Unclonable Lightweight Authentication Scheme*

Ghaith Hammouri, Erdinç Öztürk, Berk Birand, and Berk Sunar

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

Abstract. We propose a lightweight, tamper-resilient challenge-response authentication scheme. The scheme that we propose (HB+PUF) is a variant of the PUF-HB protocol [1] which utilizes Physically Unclonable Functions (PUFs). We reduce the security of (HB+PUF) in the active attacker model to solving the LPN problem. The proposed scheme enjoys strong tamper-resilience due to the PUF properties. We present a proof of concept implementation for the proposed protocol. To generate the random bits needed for the protocol, we reuse the PUF circuit as a Random Number Generator (RNG). This construction shows to be cost-effective since we will be using the same hardware for authentication as well as random number generation without incuring any significant overhead. The overall scheme including storage, tamper-resilience and RNG can be achieved with less than 1000 gates. The small footprint should be ideal for constrained environments such as RFID's, smart cards, and sensor networks.

Keywords: Provable security, tamper-resilience, lightweight, random number generation, PUF, HB+.

1 Introduction

Lightweight cryptography has been receiving more attention in the past few years [10,21]. This attention is mainly motivated by the boom in the next generation ubiquitous networks. With highly constrained devices such as wireless sensor nodes, RF identification devices (RFIDs), and smartcards as their main building block, these networks pose an urgent need for affordable cryptography. A number of the known results so far reduce the size of the hardware by serializing classical cryptographic protocols, thus decreasing the circuit's footprint [11,22,37,28]. For example, in [37] and [28] the authors present a lightweight implementation of DESL, a variant of DES. The presented function uses a single S-box 8 times, rather than using the 8 S-boxes needed for DES. With this reduction, the authors manage to serialize the implementation, therefore decreasing the footprint. Although these results are very exciting, the approach itself seems to be inherently limited. Most classical cryptographic protocols were designed

* This material is based upon work supported by the National Science Foundation under Grants No. ANI-0133297 (NSF CAREER Award) and CNS-0716306.

L. Chen, M.D. Ryan, and G. Wang (Eds.): ICICS 2008, LNCS 5308, pp. 33–48, 2008.

with little attention to the hardware implementation. Therefore, the protocol may not lend itself to serialization. Even from a broader perspective, the notion of taking classical protocols and attempting to squeeze them into a smaller circuit seems to eventually face natural limitations. A different yet exciting approach is to explore new protocols which are motivated by the hardware and which are lightweight in nature [6,31,17,41]. This approach seems harder due to the typical computational demands of cryptography. Nevertheless, it also seems more likely to provide a fundamental solution. This point of view becomes stronger when we consider the diversity of the attacks that target the hardware. While typical cryptographic protocols might be very secure in theory, one has to take into account attacks which exploit so called side-channels [26,25]. These attacks are roughly 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. In contrast, in active attacks the attacker may also inject faults during the computation [27]. Not surprisingly, active attacks are more powerful and are more difficult to prevent. A tamper-resilient hardware could help in securing devices from both passive and active side-channel attacks. With all this in mind, it becomes vital to introduce secure lightweight cryptographic protocols which are motivated by the hardware, and which are tamper-resilient.

Two promising candidates for such a task are Physically Unclonable Functions (PUFs), and the HB-based authentication family. A PUF is a physical pseudo-random function which exploits the small variances in the wire and gate delays inside an integrated circuit (IC). Even when the ICs are designed to be logically identical, the sensitive delay variances will be unique to each IC. These properties will result in providing the hardware with a level of tamper-resilience. However, these devices are not known to be provably secure. On the other hand the HB-based protocols base their security on the hardness of the learning parity with noise (LPN) problem which is known to be NP-hard [4]. Although the HB-based authentication schemes are provably secure in strong settings, they all lack any notion of tamper-resilience. More recently, PUF-HB was proposed [1]. This scheme aimed at merging the two qualities of PUF and HB to produce an interesting hybrid. However, the work presented in [1] lacked any implementation results.

Our contribution: We present a proof of concept implementation for HB+PUF, a variant of PUF-HB which is also provably secure with a much simpler security reduction. The reduction only applies to active attacks which do not have a full man-in-the-middle control of the communication channel. HB+PUF resists the known man-in-the-middle attacks against the HB^+ scheme, and shows strong indications to resist a variety of these attacks. Due to the lightweight nature of PUFs and the HB^+ protocol the presented scheme proves suitable for lightweight applications. Our implementation takes advantage of the PUF circuit in order to produce the random bits typically needed for an HB-based authentication scheme. Note that the existence of a random number generator (RNG) is

assumed in all HB-based protocols without taking into account the complexity of such a device. The overall circuit is shown to occupy less than 1000 gates.

The remainder of the paper is organized as follows. In Section 2 we review the PUF paradigm focusing on tristate PUF circuits. In Section 3 we give a review of the known HB-based authentication protocols. In Section 4 we define our notation and describe our proposed protocol. The security analysis is presented in Section 5. In Section 6 we describe our RNG construction and present analysis of its randomness. Section 7 describes the proof of concept implementation of the entire circuit. Finally, we present the conclusion in Section 8.

