PUF-HB: A Tamper-Resilient HB Based Authentication Protocol

Ghaith Hammouri and Berk Sunar

Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609-2280
{hammouri,sunar}@wpi.com

Abstract. We propose a light-weight protocol for authentication of low-power devices. Our construction PUF-HB merges the positive qualities of two families of authentication functions. PUF represents physically unclonable functions and fulfills the purpose of providing low-cost tamper-resilient challenge-response authentication. On the other hand, the Hopper Blum (HB) function provides provable cryptographic strength against passive adversaries. By building on an earlier proof of the security of HB^+ by Katz et al. [1], we rigorously prove the security of the proposed scheme against active adversaries. While the active adversary model does not include man-in-the-middle attacks, we show that a previously successful man-in-the-middle attack proposed for HB^+, does not carry to PUF-HB.

Keywords: HB+, PUF, tamper resilience, provable security.

1 Introduction

Low cost and power efficient devices are essential for the next generation ubiquitous networks. By utilizing devices ranging from wireless sensor nodes to RF identification devices (RFIDs) and smartcards, these networks are envisioned to support a large number of applications carrying out diverse tasks. It is clear that low-cost and lightweight security schemes are absolutely essential for the adoption of ubiquitous networks. To further highlight the security requirements of these networks, we observe that despite their diversity, the deployed devices have three things in common: they store sensitive information, they transmit data through unprotected wireless networks, and typically an adversary will have physical access to the device due to the nature of the applications. It is essential that sensitive information is safely stored and communicated. However, since most of the pervasive devices are passive devices with limited power sources and with stringent constraints imposed on their computational resources, implementing protection measures on these low-cost devices becomes a challenge. Furthermore, attacks that circumvent the data channel and instead exploit the so called side-channels [4,3] are posing a major threat to constrained devices. These attacks are classified into two groups. Passive attacks solely observe side-channels (e.g. computation time, power consumption, electromagnetic emanation, temperature attacks etc.) to deduce internal secrets from leaked side-channel profiles.

S.M. Bellovin et al. (Eds.): ACNS 2008, LNCS 5037, pp. 346–365, 2008.
© Springer-Verlag Berlin Heidelberg 2008

In contrast, in active attacks the attacker may also inject faults during the computation [5]. Not surprisingly, active attacks are more powerful and are more difficult to prevent. Furthermore, active attacks have also been demonstrated to be useful in circumventing countermeasure against side-channel leakage. Thus, building tamper-proof hardware is crucial for securing devices against passive and active side-channel attacks.

The recently proposed Physically Unclonable Functions (PUFs) provide a promising alternative approach for achieving tamper-resilience [6]. A PUF is a physical pseudo-random function which may be implemented by exploiting the small variances of the wire and gate delays that are unique for every single integrated circuit (IC), even if they are logically identical. These variances depend on highly unpredictable factors, which are mainly caused by the inter-chip manufacturing variations. Hence, given the same input, the PUF circuit will return a different output on different ICs. Additionally, if the PUFs are manufactured with high precision, any major external physical influence will change the PUF function and render the device non-functional. These features are indeed very attractive for any low-cost authentication scheme hoping to achieve tamper-resilience. However, all PUF based authentication schemes proposed so far rely on collecting challenge-response pairs which are unique for every PUF. These challenge-response pairs have to be stored in a database [6], so that the data can be retrieved and used to authenticate any given PUF. It is not hard to see that this solution becomes less practical when the number of devices deployed increases. Another major problem faced by the delay-based PUF design is that unless augmented by traditional cryptographic schemes (e.g. hash functions) or non-linearization techniques [7,8,12] it will not be secure against simple modeling attacks.

In complement to PUFs the HB-based authentication schemes provide a security reduction. In [13] Hopper and Blum (HB) proposed the first HB authentication scheme. The HB protocol is indeed promising for simplifying the authentication process and significantly reducing the power consumption of pervasive networks. An additional major advantage of the HB protocol is that its security is based on the hardness of the learning parity with noise (LPN) problem. The LPN problem is known to be NP-hard [25]. Unfortunately, as pointed out in [13] the HB scheme is weak against active adversaries. In [14], Juels and Weis proposed a hardened version of the HB protocol labeled HB^+ which resists active attacks in the detection based model.[1] HB^+ was shown to be secure [14,1] in the detection based model. In [18], the authors demonstrated a man-in-the-middle attack for breaking HB^+.

In this paper we merge the PUF authentication scheme along with the HB based authentication protocol to produce a hybrid protocol which enjoys the advantages of both schemes while improving the level of security. We propose an authentication scheme which is tamper resilient, and at the same time provably secure against active attacks in the detection based model. In addition, the

[1] In this model, a flag is raised whenever a tag fails to authenticate for a large number of times.

presented protocol resists the man-in-the-middle attacks proposed so far for the HB^+ scheme. From the PUF perspective, our protocol is the first PUF based authentication scheme which is provably secure. For our security proof we closely follow the proof presented in [1].

The remainder of the paper is organized as follows. In the next section we introduce PUFs and present a mathematical model. In Section 3 we give a review of the previously proposed HB based authentication protocols. In Section 4 we define our notation and describe our protocol. The security reduction is presented in Section 5. In Section 6 we describe the man-in-the-middle attack and show that the proposed protocol resists it. In Section 7 we discuss hardware implementation and tamper resilience properties of the proposed scheme. Finally, we present the conclusions in Section 8.

2 Physically Unclonable Functions

A PUF is a challenge-response circuit which takes advantage of interchip variations to achieve tamper-resilience. The idea behind a PUF is to have the same logical circuit produce a different output depending on the actual implementation parameters of the circuit. The variations from one circuit implementation to another are not controllable and are directly related to the physical aspects of the environment. These aspects include temperature, pressure levels, electromagnetic waves and quantum fluctuations. Therefore, two identical logical circuits sitting right next to each other on the same die might still have quite different input-output behavior due to the nature of a PUF.

The reason one might seek to explore this property is to prevent an attacker from cloning the function. Additionally, because of the high level of sensitivity of these physical functions it becomes virtually impossible for an attacker to accurately reproduce the hardware. Another major advantage of the sensitivity of the PUF is to prevent physical attacks on the system. Trying to tap into the circuit will cause a capacitance change and therefore change the output of the PUF. Removing the outer layer of the chip will have a permanent effect on these circuit parameters and again, will change the output of the PUF circuit. Roughly summarized, we can say that a well-built PUF device will be physically tamper-resilient up to its sensitivity level. We would like to note here that one of the major advantages of PUF circuits is their lightweight nature. Using a recent proposal which utilizes tri-state buffers to construct PUF circuits [11], one can show that for an n bit PUF about $2n$ gates is sufficient.

A delay-based PUF [6] is a $\{0,1\}^k \rightarrow \{0,1\}$ mapping, that takes a k-bit challenge a and produces a single output bit r. As shown in Figure 1 the delay-based PUF circuit consists of k stages of switches. Each switch has two input and two output bits in addition to a control bit. If the control bit of the switch is 0, the two inputs are directly passed to the outputs through a straight path. If on the other hand, the control bit to the switch is 1, the two input bits are switched before being passed as output bits. Therefore, the control bit of each switch will decide the path taken by the input signals. The k switches are serially connected

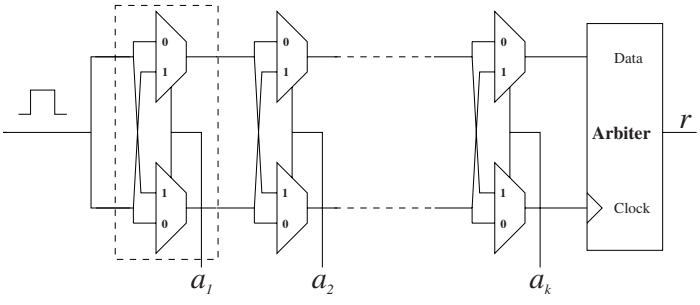

Fig. 1. A basic delay-based PUF circuit

so that the output of each switch is connected to the input of the following switch. The two outputs of the last switch are connected to a flip-flop, which is called the *arbiter*. The two inputs to the first switch are connected to each other, and then connected to a pulse generator.

