Lattice Basis Reduction Attack against Physically Unclonable Functions

Fatemeh Ganji[1], Juliane Krämer[2], Jean-Pierre Seifert[1], Shahin Tajik[1]

[1]Security in Telecommunications
Department of Software Engineering and Theoretical Computer Science
Technische Universität Berlin and Telekom Innovation Laboratories
Berlin, Germany
{fganji, jpseifert, stajik}@sec.t-labs.tu-berlin.de

[2]Cryptography and Computer Algebra
Department of Computer Science
Technische Universität Darmstadt
Darmstadt, Germany
jkraemer@cdc.informatik.tu-darmstadt.de

ABSTRACT

Due to successful modeling attacks against arbiter PUFs (Physically Unclonable Functions), the trend towards consideration of XOR arbiter PUFs has emerged. Nevertheless, it has already been demonstrated that even this new non-linear structure, with a restricted number of parallel arbiter chains, is still vulnerable to more advanced modeling attacks and side channel analyses. However, so far the security of XOR arbiter PUFs with a large number of parallel arbiter chains has not been appropriately assessed. Furthermore, as another countermeasure against modeling and physical attacks, the concept of controlled PUFs, i.e., with a limited access to challenges and responses, has also been developed. Towards a better understanding of the security of XOR arbiter PUFs, the present paper simultaneously addresses all above mentioned countermeasures by introducing a novel attack, which is a combination of a lattice basis reduction attack and a photonic side channel analysis. We present how our new attack can be successfully launched against XOR arbiter PUFs with an arbitrarily large number of parallel arbiter chains. Most interestingly, our attack does not require any access to challenges or responses. Finally, by conducting an exhaustive discussion on our experimental results, the practical feasibility of our attack scenario is proved as well.

Categories and Subject Descriptors

B.7.m [**Integrated Circuits**]: Miscellaneous; C.3 [**Special-Purpose and Application-Based Systems**]: Smartcards; E.3 [**Data Encryption**]: Code breaking

CCS'15, October 12–16, 2015, Denver, Colorado, USA.

DOI: http://dx.doi.org/10.1145/2810103.2813723.

General Terms

Theory; Hardware security

Keywords

Physically Unclonable Functions; XOR arbiter PUFs; Lattice basis reduction; Hidden subset sum problem; Photonic emission analysis

1. INTRODUCTION

Physically Unclonable Functions (PUFs) have been introduced, among many other cryptographic applications, to protect integrated circuit (IC) designs against counterfeiting [12, 25]. As the previously suggested protection techniques, such as storage of a secret key in non-volatile memory (NVM), have been rather ineffective against invasive attacks [14, 16], the importance of the concept of PUFs has been stressed by several IC manufacturers. The inherent security associated with the physical characteristics of ICs make PUFs interesting for IC manufacturers, and attackers as well. These essential characteristics can be tailored to the specific needs of hardware authentication and identification [19].

As a prime example, the principle underlying the design of arbiter PUFs is the intrinsic delays of the symmetrically designed delay lines, consisting of logic gates in an IC [17]. Due to the subtle differences between these delays, a respective arbiter PUF generates a (virtually unique) *response* to a so called given *challenge*. Although this concept is well suited for security requirements in theory, in practice several machine learning (ML) attacks against arbiter PUFs have been successfully launched [17]. Here, the adversary collects a small subset of challenge-response pairs (CRPs) to build a mathematical model of the challenge-response behavior of the arbiter PUF. By launching an ML attack, a mathematical model may be delivered, which can predict the responses of the PUF with some accuracy for an arbitrarily chosen challenge. However, the delivery of the mathematical clone is not always guaranteed. Just recently, a novel probably approximately correct (PAC) learning framework has been

developed, which ensures the ultimate delivery of a mathematical clone for pre-defined levels of confidence and accuracy [10]. Therefore, arbiter PUF manufacturers have been forced to propose countermeasures including XOR arbiter PUFs as an effective technical action [34].

When applying a challenge to an XOR arbiter PUF composed of several arbiter chains, the joined response is generated by XORing all responses of the respective arbiter chains. It has been demonstrated that XOR arbiter PUFs are also subject to advanced and significantly more complex ML attacks [27]. The number of arbiter chains can be a limiting factor for this type of attacks, hence, the attack is effective only against XOR arbiter PUFs with a small number of arbiter chains. Moreover, it has also been shown that when the number of arbiter chains exceeds this limited number, additional side channel analyses can be combined with an ML attack to compromise the security of an XOR arbiter PUF [30]. Otherwise the number of CRPs, and consequently the time, required to launch an ML attack increases drastically [37]. It remains an open question whether the challenge-response behavior of an XOR arbiter PUF with a large number of arbiter chains (e.g., more than 16) can be learned by launching ML attacks. On the other hand, although being costly, semi-invasive attacks can be a promising solution leading to a complete and linear characterization of XOR arbiter PUFs. It has been demonstrated that arbiter as well as XOR arbiter PUFs can be physically characterized by performing a high resolution temporal photonic emission analysis [13, 35]. Due to the fact that this attack does not require the response of the PUF, the number of required measurements is only linear in the number of stages.

