction, such as: diffusion, inter-slice dispersion, long term diffusion, nonlinear mapping and disrupting a symmetry. Cryptographic hash functions are fundamental components in a variety of information security application, such as digital signatures generation and verification, key derivation and pseudorandom bit generation. We briefly considered the following literature results. The duplex construction can be used as a pseudo-random bit sequence generator [6], key wrapping [13] or components of one pass authenticated encryption algorithms [13], [15]. The six SHA-3 functions are designed to provide special properties, such as resistance to collision, preimage, and second preimage attacks, c.f. [1], [11], [17], [20]. Design criteria laid by the developers for the permutation are to have no properties that can be exploited in a shortcut attack, when being used in the sponge construction: a bit-oriented structure; bitwise logical operations and fixed rotations, symmetry, parallelism, round degree 2, matryoshka structure, eggs in another basket, etc., see [1], [17], [21]. Then, in our considerations we followed an analysis of a permutation, which helps to evaluate the weak points of the Keccak function, which is exactly inside this permutation. We considered the properties used in the attacks and highlighted those used in the attack, such as CP-kernel, and observed the behavior of the state's bits, what is confirmed by the references to the work, see [12], [16]. In the preimage attack, used by [22], there were exploited two properties of θ. The first one is the CP-kernel or column parity kernel. The CP-kernel allows to stay in a very low Hamming weight difference to state through the first round. The second property is the observation that either all 5 bits in each column are left unchanged or all 5 are flipped (the authors used them in the 4-round preimage attack, see [2], [22], [14]). As mentioned above, in the attacks there were used the properties θ. Therefore, to prevent such attacks we proposed to make additional block data changes after the operation θ. Consequently, it would make impossible to know all the differences in each data column. We proposed to improve the full round of Keccak in three algorithms: Algorithm 1, 2, and 3. A new approach should provide additional strength for the preimage and the collision attacks.

The main observation of the paper was the effect of the number of rounds used in the sponge function. If the number is small, then successfully implemented attacks are possible. They are effective, and they could be used in some low-performance devices, where SHA-3 with only a small number of rounds can be applied. Also, looking to the future, it must be assumed the rapid development of algorithms that allow to use this type of attacks with many rounds. Having all the above, we proposed building and developing additional protection against the preimage attack. These countermeasures would remove the weak spots inside the round function, detailed CP-kernel and further.

We proposed inserting an additional operation after the permutation θ, using three versions of modifications used in this scheme. First, we used after the first permutation XOR of an output of some pseudorandom bitstream generator (possibly a secret key, what results with a new HMAC construction). In the second case we proposed using a pseudorandom permutation of the input data blocks. The third scheme assumed using elements of an additional algorithm, which is Petra-2-like pseudorandom permutation to provide additional randomness to the input data.

Tests and analyses were performed by changing the last bits after the operation θ. The above considerations shown the dependences and specifically the increase of the Hamming weights and the Hamming distances for buffers: “empty” and “0123”, to the random equivalent values. Analogous characteristics for the buffer “word” remained unchanged. The obtained data confirm the theoretical assumption of an increase in resistance to attacks using the CP-kernel for small number of rounds used in SHA-3. Using such changes in the low round number SHA-3 and the algorithms proposed will ensure the stability of the hash algorithm in low-performance devices and can also prevent the future development of such attacks.

REFERENCES

[1] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak sponge function family main document, http: //keccak.noekeon.org/Keccak-main-2.1.pdf.

[2] FIPS 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, csrc.nist.gov/publications/drafts/fips202/fips_202_ draft.pdf.

[3] Najjar M., Stokłosa J., The nonlinearity of homogenous Boolean functions and the design of strong cryptographic algorithms. Burakowski W., Wieczorek A. (eds.), Regional Conference on Military Communication and Information Systems. Zegrze 1999, Vol. 2, 71–76.

[4] Chengxin Qu, Seberry J., Pieprzyk J., Homogenous bent functions (preprint), University of Wollongong, NSW, Australia 1998.G. Eason, B. Noble, and I.N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955. (references)

[5] Bertoni G., Daemen J., Peeters M., and Assche G., Sponge-based pseudo-random number generators, CHES 2010: Cryptographic Hardware and Embedded Systems, CHES 2010 p.33-47.

[6] Borowski M., Gliwa R., Rozwój algorytmów uwierzytelnionego szyfrowania, www.wil.waw.pl/art_prac/2014/PTiWT_8-9_14_3.pdf.

[7] https://en.bitcoin.it/wiki/Hashcash

[8] Dinur I., Security Evaluation of SHA-3, https://www.cryptrec.go.jp/ estimation/techrep_id2402.pdf.

[9] Dinur I., P Morawiecki P., J Pieprzyk J., Srebrny M., Straus M., Cube Attacks and Cube-attack-like Cryptanalysis on the Round-reduced Keccak Sponge Function, https://eprint.iacr.org/2014/736.

[10] Morawiecki P., Malicious Keccak, https://eprint.iacr.org/2015/1085.

[11] Morawiecki P., Pieprzyk J., Srebrny M., and Straus M.: Preimage attacks on the round-reduced Keccak with the aid of differential cryptanalysis, https://eprint.iacr.org/2013/561.

[12] Dinur I., Morawiecki P., J Pieprzyk J., Srebrny M., Straus M., Practical Complexity Cube Attacks on Round-Reduced Keccak Sponge Function, https://eprint.iacr.org/2014/259.pdf

[13] Morawiecki P., Pieprzyk J and Srebrny M. Rotational cryptanalysis of round-reduced Keccak. IACR Cryptology ePrint Archive, 20 12, pp. 546-562.

[14] Chang D. , Kumar A., Morawiecki P., Sanadhya S., 1st and 2nd Preimage Attacks on 7, 8 and 9 Rounds of Keccak-224,256,384,512, http://csrc.nist.gov/groups/ST/hash/sha-3/Aug2014/documents/chang_ paper_sha3_2014_workshop.pdf.

[15] Najjar M., Stokłosa J., Petra-2 cryptographic hash function, NATO Regional Conference on Military Communications and Information Systems 2001, Zegrze, 2001, vol. I, 317–320.

[16] Lathrop J., Cube Attacks on Cryptographic Hash Functions, http://scholarworks.rit.edu/cgi/viewcontent.cgi?article=1653&context=theses.

[17] Sekar G., Bhattacharya S., Practical (Second) Preimage Attacks on TCS SHA-3, https://eprint.iacr.org/2013/150.pdf.

[18] NIST SP 800-22rev1a, A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications, April 27, 2010.

[19] https://csrc.nist.gov/projects/hash-functions/sha-3-project

[20] Dinur I., and Shami A, Cube Attacks on Tweakable Black Box Polynomials, https://eprint.iacr.org/2008/385.pdf

[21] Borowski M., Cryptographic Applications of the Duplex Construction, https://journals.umcs.pl/ai/article/download/3389/2583.

[22] Sekar G., Bhattacharya S., Practical (Second) Preimage Attacks on TCS SHA-3, https://eprint.iacr.org/2013/150.pdf.