We briefly describe the operation of PUFs as follows. When a challenge a is received each of its k bits is used as the control bit to one of the PUF switches. Next, a pulse is generated and sent to the first switch. The pulse will break into two pulses entering the upper and lower inputs of the first switch. Depending on the control bit of the first switch the two signals will either travel in a straight path or they will switch locations. Although the paths taken by the signals have been designed to have the same length, due to physical variations the two paths will have a small mismatch. This mismatch will have a different value for each of the two possible paths of the two signals. Therefore, the two pulses will acquire a time delay between them which is dependent on the control bit of the first switch. The same argument applies for the rest of the k switches. Each challenge a will impose a different path on the k switches. Consequently, the total delay mismatch between the two signals will be a function of a. The job of the arbiter at the end of the switch chain is to indicate which signal arrives first. Recall that a flip-flop has two inputs, the data input and the clock input. When the clock input observes a rising edge the data input is captured at the output of the flip-flop. In a PUF setting, if the signal connected to the data input of the arbiter arrives first, the output of the arbiter will be 1. Otherwise, the output will be 0.

In [6] the authors derive a linear delay model for the delay-based PUF. The model is represented by an equation which gives the total delay difference between the two pulses in terms of the challenge bits and the parameters of the PUF circuit. Let D_H represent the total delay of the pulse that enters the data input of the arbiter and D_L be the delay of the pulse that enters the clock input. The mathematical model derived in [6] shows that the difference between the two propagation paths D_H and D_L can be expressed algebraically as

$$\delta = D_L - D_H = \sum_{i=1}^{k} (-1)^{p_i} y_i + y_{k+1}$$

where $p_i = a_i \oplus a_{i+1} \oplus \ldots \oplus a_k$ such that a_i is the i^{th} bit of a. This relation can also be described using $(P = Ua)$ where $P = [p_1, \ldots, p_k]$ and a are represented as column vectors. U is the upper triangular matrix with all non-zero entries equal to 1 and the matrix multiplication is performed modulo 2. Here the y_i parameter denotes the imbalance in the signal propagation paths in i^{th} stage of the PUF circuit.[2] The condition for the output bit of the arbiter circuit, r, can be written as $\delta + T_s > 0 \rightarrow r = 1$ and $\delta + T_s < 0 \rightarrow r = 0$, where T_s denotes the setup time for the arbiter flip-flop. Now, let $Y = [y_1, \ldots, y_{k+1}]$ we can compute the output of the PUF using the following function,

$$r = \mathrm{PUF}_Y(a) = \mathrm{sign}\left(\sum_{i=1}^{k}(-1)^{p_i} y_i + y_{k+1}\right) \quad .^{3} \tag{1}$$

Where $\mathrm{sign}(x) = 1$ if $x > 0$, and 0 if $x < 0$. When the argument in $\mathrm{sign}(x)$ is zero the output becomes a random bit. In fact, the random output bit will in practice be observed for a slightly larger window of mismatch values. In these cases the PUF is said to be *metastable*. We will shortly address this issue by introducing the noise parameter ϵ. It is important to note that the delay variations y_i will depend on the fabrication process of the PUF circuit. Therefore, one would expect these parameters to follow a normal distribution [10]. In particular, the y_i values will follow a Gaussian distribution of mean zero, and a fixed variance. Without loss of generality, we can normalize these values and assume they belong to a normal distribution of mean 0 and variance 1.

The fact that the PUF function can be represented using a linear inequality means that given a sufficient number of challenge-response pairs $(a_{(i)}, r_{(i)})$ for a single PUF,[4] an attacker will be able to model the system using linear programming [27,28,12] or machine learning [29,6] algorithms. This observation seems to completely undermine the idea behind a device labeled unclonable. Although the delay parameters y_i are not measurable, a simple PUF circuit will leak information about these values through its challenge-response pairs. For theoretical and experimental results on PUF modeling, the reader is referred to [8,12,11]. As proposed in [12], in this paper we take advantage of the modelability of the simple PUF circuit. We provide an overall scheme which prevents an attacker from calculating the delay parameters, while at the same time allowing the owner of the circuit to take advantage of this property. We further elaborate on this point in Section 7. For the remainder of the paper we will utilize the PUF model and assume that the y_i parameters fully characterize the operation of the PUF circuit.

It should be noted that even if the best techniques are used to model a PUF circuit, there will always be a level of error in the developed model. This is

[2] The delay model derived here is slightly different from that given in [6], particularly the y_i parameters differ with a factor of two.

[3] For brevity the setup time T_s can be merged with the last delay parameter y_{k+1}.

[4] In this paper we use subscripts with parenthesis to denote different strings, e.g. $a_{(i)}$, and we use subscripts without parenthesis to denote bits of the same string, e.g. a_i.

due to multiple reasons. First, the thermal noise will cause slight fluctuations in the internal delay parameters y_i. Second, although the system is modeled with a linear equation, there will always be some source of non-linearity which will affect the system. Moreover, the two signals propagating inside a PUF circuit will sometimes enter a race condition such that the decision made by the arbiter will be random [9]. This will typically happen when the total delay difference between the two signals is less than the resolution of the arbiter. When the PUF enters such a state the circuit is said to be metastable. As a result, any PUF device can only be modeled up to a certain level of accuracy, i.e. the model will always mispredict some response bits. The best modeling schemes tested so far were shown to reduce the misprediction rate to as low as 3% [8]. Note that the 3% misprediction rate is obtained by implementing error reduction techniques such as majority voting. For the construction in this paper we utilize the built-in noise to secure the HB component. In our notation we denote the amount of error that exists in a PUF circuit model by ϵ.

Before we conclude this section, we introduce the following notation. For any bit v,

$$\mathrm{PUF}_{Y,v}(a) = \left\{ \begin{array}{ll} \mathrm{sign}\left(\sum_{i=1}^{k}(-1)^{p_i} y_i + y_{k+1}\right), & v = 0 \\ \mathrm{sign}\left(\sum_{i=1}^{k}(-1)^{\overline{p_i}} y_i + y_{k+1}\right), & v = 1 \end{array} \right\} \quad , \tag{2}$$

where $\overline{p_i}$ is the complement of p_i.[5] This notation will simplify the derivations in the following sections.

3 The HB Family

The HB authentication schemes base their security on the hardness of the LPN problem. In this section we give a quick review of the LPN problem and the different HB authentication schemes, focusing on HB and HB$^+$. For a certain $\epsilon \in \left(0, \frac{1}{2}\right)$ the LPN problem is denoted by LPN_ϵ, and defined as follows.