It is also clear that when the attacker cannot directly control the challenge-response behavior of the PUF, the aforementioned attacks cannot be mounted. For instance, when an attacker relying on the direct control of the challenges cannot apply the desired challenge, an ML or a physical attack would obviously fail. This promotes the introduction of a special type of PUF protocols (e.g., Controlled PUF [11], Slender PUF [21], and Recyclable PUFs [15]), where a part of, or all, challenge bits are generated by using either hash functions or True Random Number Generators (TRNGs). The main building blocks of these protocols are legacy arbiter PUFs and XOR arbiter PUFs [9], although the key message given in the above discussion is that these PUFs are not as secure as being claimed. This highly motivates not only the manufacturers to invest in this kind of PUFs, but also the attackers for further investigation.

Our present work is to the best of our knowledge the first study, which proves that an increase in the number of arbiter chains is not a "silver bullet", but rather a very limited step towards improving the robustness of XOR arbiter PUFs against attacks, if not being a step to reduce the security. Furthermore — and more importantly — in our attack scenario, an access to neither the challenges nor the responses is required. In brief, our main contributions are as follows:

- We present a novel mathematical proof of the vulnerability of legacy and controlled XOR arbiter PUFs to lattice basis reduction attacks. Specifically, we present how the lattice basis reduction attack proposed by Nguyen et al. [23] can be extended to compromise the security of XOR arbiter PUFs.
- We describe how a photonic side channel analysis can be combined with the lattice basis reduction attack to disclose the *hidden* challenges applied to a controlled XOR arbiter PUF with an arbitrarily large number of arbiter chains.
- Extensive experiments are conducted to validate the performance of our lattice basis reduction framework, qualitatively and quantitatively.

The present paper is organized as follows. The next chapter introduces PUFs, arbiter PUFs, and XOR arbiter PUFs. It continues with a brief introduction into the basics of lattices and the original hidden subset sum problem. Afterwards, in Section 3 we elaborate on the photonic side channel data acquisition providing input data for our lattice basis reduction algorithm. Section 4 is our main section, which explains how the notion of the extended (multi-dimensional) hidden subset sum problem can be applied to completely break the security of the XOR arbiter PUF family without knowing the challenges or their respective responses. Therefore, we explain how the original hidden subset sum must be extended and how we can overcome its main obstacles in the case of XOR arbiter PUFs. Section 5 presents subtle implementation details of our lattice basis reduction attack and its excellent results regarding the characterization of XOR arbiter PUFs with a large number of parallel chains. Section 6 concludes our paper by summarizing and outlining future work, where the newly developed framework can be applied to asses the security of XOR arbiter PUFs.

2. DEFINITIONS AND PRELIMINARIES

This section provides the background, important notations, and context information, which are useful to understand the general concept of arbiter and XOR arbiter PUFs. Furthermore, we provide a brief introduction to lattices and hidden subset sum (knapsack) problems. We also derive a representation of an XOR arbiter PUF under the notion of this subset sum (knapsack) problem.

2.1 PUF

PUFs, as a general concept, rely on intrinsic properties of a physical system, which embodies them. In general, PUFs can be thought of as mappings, which generate a *response* for a given *challenge*. Here we briefly describe the basic characteristics of an *ideal* PUF, and for a formal foundation and the formalization of the security of PUFs we refer the reader to [2].

Definition 1. *Let $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}$ be the set of challenges and the set of responses, respectively. A PUF can be represented by the function $f_{PUF} : \mathcal{X} \to \mathcal{Y}$ where $f_{PUF}(x) = y$, cf. [19]. Note that f_{PUF} is not a one-to-one mathematical function. Ideally, f_{PUF} aims to provide the following security-related properties.*

1. Evaluable*: f_{PUF} can be evaluated in polynomial time.*
2. Unique*: for a given PUF instance, the mapping f_{PUF} is instance-specific.*
3. Reproducible*: applying same challenges to f_{PUF} results in "close" responses with respect to a chosen distance metric.*
4. Unclonable*: for a given PUF f_{PUF} it is (almost) impossible to construct another mapping (i.e., physical entity) g_{PUF} so that "$g_{PUF} \approx f_{PUF}$".*
5. Unpredictable*: for a given set $U = \{(x_i, y_i) \mid y_i = f_{PUF}(x_i)\}$, it is (almost) impossible to predict a re-*

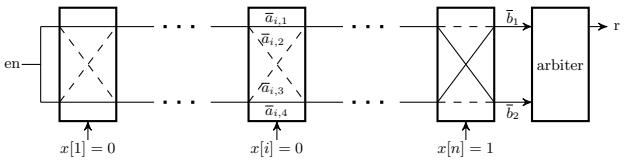

Figure 1: Schematic of an arbiter PUF composed of n stages and a final arbiter. Four different delays in each stage are shown. The signal propagates through either the direct paths or the crossed paths based upon the applied challenge $\mathbf{x}$. The binary response is generated by the final arbiter and depends on the arrival times of the signals.

sponse $y_r = f_{PUF}(x_r)$, where x_r is a random challenge and $(x_r, y_r) \notin U$.

