Hardware authentication based on PUFs and SHA-3 2nd round candidates

Susana Eiroa, Iluminada Baturone
Depto. Electrónica y Electromagnetismo, Univ. de Sevilla, IMSE-CNM-CSIC, Seville, Spain
{eiroa, lumi }@imse-cnm.csic.es

Abstract— **Security features are getting a growing interest in microelectronics. Not only entities have to authenticate in the context of a high secure communication but also the hardware employed has to be trusted. Silicon Physical Unclonable Functions (PUFs) or Physical Random Functions, which exploits manufacturing process variations in integrated circuits, have been used to authenticate the hardware in which they are included and, based on them, several cryptographic protocols have been reported. This paper describes the hardware implementation of a symmetric-key authentication protocol in which a PUF is one of the relevant blocks. The second relevant block is a SHA-3 2nd round candidate, a Secure Hash Algorithm (in particular Keccak), which has been proposed to replace the SHA-2 functions that have been broken no long time ago. Implementation details are discussed in the case of Xilinx FPGAs.**

***Index Terms*— Hash function, lightweight protocol, physically unclonable function, true random number generator.**

I. Introduction

SEVERAL protocols have been proposed based on symmetric and asymmetric cryptography that use PUFs in order to improve security features. Most of them are *ad-hoc* structures that depend on the application, such as the symmetric key protocol for IP protection proposed in [1], and the structure for off-line RFID authentication described in [2], which uses elliptic curve-based asymmetric cryptography.

Symmetric key constructions offer simpler structures than asymmetric ones. This fact, together with using PUFs, which provide tamper resistant authentication and protection against reply attacks at low cost, allow the implementation of secure symmetric key constructions with low hardware resources, what is known as lightweight cryptography. This is the approach followed by the HB-PUF protocol in [3] and the proposal in [4].

Our proposal is to implement a challenge–response Diffie-Helman authentication protocol [5] with the following building blocks so as to obtain a lightweight and secure solution: (a) ring oscillator-based PUFs (Physical Unclonable Functions) for dynamic key generation, (b) ring oscillators working as TRNG (true random number generators) to generate the protocol nonces, and (c) SHA3 2nd round candidates of the NIST (concretely Keccak) for hash function. Security relies in the secrecy of the key (which is generated on the fly), randomness of the nonces, and use of hash functions.

The paper is structured as follows. Section II gives an overview of the protocol characteristics. Section III reviews, firstly, basic principles of PUFs and focuses, secondly, on structures for secret key generation using fuzzy extractor and ring oscillator PUFs. Section IV describes ring oscillator configurations to construct TRNGs while Section V describes the features of the hash function used. All implementation results are summarized in Section VI. Finally, conclusions are given in Section VII.

II. Authentication Protocol

Figure 1 shows the generic scheme of our proposal based on Diffie-Helman protocol. To ensure privacy and integrity, the protocol uses a symmetric key scheme and hashes the responses of each side before being sent. The working principle is as follows:

- Each extreme generates a token (nonce) that must be sent to the extreme under authentication. Nonces will be generated by TRNGs to avoid replay and dictionary attacks.
- The receiver extreme generates a response that depends on both nonces and the common secret key. In order to avoid side channel attacks, key is dynamically generated by a PUF.
- Responses are hashed before being sent to ensure secrecy of the key. The response received by each

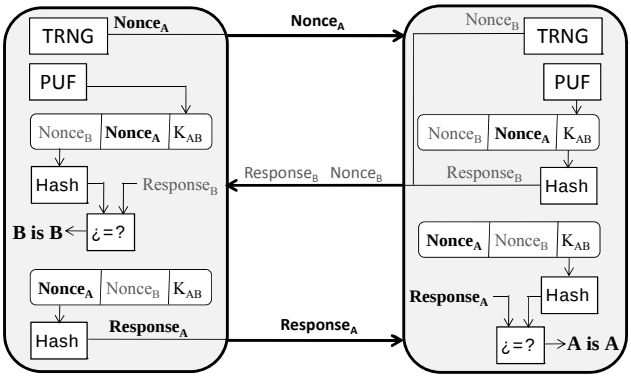

Figure 1. Authentication protocol

Manuscript received July 31, 2010. This work was partially supported by Junta de Andalucía under the Project P08-TIC-03674, by the European Community through the MOBY-DIC Project FP7-INFSO-ICT-248858 (www.mobydic-project.eu), and by Spanish Ministerio de Ciencia y Tecnología under the Projects TEC2008-04920 and DPI2008-03847.