2 PUF

The idea behind a PUF is to have identical logical circuits observe different input-output behavior. This goal is achieved by taking advantage of the interchip variations which will vary from one circuit implementation to another. These variations are directly related to physical aspects of the fabrication environment. The reason one might seek to explore such a circuit is to prevent any ability to clone the system. Additionally, because of the high sensitivity of these interchip variations, it becomes difficult for an attacker to accurately reproduce the hardware. Another major advantage of the PUF's sensitivity is to prevent physical attacks on the system. Trying to tap into the circuit will cause a capacitance change therefore changing the output of the circuit. Similarly, trying to remove the outer layer of the chip will result in a change in the output of the circuit. We mention here that there are different realizations of PUFs. For example, surface PUFs were proposed in [36,44] and further developed in [40]. In this paper we focus our attention on the delay-based PUF first introduced in [14]. For the background information we closely follow the presentation of [1].

A delay-based PUF is a $\{0,1\}^n \rightarrow \{0,1\}$ mapping, that takes a n-bit challenge (a) and produces a single bit output (r). The basic idea of a PUF circuit is to create a race between two signals which originate from the same source. The original PUF circuit proposed in [14] utilizes multiplexers (MUXs) to implement the needed switches. In this work we use a different implementation of PUFs, in particular we use tristate PUFs proposed in [9]. Instead of having two interleaved delay paths, the tristate PUF uses two separate paths of delay units. This realization of a PUF has the advantage of requiring less gates and consuming less power. As shown in Figure 1 the delay units of a tristate PUF are constructed by respectively connecting the input and the output ports of two tristate buffers. The enable ports of these tristate buffers are connected to each other, with one of the buffers having an inverted enable input. This assures that only one of the two tristate buffers will be enabled for a particular input value. When a pulse is applied at the input of the delay unit, it will pass through only one of the two buffers. The enable input will determine which tristate buffer passes the pulse. This will change the amount of delay acquired by the signal according to the value of the enable input. To build the overall PUF circuit, the delay units are cascaded to construct a serial delay path. For a PUF circuit, we

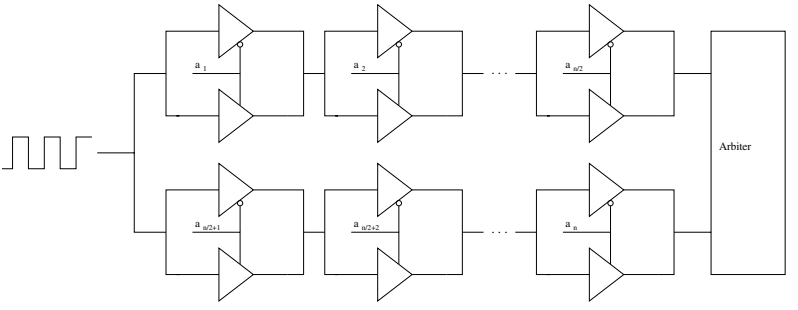

Fig. 1. PUF built with tristate buffers

need two separate delay paths. These delay paths are constructed as shown in Figure 1. The inputs of the two delay paths are generated by the same source while the outputs are fed to a flip-flop which we refer to as the *arbiter*. We assume the arbiter to be a positive edge-triggered flip-flop. The flip-flop has two inputs, the data and the clock. When the clock input has a rising edge the data input is captured at the output of the flip-flop. In the PUF setting, if the path that is connected to the data input has a smaller delay, then the output of the arbiter will be 1. Otherwise, the output will be 0.

In [9] the authors derive a linear delay model for the tristate PUF. The model is represented by an equation which gives the delay difference in terms of the challenge bits and the parameters of the PUF. Using their model we can relate the response bit r the challenge bits[1] a_i using the following function,

$$r = \mathrm{PUF}_Y(a) = \mathrm{sign}\left(\sum_{i=1}^{n}(-1)^{a_i}y_i + y_{n+1}\right).^2 \tag{1}$$

For simplicity we assume that n is even and we define $Y = [y_1, \ldots, y_{n+1}]$ where y_i denotes the imbalance in the signal propagation paths of the i^{th} stages and y_{n+1} denotes the imbalance in the delay between the upper and the lower paths and the setup time of the arbiter. These variables are dependent on the circuit itself and will vary from one PUF circuit to another. $\mathrm{Sign}(x) = 1$ if $x \geq 0$, and 0 if $x < 0$. When the argument in $\mathrm{sign}(x)$ is zero the output becomes random. It is not possible to predict the behavior of the circuit when the two signals have the exact same mismatch. In fact, this behavior will happen to an even larger window of delay mismatch values. In such cases the PUF is said to be *metastable*. We will shortly address this issue by introducing the noise parameter ϵ_P. It is important to note here that the delay variations y_i will depend on the fabrication process of the PUF circuit. Therefore, one would expect these variables to follow a normal distribution. In particular, the y_i values will follow a Gaussian distribution of

[1] In this paper we use superscripts with parenthesis to refer to sequences of strings or bits, i.e. $a^{(j)}$. We use the subscripts to denote bits of a string, i.e. a_i.

[2] This model is almost identical to the model derived for MUX based PUFs.

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 could be represented using a linear inequality means that given a sufficient number of challenge-response pairs $(a^{(i)}, r^{(i)})$ for a single PUF, an attacker might be able to model the system using standard linear programming techniques [39,2]. To get an idea of these modeling attacks, for a 128-bit PUF with about 4000 challenge-response pairs one can model the PUF with an accuracy of less than 5%. Such an observation seems to completely undermine the idea of using a PUF. Although a PUF might be tamper-resilient, it still can be easily modelable. For theoretical and experimental results on modeling a PUF the reader is referred to [14,18,9]. In this paper we view this ability as an advantage that can help create a more practical system. The ability to model a PUF will eliminate the need to store a huge database to track all the deployed devices. The server can store the simple model captured by the real vector Y for each device. However, note that even with the best ability to model a PUF there will always be a level of inaccuracy caused by metastability. This observation is due to multiple reasons. First, the thermal noise will cause slight fluctuations in the internal variables therefore causing a change in the output. Second, the two signals propagating inside a PUF will sometimes enter a race condition such that the decision made by the arbiter will be random. Race conditions are in general the more dominant reason for metastability, and they will particularly happen when the delay difference between the two internal paths is less than the resolution of the arbiter. As a result, any PUF device can only be accurately modeled within a certain level. The best modeling schemes tested so far can provide an inaccuracy as low as 3% [29]. In our notation we denote the amount of error that exists in our best model of a PUF circuit with ϵ_P.