6. One-way*: for a given random PUF instance $y = f_{PUF}(x)$, where x is drawn from a uniform distribution on $\{0,1\}^n$, we have*

$$\Pr[\mathcal{A}(f_{PUF}(x)) = x] < 1/p(n),$$

where $p(\cdot)$ is any positive polynomial. This means that the probability that any probabilistic polynomial time algorithm or physical procedure $\mathcal{A}$ can output x is negligible, cf. [26]. In other words, it is hard *to find x, if the respective response of a random instance of the PUF family is known, and the adversary can evaluate the PUF a polynomial number of times [18].*

The existing types of PUFs can only partially fulfill the above mentioned requirements. For instance, it has been demonstrated that one-wayness (as formulated in Definition 1) and unclonability cannot be fulfilled for arbiter PUFs. Nevertheless, the discussion on the formalization of PUF properties is beyond the scope of this paper and the reader is referred to [2,28] for more details.

One of the most promising candidates for PUF instances, which have been already widely accepted and utilized, are delay-based PUFs. For these PUFs, the silicon properties of a chip can be used effectively to meet the above mentioned requirements of PUFs.

2.2 Arbiter PUFs

For this special and widely accepted kind of PUFs, the non-symmetrical timings of symmetrically designed electrical paths on a chip play a major role for the implementation of the PUF. As illustrated in Figure 1, n switches (so called *stages*) form a chain, which is terminated by an arbiter. A challenge is an n-bit row vector $\mathbf{x} = (x[1], \cdots, x[n])$, where the i^{th} bit is fed into the i^{th} stage (e.g., multiplexer). $x[i]$ determines through which path in the i^{th} stage the signal propagates: if $x[i] = 1$, the crossed paths are chosen, otherwise the direct paths. When enabling the input signal at the first stage, two electrical signals propagate on two symmetrically designed paths to reach the end of the chain. The arrival times of these two signals differ due to the imperfections on the silicon. According to this difference, individual binary outputs of the arbiter are generated.

In order to represent the delays within the i^{th} stage, we define a random variable A_i that follows a Gaussian distribution with the mean μ_i and the deviation ω_i, cf. [10,29]. The four realizations of this random variable are $\overline{a}_{i,j}$ $(1 \le j \le 4)$, where $\overline{a}_{i,1}$ and $\overline{a}_{i,4}$ correspond to the delays of the upper and lower direct paths, whereas $\overline{a}_{i,2}$ and $\overline{a}_{i,3}$ are the delays of the upper and lower crossed paths, respectively, see Figure 1. Similarly, $\overline{b}_j$ is the realization of a Gaussian random variable for the total delay between the enable point and the outputs of the last stage of the PUF ($j = 1$ for upper path and $j = 2$ for lower path). In order to generate the binary output, the arbiter compares the difference of the signal arrival times with its limited precision γ. If the difference $\overline{b}_1 - \overline{b}_2$ is greater than, or equal to, the arbiter precision γ ($\gamma > 0$), it generates a "1", whereas if $\overline{b}_2 - \overline{b}_1 \ge \gamma$, the response is a "0". The case of $|\overline{b}_2 - \overline{b}_1| < \gamma$, also called the meta stable case, must have been already resolved by the manufacturer, when verifying the respective PUF properties. However, since responses of a PUF to given challenges are not considered in our attack scenario, the functionality and the efficiency of our framework are not affected in this case.

It has been shown that the accumulated real-valued delays $\overline{b}_j$ lie with probability 99.7 % within the interval $[n\mu - 3\omega\sqrt{n}, n\mu + 3\omega\sqrt{n}]$, cf. [10]. Therefore, regarding the restricted length of this interval as well as the limited precision of the arbiter (γ), cf. [20], all above mentioned real-valued delays can be mapped to certain integer values lying within the interval $[0, \lceil (n\mu + 3\omega\sqrt{n})/\gamma \rceil]$. We can therefore treat all delays $\overline{b}_j$ and also $\overline{a}_{i,j}$ as integers, bounded by some value M where

$$M = O(n).$$

The integer values corresponding to the previously defined real-valued delays are presented by $a_{i,j}$ and b_j, respectively.

To account for the different path configurations for a given challenge $\mathbf{x}$, we define the following one-bit value, cf [10]:

$$u_i = \bigoplus_{k=1}^{i} x[k].$$

Moreover, the vector of a challenge $\mathbf{x}$ is mapped to a vector $\mathbf{x}_1$. This reflects the impact of the path configurations more systematically, cf. [35]. Thus, we define the mapping $f_1 : \{0,1\}^n \to \{0,1\}^{4n}$ such that $\mathbf{x}_1 = f_1(\mathbf{x})$ presents how each element of $\mathbf{x}$ is mapped to four elements in $\mathbf{x}_1$ according to Table 1.

Obviously, for each path, which has been followed by a signal, a complementary path can be defined, see Figure 2. The

Table 1: Mapping table for each bit of the challenge vector $\mathbf{x}$.

$x[i]$	u_i	$x_1[4i-3]$	$x_1[4i-2]$	$x_1[4i-1]$	$x_1[4i]$
0	0	1	0	0	0
0	1	0	0	0	1
1	0	0	1	0	0
1	1	0	0	1	0

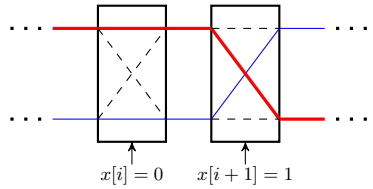

Figure 2: A path and its complementary path for two challenge bits are shown. The functions f_1 and f_2, as presented in Table 1 and Table 2, are used to define these paths more systematically.

Table 2: Challenge mapping table for the complementary path, when the challenge vector $\mathbf{x}$ is applied.