S. Eiroa, and I. Baturone are with the University of Seville and Microelectronics Institute of Seville (CNM-CSIC), SPAIN (phone: +34-954-466-666; fax: +34-954-466-600; e-mail: {eiroa, lumi}@imse-cnm.csic.es).

extreme is compared with the value calculated so as to decide authentication.

III. PUFs FOR SECRET KEY GENERATION

PUFs, which were introduced by Pappu in [6], are random functions driven by parametric properties of physical components that map a set of challenges to a set of responses. The mapping function can only be evaluated with the physical system. Pappu defined a PUF as a physical object with special properties of unclonability (both physically and mathematically) and unpredictability [6]. Ideally, difference between two responses of different PUFs to the same challenge show Hamming variation of μ=50% with σ=0% (PUF uniqueness property), and difference between two responses of the same PUF to the same challenge, shows ideally Hamming variation of μ=0% with σ=0% (PUF reliability property).

PUFs can be divided into two groups: those that require special fabrication steps (such as *Active Coating* [7] and *Optical structures* [8]) and those denominated *Silicon* PUFs [9], which can be implemented in standard FPGAs and ASICs. The latter are cheaper and easier to implement while keeping good values of uniqueness and reliability. They exploit small variations in the integrated circuit manufacturing process, which translate into different start-up values for cross-coupled structures, and different leakage currents or delays for different realizations of the same circuit [10]. Delay PUFs are the most common due to its simplicity and flexibility. Existing delay PUFs are: arbiter [11], tristate buffer [12], and ring oscillator (RO) PUFs [11]. Among them, ring oscillator PUFs are the best option concerning uniqueness and reliability [10]. Each bit response is generated by the comparison of two ring oscillator frequencies, where the chosen ring oscillator pair is determined by the challenge.

PUFs are a good option for dynamically generating secret keys. However, due to noise, when a PUF is driven by the same challenge, a noisy version of the recorded response is obtained. To cope with this problem, helper data or fuzzy extractor algorithms are the most used schemes [13]-[14]. The basic scheme of fuzzy extractor algorithms consists of the following two phases:

Enrollment: The helper data, W (public), is generated from the PUF response added to one codeword *c* (private), which has been chosen randomly from a code C.

Key reconstruction: This phase consists of two steps named 'information reconciliation' and 'privacy amplification'. In the first step, the PUF is challenged as in the enrollment, now obtaining a probably noisy response, R'. This output is subtracted to the helper data (usually XOR(W,R') is used) and the result goes through an error correcting code (ECC) decoding to recover the codeword *c*. From c, the original PUF response, R, can be reconstructed (R = XOR(W,*c*)). In the 'privacy amplification' step, a hash function is used to generate the key so as to provide well randomness.

Figure 2 shows a scheme of key generation using PUFs and fuzzy extractor.

As mentioned above, noise should be removed from PUFs to make key repeatable. For this purpose, Guajardo *et al.* in [1] proposed the use of binary BCH codes for SRAM PUFs. However, BCH requirements in these applications become complex, heavy and time cost. Bösch *et al.* in [13] introduced the idea of using combined codes (for example BCH and repetition codes) obtaining reductions of even 70% of necessary source bits. Another approach, like the one in [15], faces the problem from the perspective of using soft decision. These approaches reduce hardware requirements but at the cost of increasing the complexity of the system.

Instead of adding postprocessing, Maiti *et al.* in [16] proposed to address the problem of noise when designing the PUF. The idea is to compare the "right"couples of oscillators, that is, choosing those rings whose difference in frequency is reliable (the challenge is the variable that selects the couple). Using this approach, results in [16] show no variation of the PUF response with temperature, and a constant value of 4 unstable bits due to voltage variations (independently of the ring oscillators employed). This means that it is possible for ring oscillator PUFs to generate keys without using error correcting codes. Using this scheme, N ring oscillators are required to generate N-1 independent bits.

The PUF selected in this work is the structure proposed in [16]. The number of ring oscillators used is 33 to generate a 32 bit response. To ensure correcting any possibly small error in the PUF response, a repetition code R(3,1) has been selected.

