The Sponge Construction as a Source of Secure Cryptographic Primitives

Mariusz Borowski

Cryptology Division, Military Communication Institute,
Zegrze, Poland

Abstract: Most of modern cryptography primitives have no provably secure constructions. Their safety is defined on the basis of well-known in the given time cryptanalytic attacks. Moreover, the asymptotic nature of cryptographic definitions (and definitions of complexity theory in general) does not let us say anything about how hard it is to break a given cryptographic primitive for keys of a certain fixed length. Sponge constructions equipped with one ideal permutation and appropriate security parameters are suitable for building provably secure cryptographic primitives. The cryptographic primitives based on sponge and duplex constructions cover most symmetric crypto operations.
Keywords: the sponge construction, sponge functions, the duplex construction, authenticated encryption, key wrapping

I. Introduction

Assured security is the desirable feature of modern cryptography. Sponge constructions equipped with only one ideal permutation and appropriate security parameters can be used to provably secure cryptographic primitives building. The constructions because of its arbitrarily long input and output sizes allow building various primitives such as a hash function, a stream cipher or a message authentication code (MAC). In some implementations the input is short (e.g., a key and a nonce) while the output is long (e.g., a key stream). In other applications, the opposite occurs, where the input is long (e.g., a message to hash) and the output is short (e.g., a digest or a MAC).

Another set of applications takes advantage of the duplex construction which is closely related to the sponge construction and whose security can be shown to be equivalent. The duplex construction permits the alternation of input and output blocks at the same rate as the sponge construction. This allows one to implement an efficient reseedable pseudo random bit sequence generation and an one pass authenticated encryption scheme free from patent limitations.

II. The sponge construction

The sponge construction (Fig. 1) is a simple iterated construction for building a function with variable-length input and arbitrary output length based on fixed-length transformation or permutation *f* operating on a fixed number *b* of bits, and a sponge-complaint padding rule "pad". Here *b* is called the width. The sponge construction operates on a state of $b = r + c$ bits. The value *r* is called the bitrate and the value *c* the capacity. Different values for bitrate and capacity give the trade-off between speed and security. The higher bitrate gives the faster cryptographic function that is less secure.

It is important that the last *c* bits of the *b*-bit state are never directly affected by the input blocks and are never output during the digest producing. The capacity *c*, the most important security parameter, determines the attainable security level of the constructions, as proven in the chapter III.

First, all the bits of the state are initialized to zero. The input message is padded and cut into blocks of *r* bits. The sponge construction has two-phase processing. In the first phase (also called the absorbing phase), *r*-bit input message blocks are xored with the first *r* bits of the state, interleaved with applications of the function *f* (a random permutation or a random transformation). The absorbing phase is finished when all message blocks have been processed. In the second phase (also called the squeezing phase), the first *r* bits of the state are returned as the part of the output bits, interleaved with applications of the function *f*. The squeezing phase is finished after the designed length of output digest has been produced.