Definition 1 ([1]). *Given n random binary k-bit strings $a_{(j)}$ and the bits $z_{(j)} = a_{(j)} \cdot x \oplus \nu$ for some $x \in \{0,1\}^k$, where $a \cdot b$ denotes the binary inner product between a and b, $\nu = 1$ with probability ϵ and 0 with probability $1 - \epsilon$, then find the binary string x.*

The LPN problem is known to be NP-hard [25]. In [13], the authors show that the LPN problem is log-uniform and even hard to approximate within a ratio of 2. Kearns proved in [26] that the LPN problem is hard in the statistical query model. The best known algorithm to solve the LPN problem is the BKW algorithm [20]. However, there has been a number of improvements on the algorithm with the best running time of $2^{O(k/\log k)}$ [21,22,23].

[5] The complement may be realized by simply XOR-ing the most-significant bit of a with v.

In the HB protocol [13], the tag and the reader share a k-bit secret string x. To authenticate the tag, the reader starts sending randomly generated k-bit challenge strings $a_{(j)}$. The tag responds with the bit $z_{(j)} = a_{(j)} \cdot x \oplus \nu$ where the variables are as defined in the LPN problem. The tag and the reader repeat the same step for multiple challenges. Finally, the reader checks to see if the number of errors in the tag's response matches the noise level, and decides to accept or reject accordingly. Note that if the tag's response did not contain noise, then a passive attacker would easily be able to deduce x after collecting k challenge-response pairs using Gaussian elimination. In [13], the authors prove that given an algorithm that predicts $z_{(j)}$ for a random $a_{(j)}$ with some advantage, then this algorithm can be used to solve the LPN_ϵ problem. However, HB is only secure against passive attacks. An active attacker can easily repeat the same challenge multiple times, effectively eliminating the noise and reducing the problem to Gaussian elimination.

To secure the HB protocol against an active attacker the HB^+ protocol was proposed in [14]. In HB^+ the tag and the reader share two k-bit strings x and y. The tag starts the authentication session by sending a random k-bit string $b_{(j)}$. The reader then responds with $a_{(j)}$ just like the HB protocol. Finally the tag responds with $z_{(j)} = a_{(j)} \cdot x \oplus b_{(j)} \cdot y \oplus \nu$, where ν is defined as above. The protocol is proven to be secure against an active attack on the tag (excluding man-in-the-middle attacks). In such an adversary model an attacker is not allowed to obtain final decisions from the reader on whether this authentication session was successful or not. In [14] and [1] the authors show that in this adversary model breaking the HB^+ protocol is equivalent to solving the LPN problem. However, as we pointed out earlier, a simple man-in-the-middle attack was demonstrated on the HB^+ protocol in [18]. Note that in a detection based model this attack will not be successful.

In addition to HB and HB^+, there has been a number of other variations such as HB^{++} [16], HB-MP [15] and HB^* [24]. All these proposals attempt to prevent man-in-the-middle attacks. In a more recently proposed scheme, i.e. $\text{HB}^\#$ [19], the authors propose a modified version of HB^+ which uses Toeplitz matrices rather than vectors for a shared secret. Under a strong conjecture the scheme is proven secure against a class of man-in-the-middle attacks. In this adversary model which is referred to as *GRS-MIM-model*, the attacker can only modify data transmission from the reader to the tag but not the other way around. This model will protect against the previously mentioned attack.

4 The PUF-HB Authentication Protocol

We start by defining basic notation. In the remainder of the paper we reserve k to denote the security variable. $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ denotes the tag used in the PUF-HB protocol, where $x \in \{0,1\}^k$, $s \in \{0,1\}^{2\times n}$. We treat s_j as a 2-bit vector. and $Y = [y_1, y_2, \ldots, y_{k+1}]$ such that $y_i \in N(0,1)$ and $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. The noise parameter is $\epsilon \in \left(0, \frac{1}{2}\right)$ and n denotes the number of rounds required for authentication. We denote the

reader by $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$, where all the variables are as defined for the tag except l and u which are integers in the range $[0,n]$ such that $l \leq \epsilon \cdot n \leq u$.

With this notation we describe the basic authentication step. In the j^{th} round of the protocol, $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ outputs a randomly generated vector $b_{(j)} \in \{0,1\}^k$ and sends it to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$. The reader responds with the vector $a_{(j)} \in \{0,1\}^k$ and $e_{(j)} \in \{0,1\}^2$. Finally the tag computes $z_{(j)} = b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)} \cdot s_j}(a_{(j)}) \oplus \nu$, where $\cdot$ is the binary inner product, s_j are the j^{th} two bit vector of s and $\nu = 1$ with probability ϵ and $\nu = 0$ with probability $1-\epsilon$. For authentication, this step is repeated n times. In each round the reader checks to see if the tag's response is equal to $b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)} \cdot s_j}(a_{(j)})$. If not, the reader marks the response as wrong. At the end of the n^{th} round, the reader authenticates the tag if and only if the number of wrong responses is in the range $[l,u]$.

In general, any entity can interact with the reader and try to impersonate an honest tag. To capture such interaction, let $\mathcal{E}$ be any entity trying to authenticate itself to the reader $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$. Then PUF-HB$\big(\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big) = 1$ iff $\mathcal{E}$ is authenticated by the reader, and is equal to 0 otherwise. The following protocol formalizes this interaction:

Protocol 1: PUF-HB$\big(\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big)$

1. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ sets $c = 0$ and $j = 1$.
2. $\mathcal{E}$ sends $b_{(j)} \in \{0,1\}^k$ to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$.
3. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ choses $a_{(j)} \in \{0,1\}^k$ and $e_{(j)} \in \{0,1\}^2$ uniformly at random and sends it to $\mathcal{E}$.
4. $\mathcal{E}$ sends $z_{(j)}$ to $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$.
5. If $z_{(j)} \neq b_{(j)} \cdot x \oplus PUF_{Y,e_{(j)} \cdot s_j}(a_{(j)})$ then $c = c + 1$.
6. $\mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}$ increments j and repeats steps 2 through 5 until $j = n$.
7. If $l \leq c \leq u$ then PUF-HB$\big(\mathcal{E}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big) = 1$, otherwise it equals 0.

Provided that $\mathcal{E}$ has no information about x, s or Y, the best probability of being authenticated by the reader will be $\epsilon_s = 2^{-n} \sum_{i=l}^{u} \binom{n}{i}$. This probability represents the soundness error in the algorithm. As for an honest tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ one can see that with a very high probability PUF-HB$\big(\mathcal{T}_{(n,x,Y,s,\epsilon)}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big) = 1$. However, the tag's choice to set $\nu = 1$ with probability ϵ is independent in each round of the authentication. Therefore, it will be possible for the tag to introduce a number of errors which is outside the range $[l,u]$. This will result in a failed authentication session. We denote the probability of this incident by ϵ_c, i.e. the completeness error. If we set $l = 0$ then using the Chernoff bound we can produce the following bound, $\epsilon_c < e^{-(\epsilon n)(u/\epsilon n - 1)^2/4}$.

Protocol (1) would work identically if we run it in a parallel fashion. In this case, the n different $b_{(j)}$ vectors sent in Step 2 would be sent in a single step and similarly all the $a_{(j)}$ vectors and $e_{(j)}$ bits sent in Step 3 would also be sent in a single step. Finally all the $z_{(j)}$ bits returned in Step 4 would be sent in a single step. In general, the PUF-HB protocol is almost identical to the HB$^+$ protocol. The main difference introduced in the PUF-HB protocol is to substitute the inner product $a \cdot y$ where $y \in \{0,1\}^k$ with PUF$_{Y,s}(a)$. As we will see in the proof of Theorem 1 this substitution will not affect the security features introduced by the

HB$^+$ protocol. However, as indicated in the previous sections this substitution will make the tag tamper-resilient, and will simultaneously help resist the known man-in-the-middle attack on the HB$^+$ protocol [18].

5 Security Against Active Attacks

In this section, we reduce the security of the PUF-HB protocol in the active attacker model (which does not include man-in-the-middle attacks) to solving the LPN problem. Note that as we pointed out earlier, the proof here closely follows the proof of Katz et al. on the security of the HB$^+$ protocol [1]. However, due to the nature of the PUF circuit, a very simple part of the original proof in [1], becomes much more complex in our protocol. Such a difference is a reflection of the change from a simple binary inner product to a PUF operation. For a more elaborate explanation of the proof see [1] where the authors prove security for the parallel execution case with $\epsilon < 1/4$. Also see [17] for a similar proof when $\epsilon < 1/2$. Moreover, in the original paper where the HB$^+$ protocol was proposed [14] the authors provide an elegant proof of security against active attacks. The proof in [14] can easily be modified to prove the security of the PUF-HB protocol. However, for simplicity and completeness we use the proof in [1].

We start by quoting the following definitions directly from [1]. Let $A_{x,\epsilon}$ denote an LPN$_\epsilon$ oracle which outputs a $k+1$ bit string such that x is chosen uniformly at random from $\{0,1\}^k$ and $\epsilon \in \left(0, \frac{1}{2}\right)$. The output of the oracle is the string

$$(a, a \cdot x \oplus \nu) \ .$$

Where a is chosen uniformly at random from $\{0,1\}^k$, and $\nu = 1$ with probability ϵ and $\nu = 0$ with probability $1 - \epsilon$. We also define U_{k+1} to be the uniform oracle with outputs from the uniform distribution over $\{0,1\}^{k+1}$. We say that an algorithm M can (t, q, δ) solve the LPN$_\epsilon$ problem if

$$\Pr\left[M^{A_{x,\epsilon}}(1^k) = x\right] \geq \delta \ ,$$

provided that M runs in time t and uses q queries to the oracle $A_{x,\epsilon}$. The main theorem of this paper relies on the following lemma originally due to Regev [30] and reproven in [1].

Lemma 1. ([1]) *If there exists an algorithm D making q oracle queries, and running in time t, such that*

$$\left|Pr\left[D^{A_{x,\epsilon}}(1^k) = 1\right] - Pr\left[D^{U_{k+1}}(1^k) = 1\right]\right| \geq \delta \ ,$$

then there exist an algorithm M making $q' = O(q \cdot \delta^{-2} \log k)$ oracle queries and running in time $t' = O(t \cdot k\delta^{-2} \log k)$, such that

$$Pr\left[M^{A_{x,\epsilon}}(1^k) = x\right] \geq \delta/4 \ .$$

Before we prove the main theorem of the paper we try to give some intuition on why the proof in [1] can be applied to prove the security of the PUF-HB protocol. In addition, we state two technical lemmas which are needed for the proof of the main theorem.

First, note that the function computed in the HB$^+$ protocol is $z = b \cdot x \oplus a \cdot y$ whereas the function computed in the PUF-HB protocol is $z = b \cdot x \oplus PUF_{Y,s}(a)$. Moreover, Theorem 1 states that if there exists an algorithm $\mathcal{A}$ which starts by impersonating a reader to interact with an honest tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ (learning phase), and then impersonates a tag to interact with an honest reader (impersonation phase) therefore achieving $\text{PUF-HB}\big(\mathcal{A}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\big) = 1$ with high probability, then algorithm $\mathcal{A}$ can also be used to distinguish between an LPN oracle and a uniform oracle. As implied by Lemma 1, then algorithm $\mathcal{A}$ can be used to solve the LPN problem.

In the learning phase, the algorithm $\mathcal{A}$ will expect to interact with an honest tag. In the proof, we impersonate such a tag, and use the oracle outputs to substitute the $b \cdot x$ part of a tag's response. Note that this part of the response is in common between the HB$^+$ protocol and the PUF-HB protocol. Now since we will use an oracle for part of the response, this means that we will have no control over x which will be determined by the oracle. At the same time we will have full control over Y, s (y in the HB$^+$ protocol).

In the impersonation phase $\mathcal{A}$ will take the role of a tag interacting with an honest reader, and will therefore attempt to correctly compute the responses $z_{(i)}$. However, the responses of $\mathcal{A}$ will depend on the interaction that took place in the learning phase. If we were successful in providing outputs which were consistent with some x, then $\mathcal{A}$ will be able to provide correct responses in the impersonation phase. This will be the case if the used oracle was an LPN oracle. On the other hand, if the oracle was the uniform oracle, then we will have failed in providing consistent outputs to $\mathcal{A}$. Consequently, the responses $z_{(i)}$ produced by $\mathcal{A}$ in the impersonating phase will be random. While this presents a technique to distinguish between an LPN oracle and a uniform oracle, nevertheless, we do not have a way to know if the responses $z_{(i)}$ were correct or not. Primarily because we have no knowledge of x. To resolve this issue we run the algorithm and acquire the responses for a set of challenges $\{a^1_{(i)}\}$ along with the $\{e^1_{(i)}\}$ bits, then we rewind the algorithm and acquire a second set of responses for a different set of challenges $\{a^2_{(i)}\}$ and the $\{e^2_{(i)}\}$ bits. Adding the two responses cancels out the effect of x and retains the effect of $\text{PUF}_{Y,e^1_{(i)} \cdot s_i}(a^1_{(i)}) \oplus \text{PUF}_{Y,e^2_{(i)} \cdot s_i}(a^2_{(i)})$ (this would be $a^1_{(i)} \cdot y \oplus a^2_{(i)} \cdot y$ for the HB$^+$ protocol). With this trick we will have complete knowledge over the remaining variables. Therefore, we can check whether the responses retuned by $\mathcal{A}$ were correct or not.

What remains to be shown is that given the inputs $a^1_{(i)}$ and $a^2_{(i)}$ and no other information, the algorithm $\mathcal{A}$ will not be able to predict the output bits $\text{PUF}_{Y,e^1_{(i)} \cdot s_i}(a^1_{(i)})$ and $\text{PUF}_{Y,e^2_{(i)} \cdot s_i}(a^2_{(i)})$. This is akin to asking the question: How much can be inferred about the output, by only knowing the input. In the HB$^+$ protocol this becomes a question of linear independence. However, in the case of PUF-HB the answer becomes much more complicated. This question will be

addressed in Lemma 2 and 3. In general, one can see that the function $\mathrm{PUF}_{Y,s}(a)$ (compared to $a \cdot y$) only becomes relevant toward the end of the proof of Theorem 1. In fact, it should be clear that there is a large family of functions that could be used in place of $\mathrm{PUF}_{Y,s}(a)$ or $a \cdot y$ while maintaining the security of the protocol. However, our choice of $\mathrm{PUF}_{Y,s}(a)$ was mainly motivated by hardware simplicity and tamper resilience. The following Lemma characterizes the correlation between two PUF outputs obtained from different input challenges.

Lemma 2. *Given two k-bit strings $a_{(1)}$ and $a_{(2)}$ chosen independently and uniformly at random from $\{0,1\}^k$, and $Y_0 = [y_1, \ldots, y_{k+1}]$ where $y_i \in N(0,1)$ for $i = 1, \ldots, k$ and $y_{k+1} = 0$, then for $z_{(1)} = PUF_{Y_0}(a_{(1)})$ and $z_{(2)} = PUF_{Y_0}(a_{(2)})$ the probability that $z_{(1)}$ and $z_{(2)}$ are equal is*

$$Pr[z_{(1)} = z_{(2)}] = F_k(d) = 1 - \frac{2}{\pi} \arctan\left(\sqrt{\frac{d}{k-d}}\right). \tag{3}$$

Where d is the Hamming distance between $P_{(1)}$ and $P_{(2)}$ and $(P_{(i)} = U a_{(i)})$. The strings $a_{(i)}$ are represented as column vectors, U is the upper triangular matrix with all non-zero entries equal to 1 and the matrix multiplication is performed modulo 2.

The proof of Lemma 2 can be found in Appendix A. The next lemma, connects Lemma 2 to the main theorem.

Lemma 3. *Let $\mathcal{A}$ be an adversary who is given n strings $\{a_{(i)}\}_{i=1}^n$, where $a_{(i)}$ is chosen independently and uniformly at random from $\{0,1\}^k$. $\mathcal{A}$ also knows that s is chosen uniformly at random form $\{0,1\}^n$ and that $Y_0 = [y_1, \ldots, y_{k+1}]$ where $y_i \in N(0,1)$ for $i = 1, \ldots, k$ and $y_{k+1} = 0$. Let $z_{(i)} = PUF_{Y_0,s_i}(a_{(i)})$ then the bits $z_{(i)}$ will be uniform and independent (from the point of view of $\mathcal{A}$).*

The proof of Lemma 3 is also moved to Appendix A. We are now ready to prove the main theorem.

Theorem 1 (Compare to Theorem 3 in [1]). *If there exists an adversary $\mathcal{A}$ interacting with a tag $\mathcal{T}_{(n,x,Y,s,\epsilon)}$ in at most q executions of the PUF-HB protocol (possibly concurrently), running in time t such that*

$$Pr\left[\mathcal{A}^{\mathcal{T}_{(n,x,Y,s,\epsilon)}}(1^k) : PUF\text{-}HB\left(\mathcal{A}, \mathcal{R}_{(n,x,Y,s,\epsilon,l,u)}\right) = 1\right] \geq \delta \quad ,$$

then these exist an algorithm D making $q \cdot n$ oracle queries, running in time $O(t)$, and such that

$$\left|Pr\left[D^{A_{x,\epsilon}}(1^k) = 1\right] - Pr\left[D^{U_{k+1}}(1^k) = 1\right]\right| \geq \delta^2 - 2^{-n/2} \sum_{i=0}^{2u} \binom{n/2}{i} - e^{-\frac{n}{8}}$$

Therefore, for any $\epsilon < \frac{1}{8}$ there is an appropriate choice of n, u such that the last two terms become negligible, and thus we can conclude that the PUF-HB protocol is secure assuming the hardness of the LPN_ϵ problem.[6]

[6] Note that by parametering the length ℓ of the strings $s_i \in \{0,1\}^\ell$ and $e_{(i)} \in \{0,1\}^\ell$ we may achieve some flexibility in the parameters, i.e. $\epsilon < 0.25 - 2^{-(\ell+1)}$.

Proof. To prove the theorem we show a construction of the algorithm D. As stated by the theorem, D is given access to an oracle returning $(k+1)$-bit strings which can be broken to $(\bar{b}, \bar{z})$, where $\bar{b} \in \{0,1\}^k$ and $\bar{z} \in \{0,1\}$. D proceeds as follows:

1. D starts by choosing vectors Y_0 and s such that y_{k+1} is set to 0, and then the k remaining y_i values are chosen from $N(0,1)$. The bitstring s is chosen uniformly at random from $\{0,1\}^{2\times n}$. D runs the algorithm $\mathcal{A}$ which will expect to interact with a PUF-HB tag. In order to impersonate a real tag, D does the following to simulate a basic authentication step: D starts by obtaining a $k+1$ bit string $(\bar{b}_{(i)}, \bar{z}_{(i)})$ from the oracle, and then sends $\bar{b}_{(i)}$ to $\mathcal{A}$ as the initial b in Protocol 1. $\mathcal{A}$ will reply with a challenge $\bar{a}_{(i)}$ and the bits $\bar{e}_{(i)}$. Next, D computes $z_{(i)} = \bar{z}_{(i)} \oplus \mathrm{PUF}_{Y_0, \bar{e}_{(i)} \cdot s_i}(\bar{a}_{(i)})$ and sends $z_{(i)}$ back to $\mathcal{A}$. D repeats this step $q \cdot n$ times.
2. In the second phase of the algorithm, $\mathcal{A}$ tries to impersonate an honest tag. Looking at the parallel execution of PUF-HB, $\mathcal{A}$ starts by sending $b_{(1)}, \ldots, b_{(n)} \in \{0,1\}^k$ to a reader. Next, D randomly choses $a^1_{(1)}, \ldots, a^1_{(n)} \in \{0,1\}^k$ and $e^1_{(1)}, \ldots, e^1_{(n)} \in \{0,1\}^2$ and send them back to $\mathcal{A}$, which will in turn respond with the bits $z^1_{(1)}, \ldots, z^1_{(n)}$. D then rewinds $\mathcal{A}$ and sends randomly chosen $a^2_{(1)}, \ldots a^2_{(n)} \in \{0,1\}^k$ and $e^2_{(1)}, \ldots, e^2_{(n)} \in \{0,1\}^2$. $\mathcal{A}$ will respond with $z^2_{(1)}, \ldots, z^2_{(n)}$. Note that since the algorithm was rewound the same b values will be sent by $\mathcal{A}$.
3. D calculates $z^{\oplus}_{(i)} = z^1_{(i)} \oplus z^2_{(i)}$ and lets $Z^{\oplus} = (z^{\oplus}_{(1)}, \ldots, z^{\oplus}_{(n)})$. D also computes $\hat{z}_{(i)} = \mathrm{PUF}_{Y_0, e^1_{(i)} \cdot s_i}(a^1_{(i)}) \oplus \mathrm{PUF}_{Y_0, e^2_{(i)} \cdot s_i}(a^2_{(i)})$ and lets $\hat{Z} = (\hat{z}_{(1)}, \ldots, \hat{z}_{(n)})$. D outputs 1 iff $Z^{\oplus}$ and $\hat{Z}$ differ in at most $2u$ positions.

Now we analyze D:

Case 1: If the oracle used by D was the uniform oracle U_{k+1}, then the outputs $\bar{z}$ given to D in step 1 were uniformly distributed and independent of everything. This means that the bits $z_{(i)}$ which D sent back to $\mathcal{A}$ in step 1 were also uniformly distributed and independent of everything. Therefore, by the end of step 1 $\mathcal{A}$ has no information about either Y_0 or s. All $\mathcal{A}$ receives in the second step are the inputs $\{a^1_{(i)}\}^n_{i=1}$, $\{e^1_{(i)}\}^n_{i=1}$ and $\{a^2_{(i)}\}^n_{i=1}$, $\{e^2_{(i)}\}^n_{i=1}$. All of these inputs are uniformly and independently distributed. As shown in Lemma 3 each of the two calculated output strings $\{\mathrm{PUF}_{Y_0, e^1_{(i)} \cdot s_i}(a^1_{(i)})\}^n_{i=1}$ and $\{\mathrm{PUF}_{Y_0, e^2_{(i)} \cdot s_i}(a^2_{(i)})\}^n_{i=1}$ will be uniform over $\{0,1\}^n$ (from the point of view of $\mathcal{A}$). However, when we add these two variables and obtain $\hat{Z}$ the individual bits of the output will not always be independent. The affect of the s_i bits will actually cancel out from both terms when $e^1_{(i)} = e^2_{(i)}$ with probability 0.25. To simplify the proof, we assume in this case that the outputs are completely dependent. Using the Chernoff approximation we bound the probability of observing more than $n/2$ dependent output bits by $e^{-\frac{n}{8}}$. The probability that $Z^{\oplus}$ and $\hat{Z}$ differ in at most $2u$ positions is exactly $2^{-n/2} \sum_{i=0}^{u} \binom{n/2}{i}$. We conclude that D outputs 1 in this case with probability at most $2^{-n/2} \sum_{i=0}^{2u} \binom{n/2}{i} + e^{-\frac{n}{8}}$.