IV. TRNG IMPLEMENTATION

Since nonces must be as random and not repeatable as possible, TRNGs should be employed. Our election is the use of ring oscillators-based TRNGs due to its simplicity (digital approach and simple harvesting mechanism) and the advantage of using the same basic elements employed by the key generating structure, which allows obtaining a compact

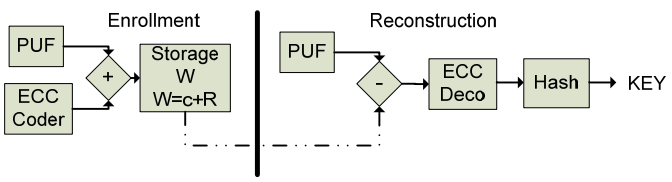

Figure 2. Cryptographic key generation with PUFs

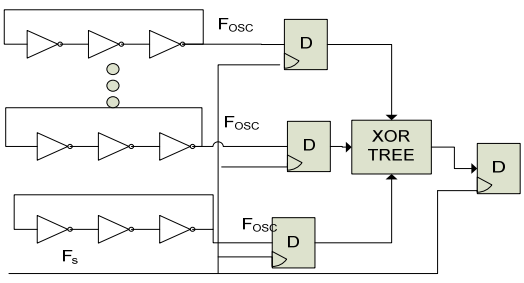

Figure 3. TRNG using ring oscillators

and efficient design because ring oscillators can be shared by both structures.

TRNGs that use ring oscillators are based on sampling phase jitter in the output of oscillator rings. They were first introduced by Sunar *et al.* in [17]. That proposal contained 114 ring oscillator (each of them with 13 inverters) and processed the output with a resilient function (implemented as a BCH code) to increase the entropy and remove bias from the random signal. Schellekens *et al.* in [18] employ the same construction as in [17], but using 210 ring oscillators with 3 inverters. Alioto *et al.* in [19] and Wold *et al.* in [20] suggested an enhancement by adding an extra D flip-flop after each ring instead of using the BCH post-processing stage (Figure 3). Alioto *et al.* confirmed with experimental results of implementations in different FPGAs and within the same FPGA that degree of randomness is essentially unaffected by process variations. Wold *et al.* [20] reduced the number of inverters per ring to 3 (Figure 3) because their experimental results showed dispersion decreases with ring length. They studied that structures with 25 and 50 rings passed NIST and DIEHARD tests. These results were confirmed (and extended to consider restart test) by Maiti *et al.* for structures from 32 to 128 rings [21]. Alioto described the probability of truly randomness, P_{rand}, of the sampled value for a group of n ring oscillators as [19]:

$$\mathbf{P_{rand}=1-[1-min\left(2\frac{\sigma j,cycle\sqrt{\frac{Fosc}{Fs}}}{Tosc},1\right)]^n} \quad (1)$$

Where the ratio $\frac{\sigma j,cycle}{Tosc}$ (between the standard deviation of the cycle-to-cycle jitter and the oscillator period) is independent of the number of inverters and has a typical value around 2% [19]. Hence, randomness increases by increasing the oscillator frequency, F_{osc} (that is, by reducing the number of inverters), by increasing the number of rings oscillators, and by reducing the sampling frequency of D flip-flops, F_s. The latter determines the throughput of the TRNG, so that its selection becomes a trade-off between randomness and bit rate.

The TRNG selected in this work is the scheme proposed in [20] and tested in [21], which uses three inverters per ring. Minimal number of oscillators needed is 25 but as increase of ring oscillators increase randomness and the selected PUF construction require 33, we would reuse these 33 ring oscillators to build the TRNG.

V. HASH SELECTION

Along last decades, several attacks have broken the hash functions available. The early DES function was replaced by SHA-1 in 1995, while SHA-1 was replaced by the SHA-2 hash family in 2002. However, SHA-2 has also been broken no long time ago. This is why the NIST (National Institute of Standards) opened a public competition to develop a new cryptographic hash algorithm still competitive in area and performance. The resulting algorithm will be called "SHA-3". The competition is in its second round stage. Comparison between all second round SHA-3 candidates in terms of area and throughput [22] points at Keccak as the possessor of a good trade-off between throughput and area while keeping good security features.

Keccak is a cryptographic hash family based on sponge functions [23]. Any instance of the Keccak function makes use of one of the seven Keccak-f permutations, denoted Keccak-f[b], where b is the width of the permutation. These permutations are iterated constructions consisting of a sequence of rounds, where each round implements five invertible steps. Keccak function is defined by a set of parameters that are related among them:

- b: width permutation (equal to 25, 50, 100, 200, 400, 800, or 1600)
- n: length of the output message
- c: capacity, limited by c < b. Choosing a c value higher than 2n avoids generic attacks with complexity below 2^n.
- r: rate, should be a power of 2 and fulfill: r=b-c
- n_r: number of rounds in Keccak-f. $n_r = 12 + 2*l$, with $w=2^l = b/25$.