$x[i]$	u_i	$x_2[4i-3]$	$x_2[4i-2]$	$x_2[4i-1]$	$x_2[4i]$
0	0	0	0	0	1
0	1	1	0	0	0
1	0	0	0	1	0
1	1	0	1	0	0

mapping $f_2 : \{0,1\}^n \to \{0,1\}^{4n}$, such that $\mathbf{x}_2 = f_2(\mathbf{x})$ explains the relationship between $\mathbf{x}_2$ and the challenge vector applied to the arbiter PUF, see Table 2. Hence, choosing a path or its complementary results in obtaining the overall delay on the lower or upper path at the output of the last stage, i.e., b_1 or b_2.

The above mentioned definitions of a path and its complementary path enable a very useful and efficient representation of the linear (more specifically, knapsack) behavior of arbiter PUFs. To this end, let the column vector $\mathbf{a} \in \mathbb{Z}^{4n}$ be composed of delays of all stages, i.e.,

$$\mathbf{a} = (a_{1,1}, a_{1,2}, a_{1,3}, a_{1,4}, \ldots, a_{n,1}, a_{n,2}, a_{n,3}, a_{n,4})^{\mathrm{t}}. \quad (1)$$

Regarding the definitions of the functions f_1 and f_2, as well as the linear challenge-response behavior of an arbiter PUF, the total delay b_j at the end of the last stage can be given concisely by the equation

$$b_j = \sum_{k=1}^{4n} x_j[k]a[k] = \mathbf{x}_j \cdot \mathbf{a}. \quad (2)$$

Now consider the column vector $\mathbf{b} \in \{0, \ldots, M\}^{2m}$, whose elements are the total delays at the end of the last stage on the upper and the lower path obtained for m challenges:

$$\mathbf{b} = (b_1^1, \ldots, b_1^i, \ldots, b_1^m, b_2^1, \ldots, b_2^i, \ldots, b_2^m)^{\mathrm{t}}. \quad (3)$$

Let $\mathbf{X} \in \{0,1\}^{2m \times 4n}$ be the matrix of m mapped challenges, i.e., its rows are the m challenges mapped by applying the functions f_1 and f_2. Thus, we have

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^1 \\ \vdots \\ \mathbf{x}_1^m \\ \mathbf{x}_2^1 \\ \vdots \\ \mathbf{x}_2^m \end{pmatrix} \quad (4)$$

which gives the following compact (knapsack-like) form of an arbiter PUF

$$\mathbf{b} = \mathbf{X} \cdot \mathbf{a}. \quad (5)$$

2.3 XOR Arbiter PUFs

XOR arbiter PUFs have been developed to improve the security of arbiter PUFs, and more precisely, to mitigate different types of attacks, ranging from ML attacks to fully invasive attacks. It has been proposed that the outputs of k arbiter chains, having the same number of PUF stages (see Figure 3), should be XOR-ed together to generate the final output of the PUF, cf. [34].

Obviously, the final response of an XOR arbiter PUF to a given challenge $\mathbf{x}$ can be simply defined as

$$f_{XOR}(\mathbf{x}) = \bigoplus_{j=1}^{k} f_{j^{\text{th}} \text{ arbiter PUF}}(\mathbf{x}).$$

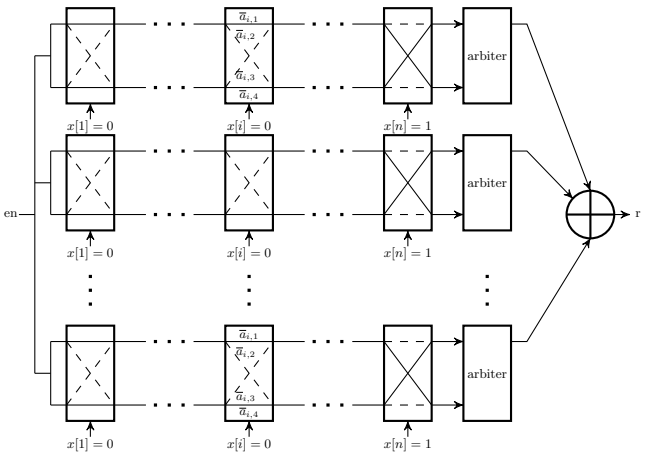

Figure 3: An XOR arbiter PUF has k parallel arbiter chains, each with n stages and an arbiter. The joint binary response is generated by XOR-ing the responses of all individual arbiter chains.

But regardless of having the XOR operator, the total delay at the last stage of each individual arbiter PUF can still be measured, e.g., following the procedure introduced by Tajik et al. [35] (see Section 3 for more details). This crucial observation is central to the following linearization of the XOR arbiter PUF.