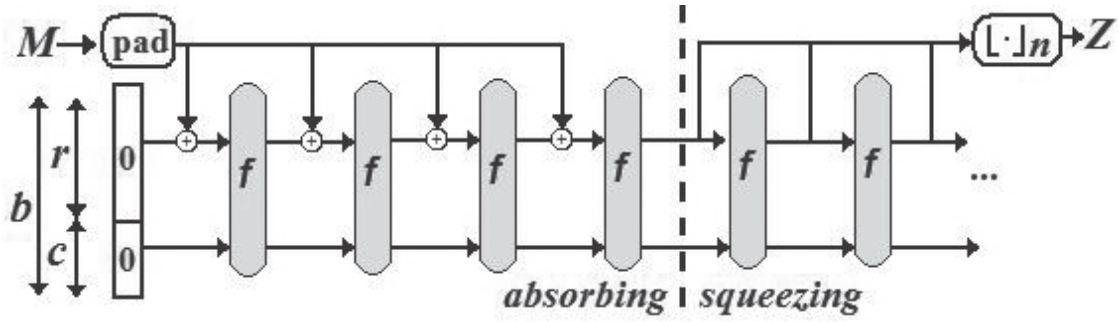

Figure 1. The sponge construction

III. Security of the sponge construction

The cryptanalytic attacks on the sponge construction can be divided into two types. The first type is showing a non-random behavior, weakness in an internal permutation or a random transformation. Such attack leads to built distinguishers on such functions (permutations or transformations). The second type is the attack on the core security properties of the whole cryptographic function based on the sponge construction (a preimage attack and a collision attack). An attack on a sponge function is a generic attack if it does not exploit specific properties of *f* (an internal permutation or a random transformation).

In the ideal model, a compressing function F (either on fixed or arbitrary input) that uses one or more underlying building blocks is viewed insecure if there exist a successful information-theoretic adversary that has query access to the idealized underlying primitives of F. The complexity of the attack is measured by the number of queries q to the primitive made by an adversary. Let's consider preimage, second preimage and collision resistance. For each of these three notations, with Adv_F^{atk}, where $atk \in \{pre, sec, col\}$, J denote the maximum advantage of an adversary to break the function F under the security notation *atk*. The advantage is the probability function taken over all random choices of the underlying primitives, and the maximum is taken over all adversaries that make at most q queries to their oracles. A random oracle is a function which provides a random output for each new query. The indifferentiability of a compressing function F, Adv_F^{pro}, maximized over all distinguishers making at most q queries of maximal length $K \geq 0$ blocks to their oracles. An indifferentiability bound guarantees security of a compressing function F against specific attacks. It has been shown by Bart Preneel that

$$Adv_F^{atk} \leq PR_{RO}^{atk} + Adv_F^{pro} \tag{1}$$

for any security notation *atk*, where PR_{RO}^{atk} denotes the success probability of a generic attack against F under attacks *atk* and *RO* is an ideal function with the same domain and range as F. If a compressing function F outputs a bit string of length n, one expects to find collisions with high probability after approximately $2^{n/2}$ queries (due to the birthday attack). Similarly, (second) preimages can be founded with high probability after approximately 2^n queries. Moreover, finding second preimages is provably harder than finding collisions, and similar for preimages (depending on the definition of F). The advantage in differentiating that the sponge construction from a random oracle is upper bounded by

$$N^2 \cdot 2^{-(c+2)}, \tag{2}$$

with N the number of calls to the underlying transformation or permutation and the capacity c, resulting in an expected time complexity of $N \sim 2^{c/2}$. This implies that with respect to generic attacks of complexity N, the sponge construction offers the same level of security as a random oracle, as long as (2) is negligible. Once N approaches $2^{c/2}$ this is no longer the case and the indifferentiability does not give any guarantees. This is due to the existence of inner collisions in a sponge function.

Guido Bertoni, Joan Deamen, Michael Peeters, Gilles Van Assche in [3] shown the sponge type hash function Keccak, that was selected by NIST as the winner of the SHA-3 competition in October 2012. The constructors adopted the sponge construction and built an underlying permutation f that should not have any structural distinguishers. The capacity c determines the claimed level of security, and one can trade claimed security for speed by increasing the capacity c and decreasing the bitrate r accordingly, or vice-versa. Keccak is a member of the sponge function family [2].

The sponge type hash function Keccak is proven indifferentiable from a random oracle up to bound

$$\Theta((Kq)^2 / 2^{c=2n}) \tag{3}$$

if the underlying permutation is assumed to be ideal. Using (1) and (3) this indifferentiability bound renders an optimal collision resistance bound for Keccak,

$$Adv^{col}_{Keccak} = \Theta(q^2 / 2^n), \tag{4}$$

as well optimal preimage and second preimage resistance

$$Adv^{pre}_{Keccak} = Adv^{sec}_{Keccak} = \Theta(q / 2^n). \tag{5}$$

The most successful result (in terms of a number of rounds) on the sponge type Keccak's permutation is the zero-sum distinguisher [10, 1]. However, the complexity of the distinguishers is very high. For example, the zero-sum distinguisher for all 24 rounds of Keccak has the complexity of 2^{1579} (instead of the theoretical complexity 2^{1600}) [10]. The result do not lead to any attacks on the Keccak hash function. The second type generic attacks have academic complexity bounded up to 8 rounds [8, 9, 14].