Our selection for the hash function has been Keccak[b=400] as it is the minimum permutation size that allows an output message (n) of 128 bits with enough security. In this case, the selected values for the basic parameters are: n=128, b=400, c=272, r=128 and n_r =20. This construction is employed in the fuzzy extractor scheme for generating the secret key, and in the protocol to generate the response of each side under authentication.

VI. RESULTS

The described scheme has been analyzed considering an FPGA Xilinx Spartan XC3S500E as the target device.

Based on the work in [21], PUF and TRNG circuits have been merged using 33 (fixed by PUF requirements) ring oscillator blocks. They are shared avoiding unnecessary hardware duplication and power dissipation. The number of inverters per ring depends on the operation mode, being three in the case of nonce generation, and five [16] in the case of PUF operation. Ring oscillator blocks and the complete PUF-TRNG module are shown in Figure 4.

Nonce size chosen is 48 bits. The TRNG structure takes around 2 cycles of the sampling frequency (f_s) plus another clock system cycle to generate one bit. This results in a total of 96 sampling frequency cycles plus 48 clock system cycles to generate a nonce. Since maximum F_s is 50MHz in the target device, required time for nonce generation becomes 2.88μs.

Key size selected is 128 bits. It results from hashing with Keccak[400] a 32 bit stream provided by the ring oscillator PUF. The PUF-TRNG module takes around 3 clock system cycles per bit, what makes a total of around 96 clock cycles to provide 32 bits. Using a repetition code R(3,1) to improve reliability means that 3 bits are generated for each required key bit. Hence, time is triplicated, resulting in 288 system clock cycles that, at 50MHz, becomes 5.76 μs.

The high-speed core approach of Keccak has been implemented [24]. Assuming that the input buffer of Keccak [400] is full, 20 clock cycles are required to obtain an output of 128 bits. Presuming 5 clock cycles to fill the buffer, then latency is less than 25 clock cycles, which results in a total of 500ns for a 50MHz system clock.

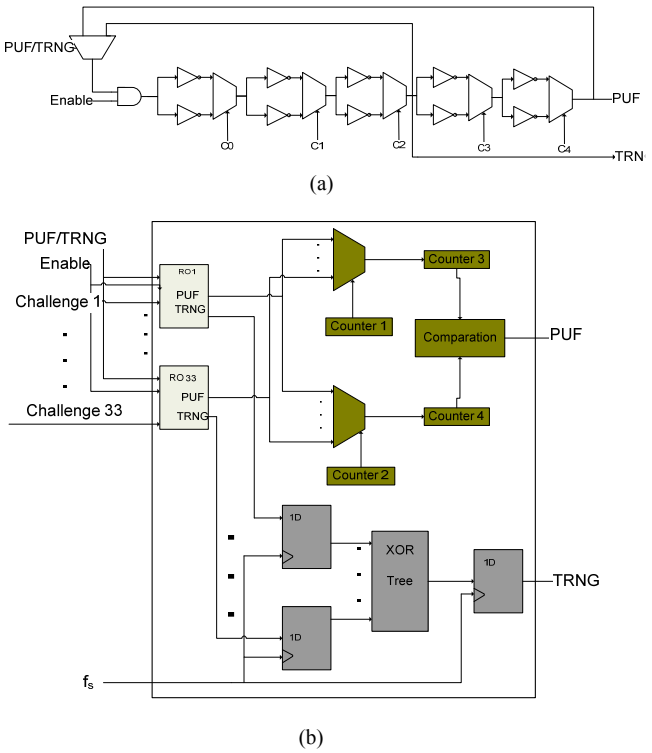

Figure 4. (a) Ring oscillator block. (b) PUF-TRNG module

Complete procedure of key generation requires hashing of PUF response, so that it takes around 6.26 μs (5.76 μs plus 500ns of hashing).

Challenges of the PUF consist of five bits for each ring oscillator, which requires a RAM of 33 words of 5 bits.

Responses are formed by hashing the concatenation of two nonces and the secret key obtained from the PUF response. Using Keccak [400] to generate a final response of 128 bits takes around 500ns.

Under these conditions, the resources employed by the different modules in a Xilinx Spartan XC3S500E are shown in Table I.