Next, we introduce an enhancement to the delay-based PUF. We use an n-bit non-zero binary string x to implement a linear bijection on the input challenge before it is fed into the PUF. Let a be the challenge string sent to the PUF. To produce the actual challenge string a' we treat x and a as elements of a finite field $GF(2^n)$ and compute the product $a' = xa \in GF(2^n)$. We next define a new PUF equation which takes this enhancement as well as the error in modeling the PUF into consideration[3]

$$\mathrm{PUF}_{Y,x,\epsilon_p}(a) = \mathrm{PUF}_Y(xa) \oplus \nu, \tag{2}$$

where $\nu = 1$ with probability ϵ_P and $\nu = 0$ with probability $1 - \epsilon_P$. The field multiplication may be implemented with low footprint using a simple linear feedback shift register (LFSR) based serial modular polynomial multiplier circuit. The choice of generating polynomial makes no difference in terms of the properties of the PUF device. Hence, for efficiency, low-weight polynomials (e.g. trinomials) may be used in the implementation [3].

[3] While this enhancement does not prevent modeling attacks, it will indeed have an affect on the man-in-the-middle attacks as we will see in Section 5.

3 LPN-Based Authentication Protocols

In this section we give a quick review of the LPN problem and the different HB authentication schemes which base their security on the hardness of LPN. We focus our attention on HB and HB$^+$. For a certain $\epsilon \in \left(0, \frac{1}{2}\right)$ the LPN problem is denoted by LPN$_\epsilon$, and stated as: *Given k random binary n-bit strings $a^{(j)}$ and the bits $z^{(j)} = a^{(j)} \cdot s \oplus \nu$ for some $s \in \{0,1\}^n$, 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 s.*[4]

The LPN problem is known to be NP-hard [4]. In [19], the authors show that the LPN problem is log-uniform and even hard to approximate within a ratio of 2. Kearns proved in [24] that the LPN problem is even hard in the statistical query model. The best known algorithm to solve the LPN problem is the BKW algorithm [5]. However, there has been a number of improvements on the algorithm with the best running time of $2^{O(n/\log n)}$ [12,30,33].

In the HB protocol [19], the tag and the reader share an n-bit secret string s. To authenticate the tag, the reader starts sending randomly generated n-bit challenge strings $a^{(j)}$. The tag responds with the bit $z^{(j)} = a^{(j)} \cdot s \oplus \nu$ 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 s after collecting n challenge-response pairs using Gaussian elimination. In [19], 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$_\epsilon$ 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 [20]. In HB$^+$ the tag and the reader share two n-bit strings s_1 and s_2. The tag starts the authentication session by sending a random n-bit string $b^{(j)}$. The reader then responds with $a^{(j)}$ just like the HB protocol. Finally the tag responds with $z^{(j)} = a^{(j)} \cdot s_1 \oplus b^{(j)} \cdot s_2 \oplus \nu$, where ν is defined as above. The protocol is proved 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 [20] and [23] 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 [16]. 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^{++} [7], HB-MP [34] and HB* [8]. A recent proposal is the HB$^\#$ [15]. In

[4] We follow the LPN formulation given in [23].

their work the authors propose a modified version of HB$^+$ which uses Toeplitz matrices rather than vectors for a shared secret. Under a strong conjecture the proposal 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 from the tag to the reader. Not that the GRS-MIM-model will essentially protect against the previously mentioned man-in-the-middle attack. Our work here is based on the more recent protocol PUF-HB [1] which introduces PUFs to the HB paradigm.

4 New Authentication Family: HB+PUF

In this section we present the proposed protocol. We will use $\mathcal{R}$ to denote the reader and $\mathcal{T}$ to denote the tag. n_1 and n_2 will be our security parameters. $\mathcal{T}$ and $\mathcal{R}$ are both characterized by the set of variables $(k, s_1, s_2, x, Y, \epsilon_P, \epsilon, u)$ where $s_1, x \in \{0,1\}^{n_1}$, $s_2 \in \{0,1\}^{n_2}$ and $Y = [y_1, y_2, \ldots, y_{n_1+1}]$ such that $y_i \in N(0,1)$ where $N(\mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2. The noise parameters are $\epsilon, \epsilon_P \in \left(0, \frac{1}{2}\right)$. We use k to denote the number of rounds required for authentication. The last variable u is an integer in the range $[0, n]$ such that $\epsilon_f k \leq u$, where $\epsilon_f = \epsilon_P + \epsilon - 2\epsilon_p\epsilon$ denotes the total noise in the scheme.

With this notation we describe the basic authentication step. In every round, $\mathcal{T}$ randomly generates $b \in \{0,1\}^{n_2}$ and sends it to $\mathcal{R}$. Upon reception $\mathcal{R}$ replies with the challenge $a \in \{0,1\}^{n_1}$. Finally, $\mathcal{T}$ computes

$$z = a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,\epsilon_p}(a) \oplus \nu \ , \tag{3}$$

where $\nu = 1$ with probability ϵ and 0 with probability $1 - \epsilon$. Notice that this is very similar to the basic authentication step in HB$^+$. The only difference is that here we add a PUF operation. In order for $\mathcal{R}$ to authenticate $\mathcal{T}$, the same basic authentication step is repeated for k rounds. In every round $\mathcal{R}$ checks to see if $\mathcal{T}$'s response is equal to $(a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,0}(a))$. If the response is not equal to this term, $\mathcal{R}$ marks the response wrong. At the end of the k^{th} round, $\mathcal{R}$ authenticates $\mathcal{T}$ if and only if the number of wrong responses is less than 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 $\mathcal{R}_\sigma$ characterized by $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$. Following the notation in [23] we define $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle := 1$ iff $\mathcal{E}$ is authenticated by the reader, and is equal to 0 otherwise. The following protocol formalizes this interaction:

Protocol 1 (HB+PUF): $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle$

1. $\mathcal{R}_\sigma$ sets the counter $c = 0$
2. $\mathcal{E}$ sends $b \in \{0,1\}^{n_2}$ to $\mathcal{R}_\sigma$
3. $\mathcal{R}_\sigma$ choses $a \in \{0,1\}^{n_1}$ uniformly at random and sends it to $\mathcal{E}$
4. $\mathcal{E}$ sends z to $\mathcal{R}_\sigma$
5. if $z \neq a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y,x,0}(a)$ then $c = c + 1$
6. Steps 2 through 5 are repeated for k iterations
7. If $c \leq u$ then $\langle \mathcal{E}, \mathcal{R}_\sigma \rangle = 1$, otherwise it equals 0.

5 Security Analysis

In this section we show that the proposed protocol HB+PUF is at least as secure as the HB$^+$ protocol. We also discuss security against man-in-the-middle attacks. Finally, we consider the parameter selection to obtain a secure implementation. The reduction from HB+PUF to HB$^+$ is in fact very simple. As can be seen from Equation 3, the HB+PUF protocol utilizes all the terms of HB$^+$, and only adds a PUF operation. Therefore, it should be expected that the HB+PUF protocol can not be less secure than the HB$^+$ protocol. We now formalize this intuition by showing that any algorithm capable of successfully attacking the HB+PUF protocol can be used to successfully attack HB$^+$. The HB$^+$ protocol uses a tag $\mathcal{T}_\tau^+$ and a reader $\mathcal{R}_\tau^+$ both of which can be characterized by the string of variables $\tau = (k, s_1, s_2, \epsilon, u)$. The variables in τ are defined as we have done for the HB+PUF variables in Section 4. We also use $\langle \mathcal{E}, \mathcal{R}_\tau^+ \rangle$ to indicate an authentication session between any entity $\mathcal{E}$ and an HB$^+$ reader $\mathcal{R}_\tau^+$ using the HB$^+$ protocol. Similar to Protocol 1, $\langle \mathcal{E}, \mathcal{R}_\tau^+ \rangle = 1$ when the reader authenticates and 0 otherwise. This notation mostly follows the work presented in [23]. Recall from the previous section that in the HB+PUF protocol we use a tag $\mathcal{T}_\sigma$ and a reader $\mathcal{R}_\sigma$ both of which can be characterized by the string of variables $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$. We next state the reduction and keep the proof to Appendix 8. We prove the reduction in the active attacker model used to prove the security of the HB$^+$ protocol. In this model the attacker interacts with the tag in a learning session before he attempts to impersonate as the tag to an honest reader[5].

Theorem 1. *Let $\mathcal{A}$ be an algorithm which interacts with an honest HB+PUF tag $\mathcal{T}_\sigma$ for q authentication sessions to achieve $\Pr[\langle \mathcal{A}, \mathcal{R}_\sigma \rangle = 1] > \delta$, where $\mathcal{R}_\sigma$ is an honest HB+PUF reader. Then, there exists an algorithm $\mathcal{A}'$ which can interact with any HB$^+$ tag $\mathcal{T}_\tau^+$ for q authentication sessions to achieve $\Pr[\langle \mathcal{A}', \mathcal{R}_\tau^+ \rangle = 1] > \delta$, where $\mathcal{R}_\tau^+$ is an honest HB$^+$ reader.*

We point out that in the PUF-HB scheme the security reduction is much more involved since the secret s_1 is replaced by the PUF operation. Theorem 1 poses an immediate question of how the HB+PUF protocol behaves in relation to the the known man-in-the-middle attack against HB$^+$ [16]. Briefly, in this attack an adversary replaces all the challenges $\{a^{(j)}\}_{j=1}^k$ sent from the reader in a single authentication session by $\{a^{(j)} \oplus w\}_{j=1}^k$ where $w \in \{0,1\}^{n_1}$. The attacker knows that the challenges will interact with the secret s_1 through $a^{(j)} \cdot s_1$. At the end of the k rounds, if the reader authenticates the tag, then the adversary can deduce with very high probability that his changes did not affect the responses of the tag, and therefore $w \cdot s_1 = 0$. On the other hand, if the reader rejects the tag, then the adversary will know with a very high probability that $w \cdot s_1 = 1$. Repeating the same attack n_1 times will allow the adversary to collect n_1 linear equations

[5] We only need HB$^+$ to prove the reduction to the LPN problem. However, HB$^+$ is reduced to the LPN problem under the same attacker model used here.

containing s_1. Therefore, the adversary can use Gaussian elimination to solve for s_1 with a high probability.