IV. The duplex construction

The duplex construction (Fig. 2), like the sponge construction, uses a fixed length transformation or permutation *f*, a padding rule "pad" to build a cryptographic scheme. Unlike a sponge function that is stateless in between calls, the duplex constructions results in an object that accepts calls that take an input string and return an output string that depends on all inputs received so far. We call an instance of the duplex construction a *duplex object* denoted by *D*. The call to a specific duplex object *D* is marked by its name *D* and a dot.

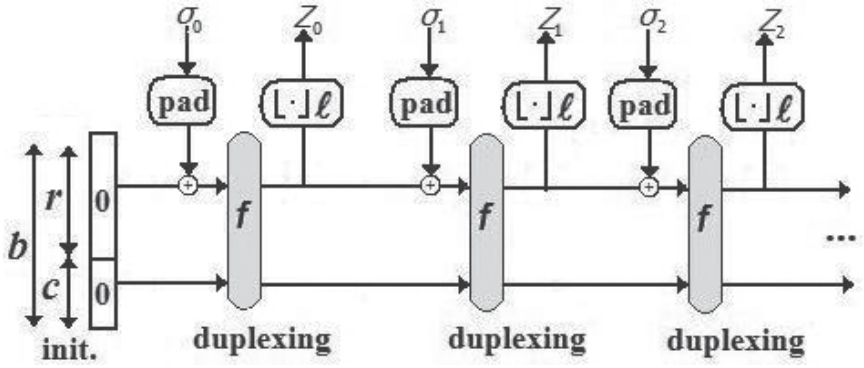

Figure 2. The duplex construction

A duplex object *D* has a state of *b* bits. Upon initialization all the bits of the state are set to zero. From then on one can send to it *D*.duplexing (σ, ℓ) calls, with σ an input string and ℓ the requested number of bits. The maximum number of bits

ℓ one can request is r and the input string should be short enough such that after padding it results in a single r-bit block.

A fundamental property of the duplex construction (called *duplexing-sponge lemma*) is that the output of a call to a duplex object can be obtained by evaluating a sponge function with the same parameters to the input (r and c) constructed from all previous inputs to the duplex object.

The lemma states that the output of a duplexing call is the output of a sponge function with an input $\sigma_0||\text{pad}_0||\sigma_1||\text{pad}_1||...||\ \sigma_i||\text{pad}_i$ and from this input that exact sequence $\sigma_0, \sigma_1, ..., \sigma_i$. As such, the duplex construction is as secure as the sponge construction with the same parameters. In particular, the duplex construction inherits its upper bound on the random oracle differentiating advantage, where the input to the random oracle is the sequence of inputs to the duplexing calls since the initialization.

V. Usage of the sponge construction

The sponge construction covers:

- plain hashing;
- salted hashing;
- a mask generation function
- message authentication codes;
- stream encryption.

A. Salted hashing

A sponge function can be used as an n-bit hash function by simple truncation of its output. If the hash function is to be used in the context of the randomized hashing a random value (the salt) can be prepended to the message. Salted hashing based on a sponge function is similar to plain hashing, but the salt is added as the first block or blocks of the combined message. We know that for the sponge construction there are no generic attack with complexity of order below $2^{c/2}$. The lower bound for the expected complexity for generating a collision is

$$min(2^{n/2}, 2^{c/2}) \tag{6}$$

and for generating a second preimage is

$$min(2^{n}, 2^{c/2}). \tag{7}$$

Therefore, if $n < c/2$, randomization based on the sponge construction increases the strength against signature forgery due to generic attacks against the hash function due to the attacks from $2^{n/2}$ to 2^{n}. If the capacity c is between n and $2n$, the strength increases is from $2^{n/2}$ to $2^{c/2}$.