TABLE I. Occupation of authentication system in a Spartan XC3S500E

	Slice	**Flip Flops**	**LUTs**	**Occupation**
Keccak [400]	699	498	1,315	5%
PUF-TRNG	51	57	97	<1%
RAM	14	5	28	<1%

VII. Conclusions

Proposed authentication system offers good cryptographic features since their components avoid side channel attacks and tampering in key generation, in one side, and reply, dictionary and man in the middle attacks in the authentication procedure, in the other side. At the same time, the resources employed by the system are less than 10% of a Spartan XC3S500E, which makes this approach a good candidate for constrained resources platforms. The time spent by each extreme is: 2.88 μs for nonce generation, 6.26 μs to generate the key, 0.5 μs to construct the response, 6.26 μs to generate again the key, and 0.5 μs to calculate the response of the other extreme. This makes a total of 16.40 μs at each authentication extreme.

References

[1] J. Guajardo, S. S. Kumar , G.-J. Schrijen , P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection", in Proc. CHES 2007.

[2] P. Tuyls, J. Guajardo, L. Batina, and T. Kerins, "Anti-Counterfeiting," In Security with noisy data, T. Kevenaar, P. Tuyls, and B. Škorić (eds.), Springer, pp. 293-312, 2007.

[3] G. Hammouri, B. Sunar, "PUF-HB: A Tamper-Resilient HB Based AuthenticationProtocol" , in ACNS 2008.

[4] L. S. Kulseng, "Lightweight mutual authentication, owner transfer, and secure protocols for RFID Systems", PhD thesis, Iowa State Univ. 2009.

[5] W. Diffie, M. E. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory* , vol. 22 (1976), pp. 644-654.

[6] R. Pappu, "Physical One-Way Functions", PhD thesis, MIT, 2001.

[7] P. Tuyls, G.-J. Schrijen, B. Skoric, J. van Geloven, N. Verhaegh, R. Wolters: "Read-proof hardware from protective coatings", in Proc. CHES 2006, pp. 369-383.

[8] K. Kursawe, A. Sadeghi, D. Schellekens, P. Tuyls, B. Škorić, "Reconfigurable physical unclonable functions enabling technology for tamper-resistant storage", in Proc. HOST 2009, pp. 22-29.

[9] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, "Silicon physical unknown functions", ACM Conf. on Computer and Communications Security -CCS 2002, pp. 148-160.

[10] S. Eiroa, I. Baturone, A. J. Acosta, J. Dávila, "Using physical unclonable functions for hardware authentication: A survey", in Proc. DCIS 2010, in press.

[11] G.E. Suh, S. Devadas,"Physical unclonable functions for device authentication and secret key generation", in Proc. Design Automation Conference, 2007.

[12] E. Ozturk, G. Hammouri, B. Sunar, "Physical unclonable function with tristate buffers", in Proc. ISCAS 2008, Seattle, USA.

[13] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, P. Tuyls, "Efficient helper data key extractor on FPGAs", in Proc. CHES 2008, pp. 181-197.

[14] Y. Dodis, M.Reyxin, A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data", in Proc. of Eurcrypt 2004.

[15] M.-D. Yu, S. Devadas: "Secure and robust error correction for physical unclonable functions." IEEE Design & Test of Computers, Jan. 2010.

[16] A. Maiti, P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators," in Proc. FPL 2009.

[17] B. Sunar, W. J. Martin, D. R. Stinson, "A provably secure truerandom number generator with built-in tolerance to active attacks," *IEEE Trans. Comput.*, vol. 56, no. 1, pp. 109–119, 2007.

[18] D. Schellekens, B. Preneel, I. Verbauwhede, "FPGA vendor agnostic true random number generator", in Proc. FPL 2006.

[19] M. Alioto, L. Fondelli, S. Rocchi, "Analysis and performance evaluation of area-efficient true random bit generators on FPGAs", in Proc. of ISCAS 2008, pp. 1572-1575, Seattle (USA), May 2008.

[20] K. Wold, C. H. Tan, "Analysis and enhancement of random number generator in FPGA based on oscillator rings", in Proc. Int. Conference on Reconfigurable Computing and FPGAs, 2008.

[21] A. Maiti, R. Nagesh, A. Reddy, P. Schaumont, "Physical unclonable function and true random number generator: a compact and scalable implementation," in Proc. GLSVLSI 2009.

[22] N. J. Hopper, M. Blum,, "Secure human identification protocols" , in Advances in Cryptology - ASIACRYPT 2001.

[23] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, "Sponge functions", in Proc. Ecrypt Hash Workshop 2007, May 2007.

[24] http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations.