HELPER-LESS PHYSICALLY UNCLONABLE FUNCTIONS AND CHIP AUTHENTICATION

Riccardo Bernardini and Roberto Rinaldo

DIEGM–University of Udine
Via delle Scienze 208, 33100 Udine, Italy
{riccardo.bernardini,rinaldo}@uniud.it

ABSTRACT

Physically Unclonable Functions (PUFs) have been recently proposed as a way to include, in chips, functions that can act as "fingerprints" of the chip, to be used in applications like chip authentication (*strong PUF*) or private ID generation (*weak PUF*). Most of the schemes proposed for weak PUF exploit *helper data* to make the PUF more reliable. The use of an helper, however, has some drawbacks such as complexity and the introduction of a possible attack point. In this manuscript we propose the use of a new type of weak PUF called *helper-less PUF*s (H-PUF) that does not require the use of helper data. We analyze theoretically the problem of designing an H-PUF and use the result to show how an H-PUF can be built. We also show that the proposed PUF, albeit being of weak type, can also be used for chip authentication.

Index Terms— security, physically unclonable functions, chip authentication, authentication protocol

1. INTRODUCTION

The increasing requirements for security motivated a good amount of research in the last years. A problem that it is currently analyzed is how to store a secret in a chip so that even an attacker that is able to physically open the chip and study it, cannot get the secret. This problem gave rise to introduction of Physically Unclonable Functions (PUFs) [1–7].

A Physically Unclonable Function (PUF) is a function that (typically) maps binary words to binary words and whose behavior depends on the uncontrollable fine details of the integrated circuit (e.g., the exact channel length of a MOSFET). This sensitivity should make the PUF practically impossible to reproduce, even for the original chip maker. In a sense, a PUF is like a fingerprint for the chip. As each person has a unique fingerprint, every chip has its own PUF; as the fingerprint minutiae are the result of casual variation during the fetal development, the PUF is the result of casual variation during chip production. As fingerprints, the ideal PUF is at the same time random and deterministic: random because it should be impossible to predict the PUF of a given chip, deterministic because the PUF of a specific chip should always give the same result when queried with a given input.

In the literature, PUFs are typically partitioned into two classes, *strong* and *weak* PUFs, depending on their domain size: *strong* PUFs have a very large domain size (exponential with the required silicon area); *weak* PUFs have a limited domain size that can reduce to the empty set, making the PUF a constant function. The typical usage of a constant PUF is to provide the chip with a unique secret ID that can be used, for example, to generate private cryptographic keys. The typical use of a strong PUF is for Challenge Response Pair (CRP) based chip authentication [8, 9].

A major problem that both strong and weak PUFs must solve is the fact that most of the proposed PUFs are not strictly deterministic and their output to a given query can change. When strong PUFs are used for authentication the problem is solved by accepting the response even if the match is not perfect. This lenience, however, also helps the attackers, who will need to reproduce only a good approximation of the PUF.

Weak PUFs used for private key creation are even more delicate. A single wrong bit in the key can make the whole system useless. In order to make the PUF output more stable, two-step schemes employing *helpers* based on error correcting codes are usually proposed. Fig. 1 shows the basic idea. In the first step (*enrollment*, carried out only once, at the first turn-on, Fig. 1a) the PUF output X is used as noise to corrupt a randomly chosen codeword C in order to obtain $Y = X \oplus C$, which is saved in a Non Volatile Memory (NVM). In the second step (carried out every time the chip is turned on, Fig. 1b), the PUF is queried again to obtain $\hat{X} = X \oplus E$, where E is usually a binary word with low Hamming weight, $\hat{X}$ is XOR-ed with the stored word Y to obtain $\hat{X} \oplus Y = C \oplus E$, from which C and E can be computed by exploiting the properties of the chosen error correcting code. From E and $\hat{X}$ one obtains the original X.

The main drawbacks of helper-based weak PUFs are the necessity of including complex error correction procedures on chip, the fact that the saved Y could leak information about X and the possibility that the error correction step could leak information in a side-channel attack [10].

In this work we propose the use of *Helper-less PUFs (H-PUFs)*, that is, weak PUFs that do not require an helper. We first analyze theoretically the problem of constructing a reliable Helper-less PUF (H-PUF) and successively, from the

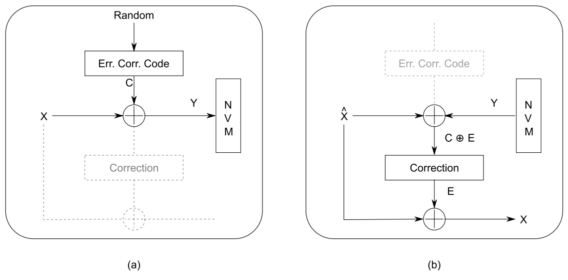

Fig. 1. Example of helper-based PUF. (a) Enrollment phase, (b) Recovering the original X.

theoretical results, we derive a procedure to build an H-PUF. Moreover, we show that the proposed H-PUF, despite being a weak PUF, can be used also for secure device authentication.

Prior Work As said above, a first group of PUFs is represented by strong PUFs. Several solutions have been proposed for strong PUFs. The most common ones are arbiter-based PUFs [2, 3], which exploit a race condition in signal propagation, and ring-oscillator PUFs [4, 5, 9] that exploit the difference in the frequency of ring-oscillators. Attacks to strong PUFs aim typically to obtain a physical model of the device to be simulated in software [3, 9, 11].

As for weak PUFs, the most commonly proposed structures are based on Static RAM (SRAM) [6, 7], exploiting the fact that, because of construction asymmetries, many SRAM cells have a preferred state they fall into when left uninitialized. Since the state is nevertheless random, many procedures have been proposed to stabilize the PUF output. The most common procedure uses a *fuzzy extractor*, usually based on error correction codes [12, 13]. In [8] a stabilizer based on pattern matching is proposed, while in [14] *Index Based Syndrome coding* is proposed. Attacks to weak PUFs usually aim to the helper value, which is stored in a NVM which could be accessible to the attacker [10].

Differences with Existing Literature Our approach is totally different with respect to what is proposed in the literature. More precisely, (i) we show that weak PUFs can be implemented without using an helper, avoiding in this way the weak point usually attacked and (ii) we show that chip authentication can be safely done with H-PUFs, removing in this way the weak point due to the necessity of being lenient with respect to result mismatch. Toward such an objective, we develop a theory of H-PUFs and use the results to show that it is possible to build an arbitrarily reliable H-PUF.

2. BUILDING AN H-PUF

Our objective is to show how to build a PUF without using any helper data. Toward such an end, we first analyze the problem from a theoretical point of view.

2.1. The model

We suppose that the device has a *Raw Generator (RG)* whose behavior is very sensitive to variations in physical characteristics (e.g., oxide thickness). Every time the RG is turned on it produces a *raw output* that is to be used to determine the secret ID. Let $\mathbb{V}$ be the set of all possible raw outputs.

Remark 2.1
Usually the raw generator will be built as an array of *elementary generators* or *cells* (e.g., an SRAM-based PUF is made by many SRAM cells), but this is not necessary for our development. If the cell outputs a single bit, we will call it a *binary cell*.

Ideally, the behavior of a given RG should be deterministic, in the sense that every time it is turned on, it should always produce the same raw output. However, since this is not true due to noise during the power-up process, the behavior of an RG is more faithfully described as a random variable V assuming values in $\mathbb{V}$. The distribution of V, however, depends in turn on the random physical variations occurred during device construction. We will model the variability in the statistical behavior by introducing a random variable (the *statistical parameter*) Q assuming values in a suitable set $\mathscr{Q}$ and distributed according to density f_q. The link between q and the behavior of the RG is described by giving the conditional density (or distribution or mass probability function) of V given Q.

Example 2.1
It is worth specializing the ideas just exposed in the case of an SRAM cell. Since the final state of an SRAM cell is a bit, the set of possible output values is $\mathbb{V} = \{0, 1\}$. Since the behavior of a single SRAM cell is uniquely determined by the probability q that the cell powers up in state 1, we can use q as the statistical parameter and the interval $[0, 1]$ as $\mathscr{Q}$. The parameter q is itself random, since it depends on cell asymmetries resulting from random manifacturing process variations [6,7,15]. Finally, in the SRAM case, the conditional probabilities of the output given the statistical parameter are $P[V = 1|Q = q] = q$ and $P[V = 0|Q = q] = 1 - q$.

2.2. The objective

Since the output of the RG is not deterministic, we are interested in post-processing it to make it more "stable." We will achieve this by using a *stabilizer*, defined as a function $\mathscr{S} : \mathbb{V} \to \mathscr{I}$, where $\mathscr{I}$ is the set of the possible IDs. Let $I := \mathscr{S}(V)$ denote the r.v. associated with the generated ID. The probability $P[I = i]$, $i \in \mathscr{I}$, clearly depends on the distribution of $V \in \mathbb{V}$ which in turn depends on the value assumed by Q. If $q \in \mathscr{Q}$, we will say that $i \in \mathscr{I}$ is *a winner*[1] in q if

$$\forall j \in \mathscr{I} \quad P[I = i|Q = q] \geq P[I = j|Q = q]. \tag{1}$$

[1]Note that the winner is not necessarily unique.

Ideally, the output of the stabilizer should not change at every turn-on. This suggests to define the *stability* in $q \in \mathscr{Q}$ as

$$\varepsilon(q) = \max_{i \in \mathscr{I}} P[I = i | Q = q] \tag{2}$$

and the *global stability* η as

$$\eta := \inf_{q \in \mathscr{Q}} \varepsilon(q). \tag{3}$$

Definition (3) of global stability is very strong since it requires that the stability is never smaller than η. It will be shown in Section 2.3 that this requirement is too strong and that a "softer" version of the idea of stability ((η, δ)-*stability*) is needed. We will say that a stabilizer is (η, δ)-*stable* if

$$P[\{q \in \mathscr{Q} : \varepsilon(q) \le \eta\}] \le \delta. \tag{4}$$

Remark 2.2

The idea is that a stabilizer is acceptable if the cases with low stability are "corner cases" that happen infrequently. We will show in Section 2.4 that arbitrarily good stabilizers (that is, with η arbitrarily close to 1 and δ arbitrarily close to 0) are possible.

2.3. Theoretical bound to global stability

Ideally, we would like to design a stabilizer with global stability η as close to 1 as possible. Unfortunately, according to the following theorem, for a large class of systems, this is not possible. We have the following result.

Theorem 1. *If $\mathscr{Q}$ is connected, then $\eta \le 1/2$.*

Remark 2.3

The proof of Theorem 1 is rather technical and will not be given here for reasons of space. An intuitive motivation can be given with the help of Fig. 2, which shows the case of a stabilizer $\mathscr{S} : \{0,1\}^2 \to \{\alpha, \beta, \gamma\}$ mapping pairs of bits from two SRAM cells into a 3-symbol output alphabet. Since the parameter set for an SRAM cell is $\mathscr{Q} = [0,1]$, the parameter set for the whole RG is $\mathscr{Q} \times \mathscr{Q} = [0,1]^2$, shown as a square in Fig. 2. Set $\mathscr{Q} \times \mathscr{Q}$ in Fig. 2 is partitioned into three sets labeled with the corresponding winning symbol. Since point q lies on the border between sets β and γ, we deduce

$$P[I = \beta | Q = q] = P[I = \gamma | Q = q] > P[I = \alpha | Q = q] \tag{5}$$

which implies

$$\eta \le \varepsilon(q) = P[I = \beta | Q = q] \le 1/2 \tag{6}$$

proving Theorem 1 in this very special case. It is possible to show that the same circumstance happens as soon as the parameter set is connected.

Note that Theorem 1 is very general, since it holds independently of the nature of the raw output that can be discrete (e.g., bits), continuous (for soft-decision stabilizers) or multi-dimensional (e.g., if the RG is queried many times to take a decision). In particular, Theorem 1 shows that it is not possible, without further processing, to build globally stable stabilizers, with $\eta \simeq 1$, for SRAM cells, the case we focus on in the following.

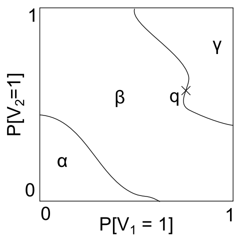

Fig. 2. Example of why $\eta \le 1/2$. In q both β and γ have the same probability that cannot be larger than 1/2.

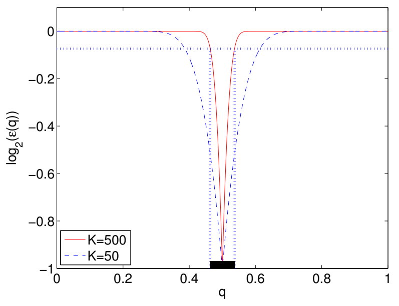

Fig. 3. Stability vs. $q = P[V = 1]$.

2.4. Building (η, δ)-stable stabilizers

Fact 1. *For every $\eta < 1$ and $\delta > 0$ an (η, δ)-stable stabilizer for binary cells (i.e., SRAM) and output alphabet $\{0,1\}$ exists.*

We will prove Fact 1 constructively. Suppose one has a binary cell, for example an SRAM cell, with $P[V = 1] = q$. The cell is turned on and off K times and the relative frequency ω of outcome "1" is measured. The output of the stabilizer is 1 if $\omega > 1/2$ and 0 otherwise. Since ω is approximately $\mathscr{N}(q, q(1-q)/K)$, with some simple derivations one gets

$$\varepsilon(q) = \Phi\left(\sqrt{K}\frac{|q - 1/2|}{\sqrt{q(1-q)}}\right) \tag{7}$$

where Φ is the cumulative distribution function of $\mathscr{N}(0,1)$. It is easy to check[2] that for every K, $\varepsilon(0) = \varepsilon(1) = 1$ and $\varepsilon(1/2) = 1/2$. Moreover, as K increases, $\varepsilon(q)$ becomes flatter around 0 and 1, with a steeper "notch" around 1/2. Fig. 3 shows $\varepsilon(q)$ for $K = 50$ and $K = 500$. The horizontal dotted line corresponds to a stability equal to 0.95, the probability that $\varepsilon(q)$ is lower than 0.95 is the probability that q belongs to the interval marked in Fig. 3. It is easy to show that for every $\eta < 1$ and $\delta > 0$ one can satisfy (4) by choosing K large enough.

2.5. From (η, δ)-stability to global stability

A (η, δ)-stable stabilizer can be transformed into a stabilizer with global stability η if one can recognize and discard, at

[2]Independently of the Gaussian approximation.

manufacturing time, those devices whose stability is lower than η. According to the definition of (η, δ)-stability, we expect to discard a fraction δ of devices. Note that in this way we actually "break" the set $\mathscr{Q}$ into unconnected pieces, escaping the grip of Theorem 1.

Remark 2.4

Estimating cell reliability at run-time. An alternative approach that does not require to store the reliability mask is to estimate the reliability at run-time. First observe that it is reasonable to assume that in every implementation the chip will need some way to check if the generated ID is correct, since every scheme that can be devised will always have a residual error probability. The check could be done, for example, by storing in a NVM a cryptographic hash of the ID.

Let C be the number of binary cells. This approach requires to measure ω_i, $i = 1, \ldots, C$, for every cell i and to set the corresponding output to 0 if $\omega_i < 1/2 - d$, to 1 if $\omega_i > 1/2 + d$ and to "unknown" otherwise. If the number of cells in the "unknown" state is small[3], one can try all the possible combinations of values for the unknown bits, until an ID that matches with the cryptographic hash is found.

Example 2.2

In the case of an SRAM cell, one can estimate q for a newly produced device by turning the device on and off K times, by measuring the relative frequency ω of "1" and discarding the device if $|\omega - 1/2| \leq d$, where d is chosen so that the probability of accepting an unreliable device is small. According to the model presented in [7], parameter q in (7) has distribution $F_q(a) = \Phi(\lambda_1 \Phi^{-1}(a))$, $\lambda_1 \simeq 0.065$. We set $d = 4/\sqrt{K}$, $K = 10000$. By conditioning on q, it is easy to evaluate $\mathrm{P}[0.5 - d < \omega < 0.5 + d] < 0.006$, which represents the fraction of discarded cells (or of the cells whose output is evaluated as unknown in the run-time procedure). Given q, we assume that outcomes ω_1 and ω_2 of the relative frequency of "1" in two different experiments (e.g., two successive power-up events) are i.i.d. with distribution (7). It is easy to evaluate $\mathrm{P}[\omega_2 > 0.5 | \omega_1 > 0.5 + d] > 1 - 10^{-11}$. These figures show that the proposed procedure is indeed feasible for the implementation of a reliable weak PUF.

3. CHIP AUTHENTICATION VIA WEAK PUF

As anticipated in the Introduction, weak PUFs are considered viable for secret ID generation, while strong PUFs are used for chip authentication via a CRP-based protocol. In this section we want to show that it is possible to use weak PUFs, and the generated secret IDs, for chip authentication, and that the resulting protocol is even stronger than the usual CRP-based approach. The procedure described in this section can be used, if desired, also to setup a common cryptographic key.

CRP authentication works as follows. Just after manufacturing, the PUF is queried in a safe environment with many inputs (challenges) and the corresponding outputs (responses) are acquired. Every challenge-response pair is saved in a database (DB). Successively, in order to authenticate the chip, a challenge is taken from the DB and given to the chip. The response is checked against the response stored in the DB.

The CRP-based authentication protocol has two weak points. Since the PUF is not deterministic, the response is accepted even if the match is not perfect. This lenience, however, helps the attackers, who will need to reproduce only an approximation of the PUF. Moreover, and more importantly, it is vital that the DB of CRP is stored in a safe place since it represents an important target for an attacker.

The procedure described in this section solves these problems by employing a weak PUF and avoiding the use of a secret DB. We will make use of an Encrypted Key Exchange (EKE) protocol (e.g., [16]), an encryption function E and a cryptographic hash H [17]. We will also need a multiplicative Abelian group $\mathbb{D}$ suitable for use in a Diffie-Hellman (DH) protocol[4] [18] and a generator g of $\mathbb{D}$ [19].

Suppose two chips (named Alice and Bob) want to authenticate each other and agree on an encryption key. Let ID_A and ID_B be Alice's and Bob's secret IDs. Alice and Bob compute and publish their "public keys" $K_A = g^{\mathrm{ID}_A}$ and $K_B = g^{\mathrm{ID}_B}$.

Remark 3.1

Note that, differently from the CRP DB, there is no need to keep the public key secret [17]. Actually, the chip can safely store it in a NVM and use it to check the generated ID (see also Section 2.4).

As well known, an EKE allows two actors that share a secret (e.g., a password) to authenticate each other and, at the same time, agree on a shared key. The simplest way to apply an EKE in our case is to use $S = (K_A)^{\mathrm{ID}_B} = (K_B)^{\mathrm{ID}_A} = g^{\mathrm{ID}_A \mathrm{ID}_B}$ as shared secret. A safer variant that does not use the long-term secret S in the EKE is the following. Alice generates a random string X_A and sends it to Bob, Bob generates X_B and sends it to Alice. Bob and Alice encrypt X_A and X_B with the long-term secret S to obtain $Y_A = E(X_A; S)$ and $Y_B = E(X_B; S)$. Finally, the short-term secret to be used in the EKE is computed as $G = H(Y_A \cdot Y_B)$, where $\cdot$ denotes string concatenation. Note that G is a throwaway secret, so, even if recovered, it is useless to a potential attacker.