Salted hashing shown in Fig. 3 can be used for password storage and password verification.

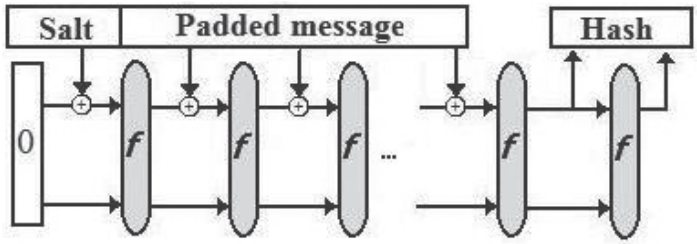

Figure 3. Salted hashing

B. A mask generation function

A mask generation function (MGF) also called a key derivation function is a pseudo-random function taking a bit string of any length as input and returning a new bit string of desired bit length. In theoretical models, MGFs are treated as random oracles. In practice mask generation functions are often based on a secure cryptographic hash functions. The sponge construction because of its arbitrarily long input and output sizes readily can serve a secure mask generation function as shown in Fig. 4. Key derivation functions are usually used in cryptographic protocols (e.g., SSL, TLS).

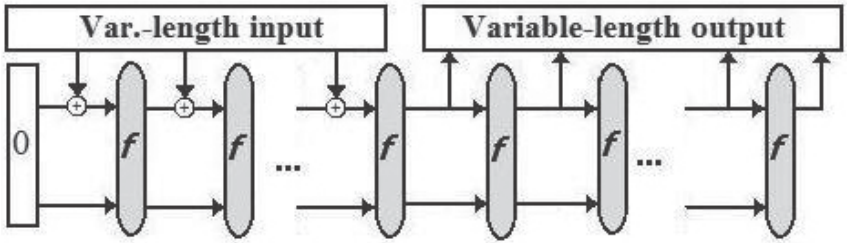

Figure 4. Mask generation function

C. Message authenticated codes

A message authenticated code (MAC) takes as a input a key, an initial value (IV) and a message. The sponge construction because of its arbitrarily long input and the output truncated to desired number n of bits ($n \leq c/2$) readily can serve as a message authenticated code scheme shown in Fig. 5. The length of a generated MAC code could be significantly longer than in the case of a MAC scheme based on a block cipher.

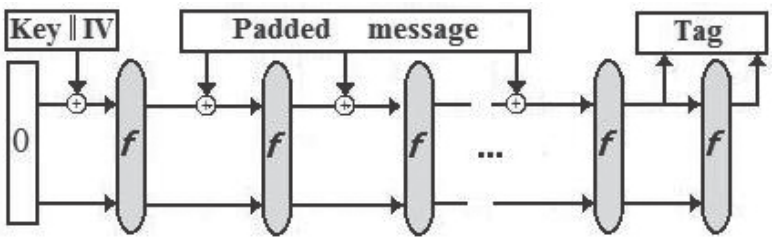

Figure 5. Message authenticated code scheme

D. Stream encryption

A sponge function can also serve as a stream cipher. One can input the key and some initial value and then get key stream in the squeezing phase. The stream cipher can be used in two modes, when we need:

- a long output stream per IV (similar to OFB mode);
- a short output stream per IV (similar to CTR mode).

Stream encryption (Fig. 6) is an example of a sponge function turned into a keyed function by including in the input a secret key. If the sponge function behaves like a random oracle, the keyed sponge function conducts as a random function to anyone not knowing the key but having access to the sponge function. Let's consider security of a sponge function used in conjunction with a key, for example in the case of our stream cipher scheme (Fig. 6). The dependences also set for message authentication codes, authenticated encryption, reseedable pseudo-random sequence generators. It was shown in [4] that he advantage in distinguishing a keyed sponge from a random oracle is upper bounded by

$$max\{(M^2/2+2M\cdot N)\cdot 2^{-c}, N\cdot 2^{-|K|})\} \tag{8}$$

where M is the data complexity, i.e. the amount of access to the keyed scheme, N is the number of queries to the underlying transformation (or permutation), and $|K|$ is the length of the key. In typical case if $M << 2^{c/2}$ time complexity is much smaller and it carries out about

$$min(2^{c-1}/M, 2^{|K|}). \tag{9}$$

The key is related with the capacity with following dependence:

$$|K| + 1 + \log_2 M \leq c. \tag{10}$$

This allows decreasing the capacity c (and thus the permutation width) for a given security level or achieving a higher security level for a given capacity. The shown bound for keyed applications allows usage of very fast lightweight sponges, especially on platforms with limited resources.

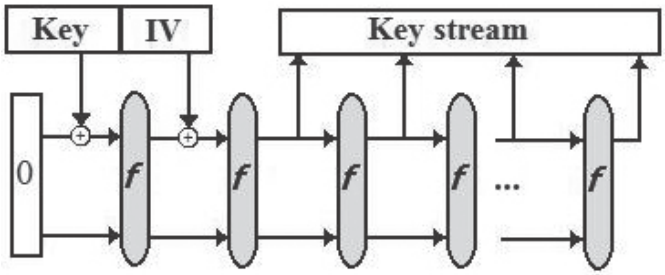

Figure 6. Stream encryption

VI. Usage of the duplex construction

The duplex construction covers:

- one pass authenticated encryption;
- key wrapping;
- residable pseudo-random bit sequence generation.

A. One pass authenticated encryption

Authenticated encryption (AE) has been extensively studied in the last ten years. Block cipher modes are a popular way to provide simultaneously both integrity and confidentiality. One pass authenticated encryption schemes based on block ciphers (OCB, IAPM, IACBC) are patent encumbered. The length of a authenticating tag is bounded by a used block cipher in the schemes. The maximal length of the tag in block cipher AE schemes is 128 bits.

The one pass AE scheme shown in Fig. 7 uses the duplex construction. Upon initialization it loads the key K. From then on one can send request to it for wrapping or unwrapping data. The key stream blocks used for encryption and the tags depend on the key K and the data sent in all previous requests. Key stream sequences give no information on tags and vice versa as they are obtained by call to different duplex instances.

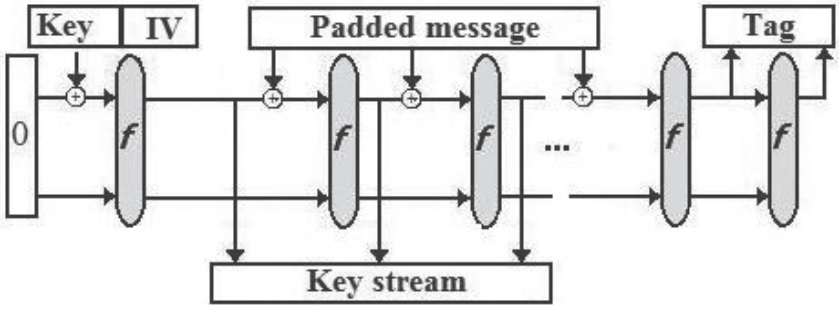

Figure 7. One pass authenticated encryption

The one pass AE scheme based on the duplex construction satisfies the security requirements of key recovery (probability of finding the key with trying N keys K is not above $N{\cdot}2^{-|K|}$), tag forgery ($2^{-|T|}$) and plaintext recovery ($N{\cdot}2^{-|K|}$) if the used sponge is secure.