In order also to obtain a system of linear equations explaining the functionality of an XOR arbiter PUF, the equations presented for an arbiter PUF can be extended as follows. First, we define the matrix $\mathbf{A} \in \{0, \ldots, M\}^{4n \times k}$ in which the i^{th} column corresponds to the integer-valued delays of the i^{th} arbiter PUF chain. This means that we have

$$\left(\mathbf{a}^1, \ldots, \mathbf{a}^k\right) := \begin{pmatrix} a_{1,1}^1 & \cdots & a_{1,4}^1 & \cdots & a_{n,1}^1 & \cdots & a_{n,4}^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{1,1}^k & \cdots & a_{1,4}^k & \cdots & a_{n,1}^k & \cdots & a_{n,4}^k \end{pmatrix}^{\mathrm{t}}, \quad (6)$$

where $a_{i,j}^l \in [0, M]$ for $1 \le i \le n$, $1 \le j \le 4$, and $1 \le l \le k$. Similarly, the matrix $\mathbf{B} \in \{0, \ldots, M\}^{2m \times k}$ is defined as a matrix composed of the accumulated delays at the output of the last stage (for upper and lower path) of all k arbiter chains, when applying m challenges, i.e.,

$$\left(\mathbf{b}^1, \ldots, \mathbf{b}^k\right) := \begin{pmatrix} b_1^{1,1} & \cdots & b_1^{1,m} & b_2^{1,1} & \cdots & b_2^{1,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_1^{k,1} & \cdots & b_1^{k,m} & b_2^{k,1} & \cdots & b_2^{k,m} \end{pmatrix}^{\mathrm{t}}. \quad (7)$$

Thus, we obtain again a very compact (and linear) form of an XOR arbiter PUF

$$\mathbf{B} = \mathbf{X} \cdot \mathbf{A}. \quad (8)$$

2.4 Lattices

Here we give only a very brief outline of the notion of lattices as well as the closely related field of subset sum problems, and especially the hidden subset sum problem. For a detailed introduction we refer the reader to [6, 22, 24].

Definition 2. *A set L is an integral lattice of dimension m if it is a discrete additive subgroup of $\mathbb{Z}^m$.*

Therefore, there exist n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^m$ such that the integer linear combinations of these vectors generate L, or in other words

$$L = L(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \cdot \mathbf{b}_i \;\middle|\; x_i \in \mathbb{Z} \right\}.$$

We call $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ a basis of L and n the rank of the lattice. For $m = n$, the lattice is said to have full rank. For a lattice L we denote by $\det(L)$ its determinant and by $\lambda_1(L)$ the Euclidean length of a shortest non-zero vector of L. The main result from Minkowski states that

$$\lambda_1(L) \leq \sqrt{n} \cdot \det(L)^{1/n}$$

for a lattice L of rank n. The Gaussian volume heuristic, cf. [24], suggests also that "random" lattices act well-behaved. It postulates that for a random lattice L of rank n we can assume that

$$\frac{\lambda_1(L)}{\sqrt{n} \cdot \det(L)^{1/n}} = \Theta(1).$$

Definition 3. *For a lattice $L \subseteq \mathbb{Z}^m$, its orthogonal lattice $L^\perp$ is defined as the set of vectors in $\mathbb{Z}^m$, which are orthogonal to all elements of L with respect to the inner product.*

We also need to define $\bar{L}$, the orthogonal lattice of $L^\perp$, i.e., the lattice $\bar{L} = (L^\perp)^\perp$. For a lattice $L \subseteq \mathbb{Z}^m$ it holds that $\det(L^\perp) = \det(\bar{L})$.

2.5 The Hidden Subset Sum Problem

The Hidden Subset Sum problem (HSS) was introduced in 1998 as a variant of the subset sum problem, where the accumulated sum is known, but the set of summands is secret, i.e., hidden [5].

Definition 4. *Given a positive integer $M \in \mathbb{Z}$ and a vector $\mathbf{b} = (b_1, \ldots, b_m) \in \mathbb{Z}_M^m$, the Hidden Subset Sum problem is to find integers $a_1, \ldots, a_n \in \mathbb{Z}_M$ so that each b_i is a subset sum modulo M of the summands $a_1, \ldots, a_n$, i.e., there exist binary vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \{0,1\}^m$ and a vector $\mathbf{k} \in \mathbb{Z}^m$ satisfying*

$$\mathbf{b} = a_1\mathbf{x}_1 + a_2\mathbf{x}_2 + \ldots + a_n\mathbf{x}_n + M\mathbf{k}.$$

2.5.1 Solving the Hidden Subset Sum Problem with Lattices

In 1999, Nguyen and Stern presented a cryptanalytical attack against the hidden subset sum problem in a breakthrough work, cf. [23]. Specifically, they described a lattice-based attack against the HSS, upon which our attack against XOR arbiter PUFs is build. Their attack has three steps. First, from the known vector $\mathbf{b}$, the lattice $\bar{L} = (L^\perp)^\perp$ for $L = L(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{k})$ with $\mathbf{b} \in L$ is determined. Due to the random construction of the underlying HSS, it is plausible to assume that the vectors $\mathbf{x}_i$ and $\mathbf{k}$ are linearly independent. To determine $\bar{L}$, first a reduced basis of $\mathbf{b}^\perp$ is computed. The lattice $\mathbf{b}^\perp$ has rank $m-1$. Afterwards, a basis for the lattice $\bar{L}$, which is "very likely" spanned by the "shortest" $m-(n+1)$ vectors of that reduced basis, is computed. The lattice $\bar{L}$ has rank $n+1$. This step is expected to succeed only if the quantity

$$n / \log_2(M)$$

is very small, which is true if M is very large. Second, from $\bar{L}$, the secret binary vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are derived. By applying the former Gaussian volume heuristic, these vectors are assumed to be short lattice vectors due to their binary entries, hence they are derived from a reduced basis of $\bar{L}$. Since vectors with entries 0, 1, and -1 might also be short vectors in such a reduced basis, the lattice is slightly modified to find the correct binary vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$. These two steps are depicted in Algorithm 1. Third, using the known vectors $\mathbf{b}, \mathbf{x}_1, \ldots, \mathbf{x}_n$, and M, the secret summands $a_1, \ldots, a_n \in \mathbb{Z}_M$ are recovered. This can be performed by adopting any approach to solve a system of modular linear equations.