As we pointed out earlier, one of the main reasons for such an attack to work is the linearity of the inner product operation. In our scheme the challenges $a^{(j)}$ are not only subjected to the inner product operation $a^{(j)} \cdot s_1$, but also to a PUF operation $a^{(j)} \cdot s_1 \oplus \mathrm{PUF}_{Y,x,\epsilon_p}(a^{(j)})$. With both operations being used, an adversary will need to find a way to modify the challenges such that he can deduce information about each of the two operations separately. To see why a PUF operation will help against the man-in-the-middle attacks, notice that on one hand the PUF is inherently non-linear due to the sign operation. Therefore, it will prevent against any simple man-in-the-middle attack trying to explore linearity, such as the attack in [16]. On the other hand, it has been shown in [1] that the probability distribution of two different challenges $a^{(1)}$ and $a^{(2)}$ yielding the same output from a PUF operation, will only depend on the Hamming distance between $a^{(1)}$ and $a^{(2)}$. This means that any successful man-in-the-middle attack would have to exploit the Hamming distances between different challenges. However, recall from the end of Section 4 that the PUF circuit used in HB+PUF implements a field multiplication over $GF\,(2^{n_1})$ with the secret string x. This multiplication will partially obfuscate the Hamming distance between different challenges. Therefore, the attacker's ability to deduce correlations between the inputs and the outputs of the PUF will be partially hindered.

Note that here we are talking with respect to the GRS-MIM model introduced in [15]. To protect against the most general class of man-in-the-middle attacks, we suggest adding a second PUF circuit to operate on the $b^{(j)}$ strings sent by the tag. In such a scheme the response of the tag would be

$$z = a \cdot s_1 \oplus b \cdot s_2 \oplus \mathrm{PUF}_{Y_1,x_1,\epsilon_{p1}}(a) \oplus \mathrm{PUF}_{Y_2,x_2,\epsilon_{p2}}(b) \oplus \nu \ . \tag{4}$$

The suggested scheme will be more demanding in terms of hardware and power. However, we predict that it will be resilient against man-in-the-middle attacks.

We finish this section by discussing security parameters for an implementation of the design. As shown by Theorem 1 our protocol is at least as secure as the HB$^+$ protocol, which in turn is at least as hard as solving the LPN problem. All with respect to the active attacker model. In [30] the authors give a careful study of the BKW algorithm for solving the LPN problem. They conclude that the parameters first introduced for the HB$^+$ protocol by [20] and then by [23] do not provide sufficient security. In our implementation we follow the new parameters suggested by [30] and later adopted by [15]. To achieve 80-bits of security we choose $n_1 = 80, n_2 = 512$, $\epsilon_f = 0.15$ and $k = 200$. Note that ϵ_f is not a separate parameter but rather a result from ϵ_p and ϵ. In our implementation we will have $\epsilon_p = 0.15$ and $\epsilon = 0$.

6 PUF-Based RNG

In Section 2 we discussed the inherent metastability in a PUF circuit. As we pointed out earlier, these metastable states result from either environmental

fluctuations, or race conditions which occur between the two propagating signals inside a PUF. In this section, we outline how metastability could be used to generate random bits. We note here that using a PUF circuit as an RNG is not a new idea. It has been previously proposed in [35]. In their design the authors use an LFSR to generate a stream of challenges. Each challenge is fed to the PUF multiple times in order to decide whether the challenge is metastable or not. Finally, the metastable outputs are used to extract randomness. In our approach, we take advantage of a PUF feedback setting. This approach will essentially remove any need to separately check each challenge for metastability. Therefore, decreasing the control logic, and increasing the throughput.

Our RNG design is based on a shift register feeding a PUF circuit in parallel. As we have concluded in Section 5 the size of the PUF and thus the size of the shift register will be 80 bits. The register is initialized to a random bit string. At every clock cycle the output of the PUF is fed back to the most significant bit of the shift register, while the least significant bit is discarded. This mode of operation will ensure a continuous stream of bits. Without metastability no randomness is expected to come out of this construction. Therefore, to assess the generated randomness we need to get a good estimate on the ratio of metastable points.

In order to get an estimate for the metastability ratio, we implemented the PUF circuit on a Xilinx XC2VP30 FPGA. In typical PUF implementations, extra precautions are taken to prevent metastability. However, we are interested in having a high level of metastability. This is the case, since we use the PUF in a *noisy* authentication scheme, and as an RNG. To help induce a higher level of metastability we allow close adjacency between the PUF circuit and other parts of the implementation. We carried out a restart test by collecting 1000 different bit streams. Each bit stream was collected after the system was reset and initialized to the same state. In a completely stable system, these bit streams would have to be identical. However, in a metastable system, every time a metastable point occurs these streams are expected to break into two groups, with each group following a different choice of the metastable point. After tracking all the bit streams we found that after 6400 bits all the 1000 streams were in completely different states, therefore suggesting the occurrence of 1000 metastable points. This yields an overall metastability ratio of about 15%. With this ratio, we can insure that the output always contains a metastable point by Xor-ing every 8 consecutive bits and using the result as the output of the RNG.

To verify the statistical quality of the RNG output, we collected a large number of bit streams and analyzed them with the NIST statistical test suite. As recommended by the NIST tools, every bit stream contained $20,000$ points. The test suite reports a *proportion* value, which reflects the ratio of bit streams which actually passed this particular test. The final results we obtained are shown in Table 1. The NIST tools return a statistical result where even a true random number generator could fail in some of the runs. We can conclude from the shown results that the proposed RNG is a reasonably good generator.

Table 1. NIST suite results

Test Name	Proportion
Frequency	100%
Frequency within block	100%
Longest run of ones in block	95%
Cumulative sum	100%
Runs	100%
Discrete Fourier Transform	100%
Non-overlapping template matching	95%
Overlapping template matching	97.5%
Maurer's Universal	100%
Approximate Entropy	97.5%
Serial	97.5%
Lempel-Ziv Complexity	100%

7 Implementation

The authentication scheme presented in Section 4 is implemented as shown in Figure 2. The PUF circuit is positioned at the center of the implementation to ensure tamper-resilience for the entire circuit. As we have verified from our FPGA implementations of a PUF circuit, any change in the surrounding hardware to the PUF circuit will result in changing the PUF's internal variables. We point out here that a PUF can easily protect against active side-channel attacks. However, for passive side-channel attacks a designer might have to resort to standard power balancing techniques [42,43,38]. Although effective, these technique will incur about $200 - 400\%$ area overhead. A cost too high for lighweight implementations. Our authentication architecture runs in two different modes of operation during the entire protocol.

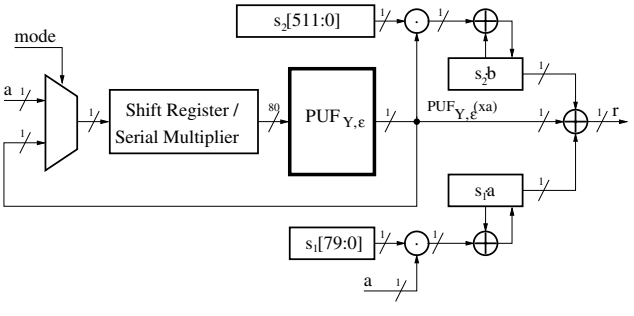

Fig. 2. PUF Authentication Scheme

RNG Mode: In this mode the PUF circuit acts as a random number generator. As explained in Section 6 the random string $b \in \{0,1\}^{512}$ is achieved using a shift register along with the PUF circuit. This shift register is initialized with the initialization value (IV) stored in an 80-bit ROM structure. The shift register will

be serially initialized to (IV) in RNG mode. For the remainder of the RNG operation the serial input of the shift register will be fed back by the output of the PUF. Conveniently enough, we do not need to store the entire random string b generated by the PUF. As b is generated 1 bit at a time, we can serially compute the inner product $b \cdot s_2$, and at the same time serially transmit b to the reader. It is important to point out that in the RNG mode, the system will not be able to detect any active side-channel attacks. With a stream of random bits, the attacker's effect is gone undetectable. This will not be a major problem since any invasive attack on the circuit will permanently affect the PUF circuit. Therefore, as soon as the circuit is back to PUF mode, the attack can be detected. In the case where more gates are dedicated for security purposes, two separate PUF circuits can be used for authentication and random number generation.

PUF Mode: In this mode we perform the serial field multiplication xa which will be the input of the PUF. The hardware component *Shift Register/Serial Multiplier* shown in Figure 2 is used for this multiplication. The serial input of this shift register comes from the input a which is serially received. The field multiplication is realized through an LFSR serial multiplier, and is carried out in parallel with the inner product operation $a \cdot s_1$. These two operations will operate serially and will take about 80 cycles. The result of the field multiplication xa is fed to the PUF as the challenge input. The response bit of the PUF is then XOR-ed with the inner products $a \cdot s_1$ and $b \cdot s_2$. Finally, the response of the entire circuit r is transmitted to the reader. Note from the last section that the ratio of metastability was about 15%. This matches the overall desired noise. Therefore, there will be no need for an added noise parameter ϵ.

To estimate the gate count, HB+PUF was developed into Verilog modules and synthesized using the Synopsys Design Compiler. For the synthesis we used the TSMC 0.13 μm ASIC library. The circuit components and their gate count are explained as follows:

-**ROM Structure:** The IV for the RNG and the private keys s_1 and s_2 are stored inside the hardware and they are unique for each tag. Instead of utilizing flip-flops to store these values, we designed a ROM structure for low-area storage. To minimize the area usage, we used a design similar to a look-up table. Separate architectures are needed to store s_1, s_2 and IV. Since s_2 is 512 bits, s_1 and IV are 80 bits, we have 672 bits of total ROM area. Synthesis results show that 110 gates are required for this storage.
-**PUF Circuit:** In our authentication scheme, we utilize an 80-bit PUF circuit. As pointed out in Section 2 we use the tristate PUF implementation presented in [9]. This particular PUF implementation is of interest due to its low-power and low-area features. When we used the tristate PUF design our synthesis results showed the area of the PUF to be 350 gates. However, with custom circuit design, where each tristate buffer utilizes about a single gate, this number was reduced to 160 gates.
-**Shift register/Serial Multiplier:** The shift register has a total size of 80 bits. The structure also contains a number of XOR gates used to achieve the

field multiplication. In addition, 2-to-1 multiplexers are used to decide which inputs feed the individual flip-flops of the register. Synthesis results show that a total equivalent of 500 gates is needed for this structure.