Case 2: If D is using the oracle $A_{x,\epsilon}$ for some random x, the simulation provided by D in the first phase will be perfect and therefore $\mathcal{A}$ will be able to impersonate an honest tag with probability at least δ. Let w be the randomness used in the first phase of running $\mathcal{A}$, then we denote the probability that $\mathcal{A}$ succeeds in impersonating an honest tag for a fixed choice of w by δ_w. Now since we rewind the algorithm, the probability that $\mathcal{A}$ succeeds in both rounds is δ_w^2. Let $E(\delta_w^2)$ denote the expected value of δ_w^2 over all possible randomness w, then we have

$$E(\delta_w^2) \geq E(\delta_w)^2 = \delta^2,$$

where the square is taken out of the expected value operator using Jensen's inequality. Now assuming that $\mathcal{A}$ succeeds in impersonating an honest tag for both rounds, then we would expect each of the response strings $z^1_{(1)}, \ldots, z^1_{(n)}$ and $z^2_{(1)}, \ldots, z^2_{(n)}$ to have at most u errors. Therefore $Z^{\oplus}$ will in turn have at most $2u$ errors.[7] When we add $z^1_{(i)}$ and $z^2_{(i)}$ to generate $z^{\oplus}_i$ we are canceling the effect of $b \cdot x$ and therefore we are left with $z^{\oplus}_{(i)} = \mathrm{PUF}_{Y_0, e^1_{(i)} \cdot s_i}(a^1_{(i)}) \oplus \mathrm{PUF}_{Y_0, e^2_{(i)} \cdot s_i}(a^2_{(i)})$. With the exception of the $2u$ errors in $Z^{\oplus}$, the strings $\hat{Z}$ and $Z^{\oplus}$ are calculating the same output. We conclude that D outputs 1 in this case with probability at least δ^2. □

This concludes our security proof, and shows that the PUF-HB protocol is secure against an active attacker provided that the LPN problem is hard to solve.

6 Man-in-the-Middle Attacks

The main weakness of the HB$^+$ protocol is the man-in-the-middle attack proposed in [18]. Briefly summarized, in this attack an adversary replaces all the challenges $\{a_{(j)}\}_{j=1}^n$ sent from the reader in a single authentication session by $\{a_{(j)} \oplus w\}_{j=1}^n$ where $w \in \{0,1\}^k$. The attacker knows that the challenges will interact with the secret y through $a_{(j)} \cdot y$. At the end of the n rounds, if the reader authenticates the tag, the adversary can deduce with very high probability that his changes did not affect the responses of the tag, and therefore $w \cdot y = 0$. On the other hand, if the reader rejects the tag, then the adversary will know with a very high probability that $w \cdot y = 1$. Repeating the same attack k times will allow the adversary to collect k linear equations in y. The adversary can use Gaussian elimination to recover the secret y. Similarly, the same attack can be carried out to deduce the other secret string x.

In the PUF-HB scheme this particular man-in-the-middle attack will not succeed due to the non-linearity of the PUF function. From Lemma 2 we know that the only type of correlations that the attacker can exploit are those related to the Hamming distance between the different input strings a. However, we saw in Lemma 3 that with the secret string s the Hamming distance information is

[7] This worst case happens when the u errors in $z^1_{(i)}$ and $z^2_{(i)}$ affect completely different bits.

masked for a single authentication session. It is still possible that an adversary can exploit Hamming distances between different sessions to launch an attack. Another potential point of weakness is the linearity in the $b \cdot x$ portion of the PUF-HB protocol. To protect against simple attacks exploiting this linearity, a second PUF circuit can be used with the b vector as its input. We label such a protocol PUF^2-HB, since it will essentially be identical to Protocol 1, with the only difference in the z bit calculated by the tag, which becomes

$$z_{(i)} = b_{(i)} \cdot x \oplus PUF_{Y_1,s_i}(a_{(i)}) \oplus PUF_{Y_2}(b_{(i)}) \oplus \nu \tag{4}$$

where the shared secret becomes (x, Y_1, Y_2, s).

7 Hardware Security

In the previous section we discussed the security of the proposed scheme under abstract security models. However, in recent years we have seen numerous side-channel attacks which directly target the hardware implementation. The PUF paradigm was aimed at protecting against active side-channel attacks. In the PUF-HB protocol there are only two strings that are to be stored by the tag: x and s. The secret Y is not really stored since it is part of the characteristics of the circuit itself. In Section 2 we discussed the resilience of PUF circuits against hardware attacks. In particular the PUF prevents an attacker from measuring the y_i parameters directly via a physical measurement. Any major changes to the surrounding temperatures or voltage levels, or any attempt to forcefully read the value of these registers will induce a change to the PUF, therefore changing the identity of the tag.

What is more impressive about the PUF circuit is that it even protects neighboring components. This is achieved by placing all registers containing the secret strings x and s sufficiently close to the PUF circuit. We have experimentally verified these claims on an FPGA hardware implementation of a PUF. Such a level of security ensures that even when the tag itself is compromised, an attacker cannot impersonate this tag by extracting the secrets from the hardware. Naturally, one might be concerned about how that will affect the modelability of the PUF in the pre-deployment phase. Before deployment of the tag, the registers are initialized to their secret values. Afterward, with the knowledge of the secret vectors the reader can develop an accurate model for the PUF circuit. Note that modeling the PUF is even possible in the existence of noise [32]. Hence, the sensitivity of the PUF does not prevent the owner from modeling it.

Finally we would like to underline that PUF-HB is not inherently protected against passive attacks, e.g. Simple Power Analysis and Differential Power Analysis. Although not trivial, side-channel profiles may be utilized to recover the secret values. If passive side-channel attacks are a concern, standard IC level power balancing techniques [33,34,35] must be employed. Although effective, these techniques tend to incur significant area overhead. An alternative approach would be to modify the implementation to balance the power consumption.

8 Conclusion

In this paper we merged the PUF authentication scheme with the HB based authentication protocol, with the goal of producing a hybrid protocol which enjoys the advantages of both schemes while improving the level of security. The main contribution of this paper is the proposed PUF-HB authentication scheme which is tamper resilient, and at the same time provably secure against active attacks in the detection based model. In addition, the proposed protocol resists the known man-in-the-middle attacks against the HB^+ scheme. From the PUF perspective, the protocol is the first PUF based authentication scheme with a security reduction. From the HB perspective, this is the first tamper-resilient HB-based authentication protocol. From a more practical perspective, the proposed scheme provides low-cost, tamper-resilient, and provably secure authentication.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants No. ANI-0133297 (NSF CAREER Award) and CNS-0716306. We would like to thank Dr. Stephen Weis for clarifications regarding the proof of security in [14] and for pointing us to [1], and Prof. William J. Martin for his comments on an earlier draft of the paper.

References

1. Katz, J., Shin, J.S.: Parallel and Concurrent Security of the HB and HB^+ Protocols. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 73–87. Springer, Heidelberg (2006)
2. Want, R.: An Introduction to RFID Technology. IEEE Pervasive Computing 5(1), 25 (2006)
3. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
4. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
5. Kulikowski, K.J., Karpovsky, M.G., Taubin, A.: DPA on faulty cryptographic hardware and countermeasures. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 211–222. Springer, Heidelberg (2006)
6. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Delay-based Circuit Authentication and Applications. In: Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 294–301 (2003)
7. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: CCS 2002: Proceedings of the 9th ACM conference on Computer and communications security, pp. 148–160. ACM Press, New York (2002)
8. Lee, J.W., Daihyun, L., Gassend, B., S., G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: Symposium of VLSI Circuits, pp. 176–179 (2004)

9. O'Donnell, C.W., Suh, G.E., Devadas, S.: PUF-based random number generation. Number 481 (November 2004)
10. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Trans. VLSI Syst. 13(10), 1200–1205 (2005)
11. Ozturk, E., Hammouri, G., Sunar, B.: Physical unclonable function with tristate buffers. In: The Proceedings of The IEEE International Symposium on Circuits and Systems 2008 – ISCAS (to appear, 2008)
12. Ozturk, E., Hammouri, G., Sunar, B.: Towards robust low cost authentication for pervasive devices. In: PERCOM 2008: Proceedings of the Sixth IEEE International Conference on Pervasive Computing and Communications (2008)
13. Hopper, N.J., Blum, M.: Secure Human Identification Protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
14. Juels, A., Weis, S.A.: Authenticating Pervasive Devices with Human Protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)
15. Munilla, J., Peinado, A.: HB-MP: A further step in the HB-family of lightweight authentication protocols. Comput. Networks 51(9), 2262–2267 (2007)
16. Bringer, J., Chabanne, H., Dottax, E.: HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In: SECPERU 2006: Proceedings of the Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, Washington, DC, USA, 2006, pp. 28–33. IEEE Computer Society, Los Alamitos (2006)
17. Katz, J., Smith, A.: Analyzing the HB and HB^{+} protocols in the "large error" case. In: Cryptology ePrint Archive, Report 2006/326 (2006), `http://eprint.iacr.org/`
18. Gilbert, H., Robshaw, M., Sibert, H.: An Active Attack Against HB^{+} - A Provably Secure Lightweight Authentication Protocol. IEE Electronic Letters 41,21, 1169–1170 (2005)
19. Gilbert, H., Robshaw, M., Seurin, Y.: $HB^{\#}$: Increasing the Security and Efficiency of HB^{+}. In: Advances in Cryptology: EUROCRYPT 2008. LNCS, vol. 4965, Springer, Heidelberg (2008)
20. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: STOC 2000: Proceedings of the thirty-second annual ACM symposium on Theory of computing, pp. 435–440. ACM Press, New York (2000)
21. Fossorier, M., Mihaljevic, M., Imai, H., Cui, Y., Matsuura, K.: A Novel Algorithm for Solving the LPN Problem and its Application to Security Evaluation of the HB Protocol for RFID Authentication. In: Proc. of INDOCRYPT, vol. 6, pp. 48–62.
22. Levieil, E., Fouque, P.: An Improved LPN Algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
23. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subsetsum problem. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX 2005 and RANDOM 2005. LNCS, vol. 3624, pp. 378–389. Springer, Heidelberg (2005)
24. Duc, D., Kim, K.: Securing HB^{+} Against GRS Man-in-the-Middle Attack. In: Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security, January 2007, pp. 23–26 (2007)

25. Berlekamp, E.R., Mceliece, R.J., van Tilborg, H.C.: On the Inherent Intractability of Certain Coding Problems. IEEE Transactions on Information Theory 24(3), 384–386 (1978)
26. Kearns, M.: Efficient Noise-Tolerant Learning from Statistical Queries. In: STOC 1993: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, pp. 392–401. ACM Press, New York (1993)
27. Roos, C., Terlaky, T., Vial, J.-P.: Interior Point Methods for Linear Optimization, 2nd edn. Springer, Heidelberg (2005)
28. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. Mathematical Programming 71(2), 221–245 (1995)
29. Agmon, S.: The relaxation method for linear inequalities. Canadian J. of Mathematics, 382–392 (1964)
30. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pp. 84–93. ACM Press, New York (2005)
31. Prudnikov, Y.A., Brychkov, A.P., Marichev: Integrals and Series, vol. 2: Special Functions. In: Gordon and Breach (1990)
32. Blum, A., Frieze, A.M., Kannan, R., Vempala, S.: A polynomial-time algorithm for learning noisy linear threshold functions. Algorithmica 22(1/2), 35–52 (1998)
33. Tiri, K., Akmal, M., Verbauwhede, I.: A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In: Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European, pp. 403–406 (2002)
34. Toprak, Z., Leblebici, Y.: Low-power current mode logic for improved DPA-resistance in embedded systems. In: Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium, pp. 1059–1062 (2005)
35. Regazzoni, F., Badel, S., Eisenbarth, T., Grobschadl, J., Poschmann, A., Toprak, Z., Macchetti, M., Pozzi, L., Paar, C., Leblebici, Y., Ienne, P.: A Simulation-Based Methodology for Evaluating the DPA-Resistance of Cryptographic Functional Units with Application to CMOS and MCML Technologies. In: IC-SAMOS 2007, pp. 209–214 (2007)

A Appendix 1

Proof of Lemma 2

Proof. First recall from Section 2 that for any string $c = [c_1, \ldots c_n]$ the PUF response is

$$\mathrm{PUF}_Y(c) = \mathrm{sign}\left(\sum_{i=1}^{k}(-1)^{p_i} y_i + y_{k+1}\right),$$

where $p_i = c_i \oplus \ldots \oplus c_k$ and $P = [p_1, \ldots, p_k]$, as noted above ($P = Uc$). This mapping is a linear bijection and will therefore preserve uniformity and independence. Since $a_{(1)}$ and $a_{(2)}$ were chosen uniformly at random from $\{0,1\}^k$, the same can be said about the distribution of the corresponding P vectors $P_{(1)} = [p_1^{(1)}, \ldots p_k^{(1)}]$ and $P_{(2)} = [p_1^{(2)}, \ldots p_k^{(2)}]$. Let $Z(P_{(j)}, Y_0) = \sum_{i=1}^{k}(-1)^{p_i^{(j)}} y_i$, and let d be the Hamming distance between $P_{(1)}$ and $P_{(2)}$. Without loss of generality we may assume that the different bits of $P_{(1)}$ and $P_{(2)}$ are in the first d

bit positions. Now we can write

$$Z(P_{(1)}, Y_0) = \sum_{i=1}^{d} (-1)^{p_i^{(1)}} y_i + \sum_{i=d+1}^{k} (-1)^{p_i^{(1)}} y_i = D_d + S_{k-d}$$
$$Z(P_{(2)}, Y_0) = -D_d + S_{k-d} \ ,$$

where $D_d = \sum_{i=1}^{d} (-1)^{p_i^1} y_i$ and $S_{k-d} = \sum_{i=d+1}^{k} (-1)^{p_i^1} y_i$. Note that since $z_{(j)} = \text{sign}(Z(P_{(j)}, Y_0))$, then for $z_{(1)}$ to be equal to $z_{(2)}$ we need $Z(P_{(1)}, Y_0)$ and $Z(P_{(2)}, Y_0)$ to have the same sign. For this, D_d needs to have a smaller magnitude than S_{k-d} which means $|D_d| < |S_{k-d}|$. Therefore

$$\Pr[z_{(1)} = z_{(2)}] = \Pr[|S_{k-d}| - |D_d| > 0] = \Pr[R_d > 0],$$

where $R_d = |S_{k-d}| + (-|D_d|)$. Let $f_R(R_d), f_D(D_d)$ and $f_S(S_{k-d})$ represent the probability distribution function (PDF) for each of the random variables R_d, D_d and S_{k-d} respectively. Each term in the summations making up D_d and S_{k-d} involves one of the bits $p_i^{(1)}$ and the real value y_i. Since $y_i \in N(0,1)$ has mean zero and is symmetric around the y-axis, we can easily see that multiplying with $(-1)^{p_i^{(1)}}$ will not affect the normal distribution and therefore $(-1)^{p_i^{(1)}} y_i \in N(0,1)$. Now each of the variables D_d and S_{k-d} is a summation of respectively d and $k-d$ normal distributions $N(0,1)$. Thus, $f_D(D_d) = N(0,d)$ and $f_S(S_{k-d}) = N(0,k-d)$.[8] We are interested in the PDF of $-|D_d|$ and $|S_{k-d}|$. These can easily be calculated as follows:

$$f_{|S|}(x) = \left\{ \begin{matrix} 2N(0,k-d) & x > 0 \\ 0 & x \leq 0 \end{matrix} \right\} = \left\{ \begin{matrix} 2\frac{e^{\frac{-x^2}{2(k-d)}}}{\sqrt{2\pi(k-d)}} & x > 0 \\ 0 & x \leq 0 \end{matrix} \right. ,$$

and

$$f_{-|D|}(x) = \left\{ \begin{matrix} 0 & x > 0 \\ 2N(0,d) & x \leq 0 \end{matrix} \right\} = \left\{ \begin{matrix} 0 & x > 0 \\ 2\frac{e^{\frac{-x^2}{2d}}}{\sqrt{2\pi d}} & x \leq 0 \end{matrix} \right. .$$

Now we can calculate the desired probability

$$\begin{aligned} \Pr[z_{(1)} = z_{(2)}] &= \Pr[R_d > 0] \\ &= \int_0^\infty f_R(w)\ dw = \int_0^\infty f_{-|D|}(w) * f_{|S|}(w)\ dw \\ &= \int_0^\infty \frac{4}{2\pi\sqrt{d(k-d)}} \cdot \\ &\qquad \left[\int_{-\infty}^{\infty} e^{\frac{-x^2}{2(k-d)}} \cdot U(x) \cdot e^{\frac{-(w-x)^2}{2d}} \cdot U(x-w)\ dx \right] dw \end{aligned}$$

[8] This is a straightforward application of the Central Limit Theorem. It is also very easy to derive directly from the PDF of a normal random variable.

where $*$ denotes the convolution operator and $U(x)$ is the unit step function. By rearranging the terms we obtain

$$\begin{aligned}\Pr[z_{(1)} = z_{(2)}] &= \frac{4}{\sqrt{2\pi(k+1)}} \int_0^\infty \left(\frac{1}{\sqrt{2\pi\sigma^2}} \int_w^\infty e^{\frac{-(x-\mu)^2}{2\sigma^2}}\, dx \right) e^{\frac{-w^2}{2k}}\, dw \\ &= 2 \int_0^\infty \frac{1}{\sqrt{2\pi k}} \left[1 - \mathrm{erf}(\alpha w)\right] e^{\frac{-w^2}{2k}}\, dw \\ &= 2 \left[\int_0^\infty \frac{e^{\frac{-w^2}{2k}}}{\sqrt{2\pi k}}\, dw - \int_0^\infty \frac{e^{\frac{-w^2}{2k}} \cdot \mathrm{erf}(\alpha w)}{\sqrt{2\pi k}}\, dw \right] \\ &= 2 \left[\frac{1}{2} - \frac{1}{\pi} \arctan\left(\alpha\sqrt{2k}\right) \right]\end{aligned}$$

where $\sigma^2 = \frac{d(k-d)}{k}$, $\mu = \frac{\sigma^2 w}{d}$, $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$ and $\alpha = \sqrt{\frac{d}{2k(k-d)}}$. The first of the two integrations is just a Gaussian over half of the space. As for the second integration this is a known definite integral [31]. Finally, substituting back with the original variables k and d we obtain

$$\Pr[z_{(1)} = z_{(2)}] = F_k(d) = 1 - \frac{2}{\pi} \arctan\left(\sqrt{\frac{d}{k-d}} \right) .$$

□

Proof of Lemma 3

Proof. To show that the bits $\{z_{(i)}\}_{i=1}^n$ are uniform and independent, we need to show that the probability of $z_{(i)} = 0$ for any $i \in [1, n]$ is 0.5 and that the probability of $z_{(i)} = z_{(j)}$ for any $i, j \in [1, n]$ is 0.5. It is clear that when $y_{k+1} = 0$ the output of a PUF will be balanced. Therefore, it is straightforward to see that when $\{a_{(i)}\}_{i=1}^n$ are independent and chosen uniformly at random the bits $\{z_{(i)}\}_{i=1}^n$ will also be uniformly distributed. What remains to show is that there is no correlation between the bits $\{z_{(i)}\}_{i=1}^n$.

From Lemma 2 and as can be seen from Equation 3 the probability of any two PUF outputs being equal (or not equal) depends on the Hamming distance d between $P_{(i)} = Ua_{(i)}$ and $P_{(j)} = Ua_{(j)}$ and not the specific values of $a_{(i)}$ and $a_{(j)}$. Furthermore, we can deduce from Equation 3 that

$$\overline{F}_k(d) = \Pr[z_{(i)} \neq z_{(j)}] = 1 - F_k(d) = \frac{2}{\pi} \arctan\left(\sqrt{\frac{d}{k-d}} \right) .$$

The two probability distributions $F_k(d)$ and $\overline{F}_k(d)$ are reflections of each other around $d = \frac{k}{2}$. Therefore, $\overline{F}_k(d) = F_k(k-d)$. This means that when the probability distribution of the Hamming distance between $P_{(i)}$ and $P_{(j)}$ is symmetrized around $\frac{k}{2}$, then $z_{(i)}$ and $z_{(j)}$ will be uncorrelated. To prove the rest of the lemma, we only need to show that the probability distribution of the Hamming distances between the different $P_{(i)}$ strings will be symmetric around $\frac{k}{2}$.

The n bit strings $\{a_{(i)}\}_{i=1}^{n}$ given to $\mathcal{A}$ will induce $\frac{n(n-1)}{2}$ different Hamming distances between the corresponding $P_{(i)}$ strings. Recall from the lemma that $z_{(i)} = PUF_{Y_0,s_i}(a_{(i)})$. Therefore, as indicated by Equation 2 the PUF circuit will invert the $P_{(i)}$ strings based on the value of the bit s_i. From $\mathcal{A}$'s perspective s is chosen uniformly at random from $\{0,1\}^n$. Therefore, each of the $P_{(i)}$ strings will be inverted with probability 0.5. For any two strings $P_{(i)}$ and $P_{(j)}$, if both of the strings or neither of them are inverted the Hamming distance d will not be affected. On the other hand, if only one of the two strings is inverted then the Hamming distance d will become $k - d$. Therefore, the Hamming distance between any two strings $P_{(i)}$ and $P_{(j)}$ will be d with probability 0.5 and will be $k - d$ with probability 0.5. This distribution is symmetric around $\frac{k}{2}$. □