Algorithm 1 Algorithm for disclosing the hidden coefficients of a random HSS as proposed by [23]

Require: Vector $\mathbf{b} \in \mathbb{Z}^m$, m, and n
Ensure: Vectors $\mathbf{x}_i$, $1 \leq i \leq n$

1: Calculate the orthogonal lattice $\mathbf{b}^\perp$
2: Compute a reduced basis of $\mathbf{b}^\perp$, i.e., $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{m-1}) := BKZ(\mathbf{b}^\perp, blocksize = 20)$
3: Compute the hidden lattice: $\bar{L}_\mathbf{x} := BKZ((\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{m-n-1})^\perp, blocksize = 20)$
4: $L'_\mathbf{x} = 2\bar{L}_\mathbf{x} + \mathbb{Z} \cdot (1, \ldots, 1)$
5: $BKZ(L'_\mathbf{x}, blocksize = 20)$
6: Find the basis vectors of $L'_\mathbf{x}$ with the form $\pm(2\mathbf{x}_i - (1, \ldots, 1))$
7: **return** $\mathbf{x}_i$

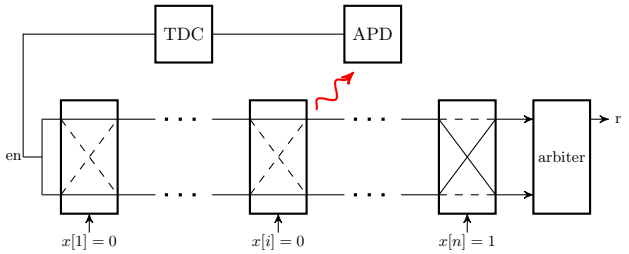

Figure 4: A high resolution Time to Digital Converter (TDC) measures the duration between the triggering time of the PUF and the time when a photon from the first output of the i^{th} stage is captured by an Avalanche Photodiode.

3. PHYSICAL DATA ACQUISITION FOR ARBITER PUF ENTITIES

Our lattice basis reduction can be launched only if the adversary can measure the delays at the output of the last stages, and more generally, between any two arbitrary chosen stages. We explain here how to achieve this in practice for a given PUF implemented on a chip.

The inventors of arbiter PUFs have assumed that an attacker cannot directly measure the individual delays of stages. Moreover, they also assume that the delays could only be measured by applying fully-invasive techniques, which would lead to physical changes, and finally to substantial changes in the PUF and its intrinsic challenge/response behavior. Therefore, in their scenario the attacker is only able to set the challenges and read their respective responses.

However, it has been shown that different side channel analyses can indeed reveal the delays at the outputs of the last stage of the PUF [3, 30, 35]. Power analysis of delay-based PUFs enables the attacker to measure the delay difference between the PUF excitation time and the logical transitions time of the latch (i.e., the arbiter) terminating the chain [3, 30]. The delays and the binary response of the

PUF can be extracted by gathering several power traces for a single challenge and applying statistical signal processing techniques on them to reduce the noise of measurements. The main limitation of power analysis is that the delays of individual upper and lower paths of an arbiter PUF cannot be derived from power traces. Moreover, in the XOR arbiter PUF architecture, the delays of each individual arbiter chain cannot be differentiated in the traces.

On the other hand, timing side channel analyses enable the attacker to measure the delays of both upper and lower path individually [30]. To collect the timing information, the attacker has to access the dedicated debugging clock sweeping circuits on the chip, which are embodied in the chip by the manufacturer [20]. By changing the frequency of the clock, which is swept through the arbiter chains of the XOR arbiter PUF, the attacker can measure the delays of each path. Although timing side channels enable the attacker to measure the delays with picosecond resolution, the assumption that the attacker can utilize this kind of clock sweeping circuit might not be valid in a real attack scenario.

In order to measure the delays of the paths without using any extra circuitry, the high resolution temporal photonic emission analysis from the IC backside can be utilized, cf. [35]. In this case, the attacker can measure the delay between the enabling time of the PUF and the emission time of photons from a CMOS transistor by using an Avalanche Photodiode (APD) at the outputs of *any arbitrary* stage, see Figure 4. XOR arbiter PUFs can be implemented effectively on programmable logic devices, such as Complex Programmable Logic Devices (CPLDs) and Field Programmable Gate Arrays (FPGAs). On such platforms, each stage of the arbiter PUF is implemented by two logic elements (LEs), see Figure 5 (a). The location of different combinational circuits, e.g., an arbiter PUF, can be found on the chip with the help of spatial photonic emission analysis [35, 36]. After finding the PUF circuitry and the respective input/outputs of each stage, the delay between the inputs of the i^{th} stage and outputs of the j^{th} stage can be measured in three steps.

First, the attacker measures the delay between the excitation time of the electrical pulse on the enable input of the chip and the photon emission time of input transistors of the i^{th} LE, see Figure 5 (b). Second, the delay between the enabling time and the photon emission time of the output transistors of the j^{th} stage has to be measured in a similar fashion, see Figure 5 (b). Third, by subtracting these two measured values, the desired delay value between two stages can be derived.

It has been proved that an arbiter PUF can be physically characterized by measuring just the total delays at the end of each path for $n+1$ challenges, where the difference of their Hamming distances is equal to 1 [35]. Based on this technique, a comparison between the reference challenge and any arbitrary challenge can be made to predict the delay at the end of the path, and hence, the response. The same methodology can be deployed to physically characterize XOR arbiter PUFs, regardless of the number of parallel arbiter chains, by simply controlling two APDs for each individual arbiter chain. To this end, each arbiter chain is characterized individually. The main drawback of this attack is that the adversary has to control the challenge in order to set $n+1$ challenges, with Hamming distance from one another equals 1.

However, if there is no electrical access to the challenges of the PUF (such as controlled PUFs), the attacker faces the problem of so called hidden challenges and obtains finally only the total delays corresponding to the applied, but hidden challenges. Nevertheless, the attack proposed in this paper does not require any special challenges (in terms of the Hamming distance) to be applied. To launch this attack, the attacker measures only the delays at the outputs of the desired stages by programming the coordination of the optical objective, and consequently, automatically moving them to the desired positions to capture the emitted photons.

Although the measurement noise seems to have a destructive impact on the effectiveness of our attack, it is irrelevant due to the following reason. The precision of the measurement setup used for the raw physical data acquisition is naturally higher than the precision of the arbiter terminating each chain. Hence, the impact of the measurement noise is irrelevant, as the delays are mapped to integer-valued delays (see Section 2.2), where the mapping absorbs the impact of measurement noise.

4. EXTENDED HIDDEN SUBSET SUM PROBLEM

In this section we aim to reveal the hidden coefficients of a controlled XOR arbiter PUF by using the techniques introduced in [23] as well as Section 2.5 of this paper.

In order to explain how our proposed attack can be launched on XOR arbiter PUFs, we begin by comparing Eq. (5) for an ordinary arbiter PUF with the definition of the hidden subset sum as given by Definition 4. Excluding minor differences, which will be resolved in this section, our intentionally chosen notations show a strong similarity between a PUF in the form of $\mathbf{b} = \mathbf{X} \cdot \mathbf{a}$ and the hidden subset sum problem. As previously explained, for a given physical PUF (i.e., in the form of a real circuit) and its abstract model represented by the vector $\mathbf{a}$, it is indeed possible to construct the total delay vector $\mathbf{b}$ by simply measuring certain signal propagation delays, without even knowing the set of challenges applied to the PUF. This means that the matrix of mapped challenges $\mathbf{X}$ is hidden and solely the vector $\mathbf{b}$ is accessible. Nevertheless, we know that the vector $\mathbf{b}$ is constructed according to Eq. (5) and along with all the specific subtleties of a given arbiter PUF. According to this sketch, the relationship between an arbiter PUF in the form of Eq. (5) and the HSS becomes obvious. However, the considerable differences between the HSS and an XOR arbiter PUF modeled by Eq. (8) should be carefully taken into account. Section 4.1 aims to consider these differences. Section 4.2 is devoted to investigation of the multidimensional HSS as another novelty from us, which follows the requirements of our lattice basis reduction attack. Until today, this highly interesting extension of the HSS has been completely unknown. The principles underlying the multidimensional HSS are inspired by Nguyen's work [23], however, the giant step related to the calculation of a new upper bound on the weights of the multidimensional HSS is completely performed in our paper.

4.1 Adjustments

Here we focus on resolving the minor difference between the model of an arbiter PUF (Eq. (5)) and the HSS.