-**Serial inner products:** The AND and XOR components shown in Figure 2 are utilized for serial inner product operations. The boxes labeled as $s_2 \cdot b$ and $s_1 \cdot a$ are single flip-flops storing the results of the inner products $s_2 \cdot b$ and $s_1 \cdot a$ of Protocol 1. They work as accumulators. In each clock cycle, one bit of s_2 and one bit of b pass through an AND gate and the result is XORed with the value in the accumulator flip-flop. The same procedure is repeated for s_1 and a. In the end, the results in the accumulator registers $s_2 \cdot b$ and $s_1 \cdot a$ are XOR-ed with the result of the $\mathrm{PUF}_{Y,\epsilon}(xa)$ and the result is sent to the output as r. The area for these operations is estimated at 50 gates.

-**Control logic:** The control logic for this scheme is quite simple. For the ROM structures storing s_1 and s_2, a single 9-bit counter is needed. Since the inner products for s_1 and s_2 are operated in parallel, a single counter is enough for both operations. For the RNG a 3-bit counter is needed to track the XOR-ing of each 8 consecutive bits. This can be interleaved with the inner product operation $s_2 \cdot b$. The architecture has only 2 modes of operation. Therefore, a single flip-flop would suffice to track the state of the hardware. The total area of the control block is estimated at about 150 gates.

The total area of the authentication hardware is 970 gates. This is below 1K gates, a typical threshold for the RFID's footprint allotted for security [37].

8 Conclusion

In this paper we presented a tamper-resilient and provably secure authentication scheme requiring less than 1K gates. We proved the security of our scheme against active attacks, and against known man-in-the-middle attacks. Moreover, the proposed scheme seems resilient against a more general class of man-in-the-middle attacks. We also demonstrated an efficient method for generating the random bits needed for our proposed protocol. This was done without incurring significant overhead to the hardware, and by reutilizing parts of the authentication circuit.

Our work here opens an interesting avenue for exploring cryptographic algorithms naturally supported by the hardware. For future work one might hope to explore modifications to the current protocol which might yield provable security against man-in-the-middle attacks. This problem has been addressed by multiple proposals, but no proof has been provided.

References

1. Hammouri, G., Sunar, B.: PUF-HB: A Tamper-Resilient HB based Authentication Protocol. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 346–365. Springer, Heidelberg (2008)
2. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. Mathematical Programming 71(2), 221–245 (1995)

3. Berlekamp, E.R.: Algebraic coding theory. McGraw-Hill, New York (1968)
4. 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)
5. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: Proceedings of STOC 2000, pp. 435–440. ACM, New York (2000)
6. Bogdanov, A., Leander, G., Knudsen, L.R., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT - An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
7. Bringer, J., Chabanne, H., Dottax, E.: HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In: Proceedings of SECPERU 2006, Washington, DC, USA, pp. 28–33. IEEE Computer Society, Los Alamitos (2006)
8. 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, pp. 23–26 (2007)
9. Ozturk, E., Hammouri, G., Sunar, B.: Physical Unclonable Function with Tristate Buffers. In: Proceedings of ISCAS 2008 (2008)
10. Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., Uhsadel, L.: A Survey of Lightweight Cryptography Implementations. IEEE Design & Test of Computers – Special Issue on Secure ICs for Secure Embedded Computing 24(6), 522–533 (2007)
11. Feldhofer, M., Dominikus, S., Wolkerstorfer, J.: Strong Authentication for RFID Systems Using the AES Algorithm. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156. Springer, Heidelberg (2004)
12. 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
13. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of CCS 2002, pp. 148–160. ACM, New York (2002)
14. 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)
15. Gilbert, H., Robshaw, M., Seurin, Y.: $HB^{\#}$: Increasing the Security and Efficiency of HB^{+}. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 361–378. Springer, Heidelberg (2008)
16. Gilbert, H., Robshaw, M., Sibert, H.: An Active Attack Against HB+ A Provably Secure Lightweight Authentication Protocol. IEE Electronic Letters 41, 1169–1170 (2005)
17. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., et al.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
18. Ozturk, E., Hammouri, G., Sunar, B.: Towards Robust Low Cost Authentication for Pervasive Devices. In: PERCOM 2008, Hong Kong, March 17-21 (2008)
19. Hopper, N.J., Blum, M.: Secure Human Identification Protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)
20. 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)

21. Kaps, J., Gaubatz, G., Sunar, B.: Cryptography on a Speck of Dust. Computer 40(2), 38–44 (2007)
22. Kaps, J.-P., Sunar, B.: Energy Comparison of AES and SHA-1 for Ubiquitous Computing. In: Zhou, X., Sokolsky, O., Yan, L., Jung, E.-S., Shao, Z., Mu, Y., Lee, D.C., Kim, D.Y., Jeong, Y.-S., Xu, C.-Z. (eds.) EUC Workshops 2006. LNCS, vol. 4097, pp. 372–381. Springer, Heidelberg (2006)
23. 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)
24. Kearns, M.: Efficient Noise-Tolerant Learning from Statistical Queries. In: Proceedings of STOC 1993, pp. 392–401. ACM Press, New York (1993)
25. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
26. 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)
27. 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)
28. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, p. 196. Springer, Heidelberg (2007)
29. Lee, J.W., Daihyun, L., Gassend, B., Samd, 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)
30. Levieil, E., Fouque, P.: An Improved LPN Algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, p. 348. Springer, Heidelberg (2006)
31. Lim, C., Korkishko, T.: mCrypton-A Lightweight Block Cipher for Security of Low-cost RFID Tags and Sensors. In: WISA, vol. 5, pp. 243–258
32. 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)
33. Lyubashevsky, V.: The parity problem in the presence of noise, decoding random linear codes, and the subsetsum problem. In: APPROXRANDOM (2005)
34. Munilla, J., Peinado, A.: HB-MP: A further step in the HB-family of lightweight authentication protocols. Comput. Networks 51(9), 2262–2267 (2007)
35. O'Donnell, C.W., Suh, G.E., Devadas, S.: Puf-based random number generation. Number 481 (November 2004)
36. Posch, R.: Protecting Devices by Active Coating. Journal of Universal Computer Science 4(7), 652–668 (1998)
37. Poschmann, A., Leander, G., Schramm, K., Paar, C.: New Ligh-Weight Crypto Algorithms for RFID. In: Proceedings of ISCAS 2007, pp. 1843–1846 (2007)
38. Regazzoni, F., Badel, S., Eisenbarth, T., Grobschadl, J., Poschmann, A., Toprak, Z., Macchetti, M., Pozzi, L., Paar, C., Leblebici, Y., et al.: 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)
39. Roos, C., Terlaky, T., Vial, J.-P.: Interior Point Methods for Linear Optimization, 2nd edn. Springer, Heidelberg (2005)