The authenticated encryption based on the duplex construction has some advantages:

- it is one pass and requires only one fixed-length permutation;
- it supports the alternation of strings that require authenticated encryption and strings that only require authentication;
- it has a strong security bound against generic attacks with a simple proof, that relies on the bound of the *RO* differentiating advantage of the sponge

construction (or the security of a keyed sponge function) and on the sponge-duplexing lemma;
- it is flexible as the bitrate r can be freely chosen as long as the capacity c is larger than some lower bound;
- the length of the authenticating tag is bounded by $c/2$, considerably more than in a block cipher AE case.

B. Key wrapping

Key wrapping is the example of the practical use of one pass authenticated encryption shown in Fig. 7. In such scheme a payload key is wrapped with a key-encrypting key (*KEK*). The *Key* is equal to the *KEK* and the data body is the payload key value. If each key is associated to a unique identifier it is sufficient to include the identifier of the payload key in the header and two different payload keys will be enciphered with different streams.

Key wrapping is very important in key management systems [5]. It helps to protect the secrecy and integrity of cryptographic keys in transport or storage.

C. A residable pseudo-random bit sequence generator

The duplex construction can readily be used as a residable pseudo random bit sequence generator shown in Fig. 8. Seeding material can be fed via σ inputs in it *D*.duplexing () call and the responses can be used as a pseudo-random bits.

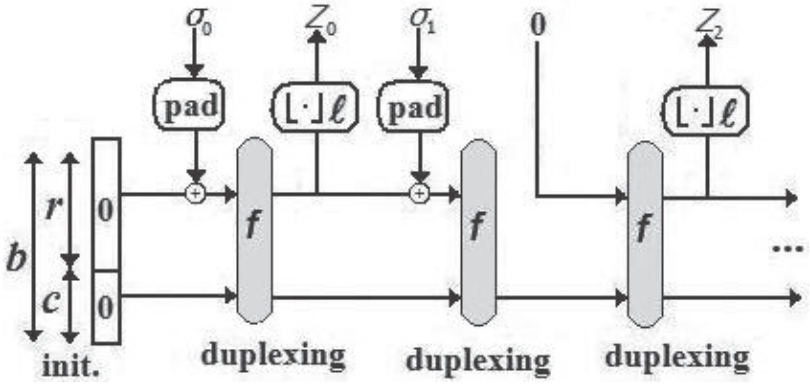

Figure 8. Residasble pseudo-random bit sequence generator

The only limitation of this is that the user must split his seeding material in strings of at most ρ bits and at most r bits can be requested in a single call.

Binary random sequences have numerous applications in many fields of science and technology. The most important ones are applied in such fields as cryptography, statistics, numerical computation, stochastic simulations using the Monte Carlo method, and many others. Military Communication Institute (MCI) developed a family of hardware random bit generators, the first in the nineties. The generators

can generate random sequences with an output rate 115.2 kbit/s, 8 Mbit/s up to 100 Mbit/s and they were certified by the Polish national security authority according to "The Protection of Classified Information Act" and can be used in cryptographic systems up to "TOP SECRET" level [6]. As a scientific tool the SGCL-100M generator (the generator with an output rate 100 Mbit/s) can be used in advanced researches in many fields of science and technology. Since the generator is a quite complex and costly device [10] with a very high output rate it can be assumed that it could be used as a source for random sequence servers in R&D centers.

In 2012 MCI decided to built a cheap but very fast pseudo-random sequences generator based on the duplex construction and our old and slow random bit generator as a source of seeding material. The generator applies a sponge function f to the sequences of values $seed+1$, $seed+2$, ... The output sequence is $f(seed+1)$, $f(seed+2)$, ... Because of the property of forward security, it is necessary to keep only a few bit of the output values $f(seed+i)$ in order to remove possible correlation between successive values. Ideally, secrets required in cryptographic algorithms and protocols should be generated with a true random generator. A residable pseudo random bit sequence generator, based on the Keccak-like sponge function and a slow hardware random bit generator can pass the tests for randomness of the NIST Statistical Test Suite and MCI battery tests [13, 12]. The generator is very efficient and can produce binary random sequences with the potential throughput (amount of data per unit time) higher than 150 Mbit/s in the relatively cheap way. It will be able to produce a little more than 1.5T bytes per day and act as a practically "infinite" source of such pseudo-random sequences with very good statistical quality.

