An Enhanced Sponge Function (ESP)

Magdy Saeb

Computer Engineering Department,
Arab Academy for Science, Technology & Maritime Transport,
Alexandria, Egypt.

mail@magdysaeb.net

Abstract: *The Enhanced Sponge Function (ESP) is constructed using two permutation functions. This is in contrast to using only one function as presented in the seminal paper on sponge functions by G. Bertoni et al. The use of only one function carries traces of Merkle-Damgard structure. The shuffling between two permuting functions is performed using a message-dependent pseudo-randomly generated key. Using two pseudo-randomly shuffled functions, providing a non-deterministic data path, increases the uncertainty or entropy in the sponge function output bits. We argue that the proposed construction, without any loss of generality, closely emulates a random oracle more than the original sponge structure using only one function.*
Keywords: *Sponge Function, Hash function.*

1. Introduction

Sponge functions are generalizations of hash functions. A hash function output is a fixed number of bits while a sponge function output is a variable number of bits. The sponge function is constructed using an iterated construction providing a variable-length inputs and outputs employing fixed-length permutations. A pseudo-random sponge function emulates a random oracle. However, there is an exception here which is the finite memory effect. There are a large number of applications of this type of functions in security proofs and other practical applications.

The Enhanced Sponge Function (ESP), as the name implies, is an enhancement of the sponge function proposed by G. Bertoni et al., in their seminal paper [Bert07]. This enhancement includes two major additions to the design proposed in the aforementioned paper. These additions are:

- Using two permuting functions instead of one function that are pseudo-randomly selected based on a pseudo-randomly generated key
- The key is pseudo-randomly generated utilizing the message itself and two correctors; Von Neumann corrector and the XOR corrector.

In section 2, for the purpose of self-containment, we provide a tutorial discussion, based on [Bert07, Bert11, Bert12, Andre12], that sheds some light on sponge function construction, their applications and other related details and notations. In section 3, we discuss our design enhancements and provide some justification of these enhancements. In section 4, we show that the security proofs issued for the sponge function of [Bert11] apply also to the Enhanced Sponge Function without any changes. Finally, we provide a summary and our conclusions.

2. Background on Sponge Functions

The sponge functions are constructed using an iterated successive use of a fixed-length permutation function (f) to build a function (F) with variable-length inputs and outputs as shown in [Bert11]. In these references, b, r and c are the width, bit rate and capacity respectively, where

$$b = r + c$$

The process starts with the zero-bit initialization and the message is padded and divided into a number of r-size blocks. There are two phases of the sponge function process; the absorbing phase and the squeezing phase. In the absorbing phase, the r-bit blocks are XORed into the first r bits of the state, interleaved with applications of the function f. In the squeezing phase, the first (r) bits of the

Submitted 7/5/2012, Reviewed 7/20/2012

state are returned as output blocks, interleaved with applications of the function f. The last c bits of the state are not directly affected by the input blocks and are not yielded during the squeezing phase.

This type of construction provides a generalized cryptographic primitive that can be used in various types of security applications.
Potential applications of the sponge functions include:

- Pseudo-random bit sequence generation
- Key derivation
- Stream Cipher-based encryption
- Message Authentication Code

In this construction a single permuting function (f) is utilized which, in a sense, reminds us of the classical Merkle-Damgard construction of hash functions. We hypothesize that using a single permuting function (f) in cascade can be hardly assumed to provide the required level of uncertainty in the output bits.

3. The Proposed ESP Structure

The construction of the proposed Enhanced Sponge Function uses two permutation functions (f_0 and f_1). The idea is to provide pseudo-random shuffling of the message blocks to increase the level of uncertainty or entropy of the output bits. As shown in Figure 1, data is redirected to f_0 and f_1 in the first stage using two DMUXs. In the next stage which is repeated in the following stages two MUXs are used to shuffle the data pseudo-randomly between these two permutation functions. In a software implementation these MUXs and DMUXs are nothing but function calls. The key is generated using the message bits, before padding, and two correctors; Von Neumann corrector [Dave00] and an XOR corrector. The Von Neumann corrector acts successively on two consecutive bits of the message. If these two bits are similar they are skipped. However, if they are different either the left or right bit is chosen as part of the pseudo-randomly generated key. To increase the randomness in the key bits, we apply the XOR operation between successive bits resulting from the Von Neumann corrector operation. There are two major data paths in this construction; the green dotted path and the red solid path. The green dotted path is adopted when the key bit is zero, while the red solid path is adopted when the key bit is 1. The key bits are inverted in the middle of various iterations to provide better shuffling. It allows no bias in the use of either f_0 or f_1. This inverter is clearly shown in Figure 1. The construction is then repeated as many times as required.

Thus the basic structure of G. Bertoni et al. Construction is preserved. Accordingly, Bertoni's security proofs are also preserved. Still, we have appreciably added to the security of the sponge function by increasing the entropy of the resulting output bits. We conjecture that this construction more closely emulates a random oracle since the data path is non-deterministic. This construction moves another step away from the standard Merkle-Damgard construction where the path is deterministic and applying one function in a pipeline-like structure.