40. Skoric, B., Maubach, S., Kevenaar, T., Tuyls, P.: Information-theoretic Analysis of Coating PUFs. Cryptology ePrint Archive, Report 2006/101 (2006)
41. Standaert, F., Piret, G., Gershenfeld, N., Quisquater, J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: Workshop on RFID and Lightweight Crypto, Graz, Austria (2005)
42. 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: Proceedings of ESSCIRC 2002, pp. 403–406 (2002)
43. Toprak, Z., Leblebici, Y.: Low-power current mode logic for improved DPA-resistance in embedded systems. In: ISCAS 2005, pp. 1059–1062 (2005)
44. Tuyls, P., Skoric, B.: Secret Key Generation from Classical Physics: Physical Uncloneable Functions. Philips Research Book Series. Springer, Heidelberg (2006)

Appendix A

Proof of Theorem 1

Proof. The basic operation of $\mathcal{A}'$ is to map a given $\tau = (k^+, s_1^+, s_2^+, \epsilon^+, u^+)$ characterizing the HB$^+$ tag and reader to $\sigma = (k, s_1, s_2, x, Y, \epsilon_p, \epsilon, u)$ used to characterize an HB+PUF tag and reader. Note that all the variables in the HB$^+$ protocol are still used in the same manner in the HB+PUF protocol. Therefore we can create $\sigma^+ = \big(k = k^+, s_1 = s_1^+, s_2 = s_2^+, x, Y, \epsilon = \epsilon^+, \epsilon_p = 0, u = u^+\big)$. The variable x is chosen randomly to be any string in $\{0,1\}^{n_1}$. The (n_1+1) real vector Y is chosen such that $y_i \in N(0,1)$. $\mathcal{A}'$ runs as follows: It initializes $\mathcal{A}$ and allows it to carry its communication with $\mathcal{T}_\tau^+$. In particular, $\mathcal{A}'$ passe the vector b sent by $\mathcal{T}_\tau^+$ to $\mathcal{A}$ which will reply with the vector a. Again $\mathcal{A}$ passes a back to $\mathcal{T}_\tau^+$. Finally, when $\mathcal{T}_\tau^+$ returns its response z, $\mathcal{A}'$ returns $\hat{z} = z \oplus \mathrm{PUF}_{Y,x,0}(a)$ to $\mathcal{A}$. The same step is followed for all q authentication rounds between $\mathcal{T}_\tau^+$ and $\mathcal{A}$. When $\mathcal{A}'$ wants to authenticate itself to $\mathcal{R}_\tau^+$, it again runs $\mathcal{A}$ in its authentication phase. $\mathcal{A}$ will start by sending the radnom string $b^{(i)}$. $\mathcal{A}'$ will pass the string directly to $\mathcal{R}_\tau^+$ which will respond with the vector $a^{(i)}$. $\mathcal{A}'$ passes $a^{(i)}$ back to $\mathcal{A}$. Finally, when $\mathcal{A}$ returns its response $z^{(i)}$, $\mathcal{A}'$ returns $\hat{z}^{(i)} = z^{(i)} \oplus \mathrm{PUF}_{Y,x,0}(a^{(i)})$ to $\mathcal{R}_\tau^+$. The algorithm $\mathcal{A}'$ repeats these steps for all k rounds of the authentication session, such that $i = 1 \ldots k$.

To see why this will actually work, notice that in the first q rounds $\mathcal{A}$ is getting the response $\hat{z}$ which is effectively responses from a tag $\mathcal{T}_{\sigma^+}$. This means that at the end of the q authentication sessions $\mathcal{A}$ will have effectively communicated with $\mathcal{T}_{\sigma^+}$. In the authentication phase, when $\mathcal{A}'$ tries to authenticate itself to $\mathcal{R}_\tau^+$, it uses the algorithm $\mathcal{A}$ which will be trying to authenticate itself to $\mathcal{R}_{\sigma^+}$. Assuming that $\mathcal{A}$ will succeed in impersonating $\mathcal{T}_{\sigma^+}$ with probability larger than δ, then the responses returned by $\mathcal{A}$ which are $z^{(i)}$ will match the responses of an honest tag with probability larger than δ. However, this immediately implies that the responses returned by $\mathcal{A}'$ to $\mathcal{R}_\tau^+$ which are $\hat{z}^{(i)}$ and which differ from $z^{(i)}$ with the term $\mathrm{PUF}_{Y,x,0}(a^{(i)})$ will match the responses of an honest tag with probability larger than δ. Therefore, $\mathcal{A}'$ should also succeed in impersonating a tag $\mathcal{T}_\tau^+$ with probability larger than δ.