4. CONCLUSIONS

We analyzed the problem of designing a reliable helper-less weak PUF. We showed that, although in many cases it is not possible to have an H-PUF with global stability larger than $1/2$, it is possible, for every choice of $\eta < 1$ and $\delta > 0$, to build a (η, δ)-stable H-PUF based on binary cells. We also proposed procedures for the generation of secret IDs and chip authentication using SRAM. Quantitative figures confirm that the presented procedures are indeed feasible for practical implementation.

[3]This happen if the PDF of q is well "concentrated" around 0 and 1, so that the probability of getting $|\omega_i - 1/2| \leq d$ is small.

[4]For example, an elliptic curve or $\mathbb{Z}/p\mathbb{Z}$ for a large prime p.

5. REFERENCES

[1] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*, New York, NY, USA, 2002, CCS '02, pp. 148–160, ACM.

[2] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M. van Dijk, and S Devadas, "Extracting secret keys from integrated circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 13, no. 10, pp. 1200–1205, 2005.

[3] Daihyun Lim, *Extracting Secret Keys from Integrated Circuits*, Ph.D. thesis, MIT, May 2004.

[4] G.E. Suh and S Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, 2007, pp. 9–14.

[5] G.E. Suh and S Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, 2007, pp. 9–14.

[6] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu, "Initial SRAM state as a fingerprint and source of true random numbers for rfid tags," in *In Proceedings of the Conference on RFID Security*, 2007.

[7] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for SRAM PUFs," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, 2009, pp. 2101–2105.

[8] Z. Paral and S Devadas, "Reliable and efficient PUF-based key generation using pattern matching," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*, 2011, pp. 128–133.

[9] Jeroen Delvaux and Ingrid Verbauwhede, "Fault injection modeling attacks on 65nm arbiter and ro sum PUFs via environmental changes," Cryptology ePrint Archive, Report 2013/619, 2013, http://eprint.iacr.org/.

[10] Jeroen Delvaux and Ingrid Verbauwhede, "Attacking PUF-based pattern matching key generators via helper data manipulation," Cryptology ePrint Archive, Report 2013/566, 2013, http://eprint.iacr.org/.

[11] Dominik Merli, Dieter Schuster, Frederic Stumpf, and Georg Sigl, "Semi-invasive em attack on fpga ro PUFs and countermeasures," in *Proceedings of the Workshop on Embedded Systems Security*, New York, NY, USA, 2011, WESS '11, pp. 2:1–2:9, ACM.

[12] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology - EUROCRYPT 2004*, Christian Cachin and JanL. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, pp. 523–540. Springer Berlin Heidelberg, 2004.

[13] Boris Skoric and Niels de Vreede, "The spammed code offset method," Cryptology ePrint Archive, Report 2013/527, 2013, http://eprint.iacr.org/.

[14] Meng-Day (Mandel) Yu and Srinivas Devadas, "Secure and robust error correction for physical unclonable functions.," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 48–65, 2010.

[15] Bin Zhang, A. Arapostathis, S. Nassif, and M. Orshansky, "Analytical modeling of sram dynamic stability," in *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, 2006, pp. 315–322.

[16] Y. Sheffer, G. Zorn, H. Tschofenig, and S. Fluhrer, "An EAP authentication method based on the encrypted key exchange (EKE) protocol," RFC 6124, Feb. 2011, http://tools.ietf.org/html/rfc6124.

[17] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons, 1994.

[18] Bailey Diffie, Martin Hellman, and Ralph Merkle, "Cryptographic apparatus and method," PatentUS US 4200770 A, Apr. 1980.

[19] N. Jacobson, *Basic Algebra I*, W.H. Freeman, New York, NY, 1985.