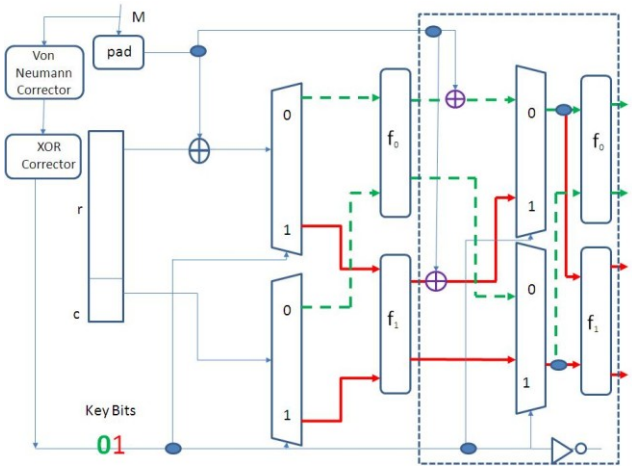

Figure 1. The structure of the proposed Enhanced Sponge Function (ESP)

Figure 1 demonstrates only the enhancement applied to the sponge function. The iterated structure is constructed by repeating the leftmost part, including the MUXs and f_0 and f_1, of the Figure in a cascaded form similar to the structure found in [Bert07]. This part to be repeated is shown in the dotted rectangle.

4. Security Proof

In the previously discussed proposed enhanced construction, we have added another dimension to the permutation operation. One can call this permuting "an Outer Permuting". This is in contrast to the use of a single permuting function (f) where one can call this operation as "Inner Permuting". All the basic elements of Bertoni's construction are well-preserved. Therefore, we conjecture that we have added to the security of the this basic construction. In addition, the security proofs are relevant to the Enhanced Sponge Function. In the

security proofs provided in [Bert11], the only feature that sets a random sponge apart from a random oracle is the existence of inner collisions. In their paper, the authors have provided an upper bound on the success probability of an attacker distinguishing between a random oracle (RO) and a sponge function output. This bound covers an adversary that has access to f and $f^{-1}$and allows replacing a random oracle by a random sponge in any application. In our proposed structure, the attacker has to have an access to f_0, f_0^{-1}, f_1, f_1^{-1} in order to launch her attack. Another important feature of a sponge function is the probability that an adversary, with no direct access to the permutation given by f_0, f_1, is able to distinguish between a random oracle and the sponge function output. The cost N of a query is the total number of calls to F it would yield if X = S [F]. This is based on the fact that an adversary who is presented with a system X that is either S [F] or RO. The a priori probability of X being either RO or S [F] is 1/2. In this analysis, Bertoni et al. [Bert11], have reached the conclusion that the distinguishing advantage of the adversary, represented by an algorithm A, is given by:

$$\text{Adv}(A) = \Pr\{A[S[F]] = 1\} - \Pr\{A[RO] = 1\}$$

The authors continued with their analysis to provide an upper bound of the success probabilities of generic attacks. We conjecture that this proof, developed only for one permutation function f, is also applicable to the case of two permutation functions. As a matter of fact, the adversary will have a higher cost of queries to achieve a successful attack in this case. The ideal case is when Adv (A) = 0 where the detection probabilities are equal. Considering this Adv (A) measure, one notices that it may yield a negative probability. To avoid this situation, Bertoni's used the absolute value of the equation. However, we can visualize this situation, where Pr {A [S [F]] = 1} < Pr {A [RO] = 1}, as a defective distinguishing algorithm A that returns the RO output as less random than that of a sponge function. The algorithm A depends on the set of queries sent and the guessing rule. Therefore, it is conceivable, for an algorithm A, that the set of queries or the guessing rule or both may render the situation where negative probability or Adv (A) arises.

Summary and Conclusions

In this work, we have presented an enhanced version of a sponge function. This enhancement is based on the following modifications:

- Utilizing two permutation functions instead of only one function
- Pseudo-random shuffling these two permutation functions using a message-dependent pseudo-randomly generated key
- The key is generated using the message bits and two correctors; Von Neumann and XOR correctors.

The security proof is directly related to the security proof given in the seminal paper by Bertoni et al. on sponge functions. We have also discussed that the case where Adv (A) which is basically a probability can be negative depending on the set of queries or the guessing rule or both used in the algorithm A. We conjecture that the enhanced sponge function ESP is more secure since the adversary has to possess f_0, f_0^{-1}, f_1, f_1^{-1} that are pseudo-randomly shuffled instead of f and f^{-1}.

References

[Bert07] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, *Sponge Functions*, in ECRYPT Hash Workshop, 2007.

[Bert11] G. Bertoni, J. Daemen, M. Peters, G. V. Assche, Cryptographic sponge functions, (Report), http://sponge.noekeon.org/

[Bert12] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, The Sponge Functions Corner, http://sponge.noekeon.org/, accessed June, 2012.

[Andre12] Elena Andreeva, Bart Mennink, Bart Preneel, "The Parazoa family: generalizing the sponge hash functions," Int. J. Inf. Sec. 11(3): 149-165, 2012.

[Dave00] R. Davies, Hardware Random Number Generators, http://www.robertnz.net/hwrng.htm, accessed June, 2012.

Magdy Saeb received the BSEE., School of Engineering, Cairo University, in 1974, the MSEE, and Ph.D. Degrees in Electrical & Computer Engineering, University of California, Irvine, in 1981 and 1985, respectively. He was with Kaiser Aerospace and Electronics, Irvine California, and The Atomic Energy Establishment, Anshas, Egypt. Currently, he is a professor in the Department of Computer Engineering, Arab Academy for Science, Technology & Maritime Transport, Alexandria, Egypt; He was on-leave working as a principal researcher in the Malaysian Institute of Microelectronic Systems (MIMOS). His current research interests include Cryptography, FPGA Implementations of Cryptography and Steganography Data Security Techniques, Encryption Processors, Mobile Agent Security. www.magdysaeb.net.