VII. Summary

The cryptographic primitives based on the sponge and duplex constructions can be used for protection classified and unclassified information. The primitives are provably secure. The creation of the ideal permutation or the random transformation is the key matter. Guido Bertoni with his team shown as such safe permutations can be built. The effectiveness of the method was confirmed by third-party cryptanalysis.

The cryptographic primitives based on sponge and duplex constructions showed in the paper cover common applications of symmetric cryptography. The constructions are very flexible and other applications can be readily built. It is possible to first fix the capacity c such that $c/2$ is at least the desired security level and then choose the remaining parameters. It is very important that for the sponge and duplex constructions there are no generic attacks with complexity of order below $2^{c/2}$.

The keyed sponge functions are used for MAC computation, stream encryption, one pass authenticated encryption schemes free from patent limitations and reseedable pseudo-random sequence generators. Usage of the key in the applica-

tions allow decreasing the capacity (and thus the permutation width) for a given security level or achieving a higher security level for a given capacity. The bound for keyed applications permits usage of very fast lightweight sponges, especially on platforms with limited resources.

REFERENCES

[1] J.P. Aumasson, W. Meier, "Zero-sum distinguishers for reduced Keccak-*f* and for the core functions of Luffa and Hamsi", CHES 2009, http://131002.net/data/papers/AM09.pdf.
[2] G. Bertoni, J. Deamen, M. Peeters, G. Van Assche, "Cryptographic sponge functions", January 2011, http://sponge.noekeon.org.
[3] G. Bertoni, J. Deamen, M. Peeters, G. Van Assche, "The Keccak reference Version 3.0", STMicroelectronics 2011, http://sponge.noekeon.org.
[4] G. Bertoni, J. Deamen, M. Peeters, G. Van Assche, "On the security of keyed sponge constructions", Symetric Key Encryption Workshop, 2011.
[5] M. Borowski, M. Leśniewicz, R. Wicik, M. Grzonkowski, "Generation of random keys for cryptographic systems", Annales UMCS Informatica AI XII, 3 (2012).
[6] M. Borowski, R. Wicik, "A one-time cipher machine for Polish Army," Military Communication Conference," Prague, 2008.
[7] M. Duan, X. Lai, "Improved zero-sum distinguisher for full round Keccak-f permutation", 2011, http://eprint.iacr.org/2011/023.pdf.
[8] I. Dinur, O. Dunkelman, A. Shamir, "New attacks on Keccak-224 and Keccak-256", FSE 2012, pp.462-461, LNCS 6147, Springer-Verlag, 2012.
[9] I. Dinur, O. Dunkelman, A. Shamir, Self-differential cryptanalysis of up to 5 rounds of SHA-3, http://eprint.iacr.org/2012/672.pdf.
[10] M. Leśniewicz, Sprzętowa generacja losowych ciągów binarnych. Hardware generation of binary random sequences, WAT, Warszawa 2009, ISBN 978-83-61486-31-2.
[10] JM. Naya-Plasenica, A. Rock, W. Meier, "Practical analysis of reduced-round Keccak", INDOCRYPT 2011, pp.236-254, LNCS 7107, Springer-Verlag, 2011.
[11] W. Schindler, W. Killmann, "Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications," Workshop on Cryptographic Hardware and Embedded Systems CHES,2002, Springer-Verlag Berlin Heidelberg 2003.
[12] R. Wicik, M. Borowski, "Randomness testing of some random and pseudorandom sequences," Military Communication Conference, Prague, 2008.