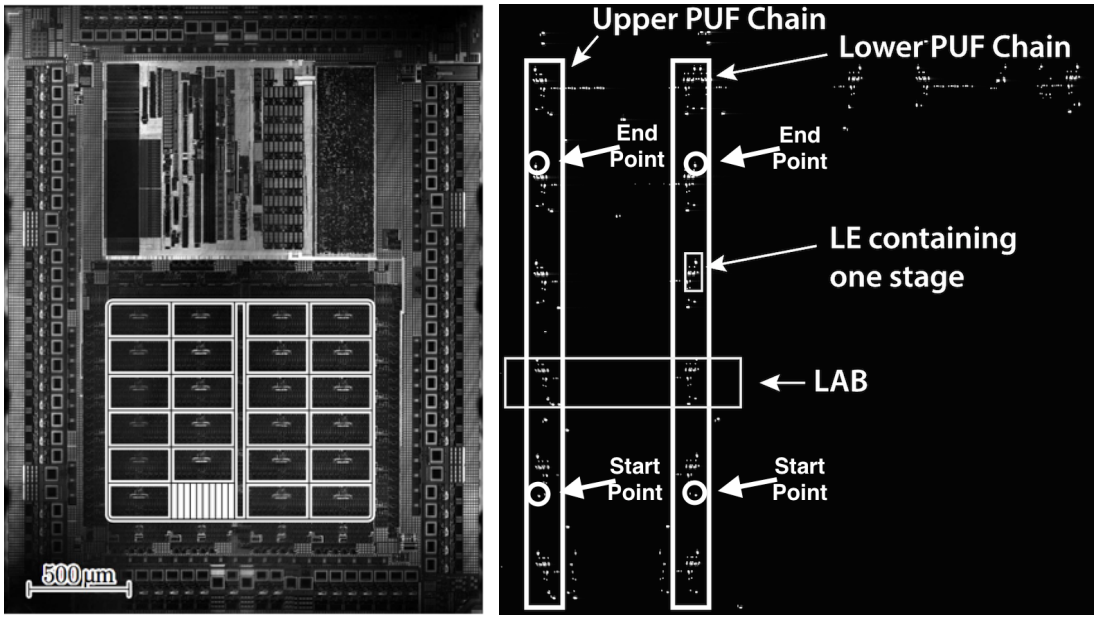

Figure 5: (a) The reflected backside image of an Altera MAX V CPLD based on a 180 nm technology, captured by a laser scanning microscope [36]. The chip consists of Logic Array Blocks (LABs). Each LAB contains 10 Logic Elements (LEs). (b) The emission image of an 8-bit arbiter PUF on the MAX V CPLD, captured by a Charge-Coupled Device (CCD) camera [35]. The PUF circuit is enabled by a 4MHz signal. Each bright spot shows a switching NMOS or PMOS transistor, which emits photons. The attacker can measure the signal propagation delays between start points and end points by moving the objective to the inputs and outputs of the desired LEs.

4.1.1 Excluding the Modulo Operation

Eq. (5) is an integral equation, whereas the HSS is defined in terms of modular arithmetic. Reconsidering the respective mathematical analysis in Section 2.5.1 reveals that only the rank of the lattice $\bar{L}$ would change in this non-modular case, from $n+1$ to n. This means that asymptotically no further change in the mathematical analysis is required. Hence, this minor difference is easily resolved.

4.1.2 Distribution of the Vector $\mathbf{a}$

From Section 2.2, it can be understood that the vector $\mathbf{a}$ is generated according to a specific Gaussian distribution such that the entries of $\mathbf{b}$ can be bounded by M in their expectation value. To explain our lattice basis reduction attack in an easier manner, we will assume a slightly different situation. However, we emphasize that for the real distribution a more complex analysis can be performed that would yield an (asymptotically) equivalent result. Moreover, due to the heavy use of the Gaussian volume heuristic for the resulting random lattices, this fine distinction is anyhow quite superficial and therefore omitted. Hence, we can simplistically assume that the elements of the vector $\mathbf{a}$ are uniformly chosen from the set $\{0,\ldots,M\}$, and thus we can follow the original HSS analysis of [23].

4.1.3 Mapped Challenges $\mathbf{X}$ are not Uniformly Chosen

Although we assume (as in a real PUF setting) that the challenge bits are drawn uniformly from the set $\{0,1\}$, this is not valid for the matrix of mapped challenges $\mathbf{X}$. Indeed, all its respective challenge vectors $\mathbf{x}_j^l \in \{0,1\}^{4n}$, $1 \leq l \leq m$ and $1 \leq j \leq 2$, have the *fixed* Hamming weight n. Resolving this difference with the original HSS results in three changes made in Algorithm 1 (see Section 2.5.1):

1. The lattice $L'_{\mathbf{x}}$ as defined in step 4 of Algorithm 1 has to be changed to $L'_{\mathbf{X}} = 16\overline{L}_{\mathbf{X}} + \mathbb{Z}\cdot(1,\ldots,1)$. We refer the reader to [8] for a theoretical explanation.
2. In contrast to the approach in [23], the multidimensional HSS is analyzed.
3. Three centering groups of mapped challenges are applied in our attack scenario.

The first two of these points account for the specific Hamming weight, whereas the third one addresses the non-uniform distribution. We explain the second and the third changes step-by-step.

4.2 Main Result: The Multidimensional HSS

As outlined in Section 2.5.1 and stressed in [23], Algorithm 1 is able to solve a random HSS only if the respective quantity M is very large, i.e., M must be exponential in n. However, as explained in Section 2 and very carefully in [10], for arbiter PUFs the value M representing the maximum variation of delays is in the order of $O(n)$ in the best case scenario. Looking closer at our attack scenario for XOR arbiter PUFs, we aim at solving Eq. 8, where the unknown matrices $\mathbf{X}$ and $\mathbf{A}$ are chosen randomly as explained previously. The difference is that we now have k vectors $(\mathbf{a}^1,\ldots,\mathbf{a}^k)$, instead of a single one as in the ordinary HSS problem. Assuming this enlarged system, we have to go through the explanation of [23], and adapt their Section 4.1 to our case.

To this end, we begin in this extended case from the lattice $\{\mathbf{b}^1,\ldots,\mathbf{b}^k\}^\perp$ of rank $2m-k$. Applying a strong lattice basis reduction algorithm, we aim to obtain from this lattice the

lattice $\overline{L}_{\mathbf{X}}$, whose rank is with high probability $4n$, what we assume to be true from now on. To ensure the success of the lattice basis reduction algorithm, we first have to estimate λ_1 of the lattice $\{\mathbf{a}^1,\ldots,\mathbf{a}^k\}^{\perp}$. Using the Hadamard inequality yields that $\det(\{\mathbf{a}^1,\ldots,\mathbf{a}^k\}^{\perp}) \leq O((M\sqrt{n})^k)$. Assuming now the Gaussian volume heuristic to be fulfilled, we can estimate the first successive minimum by

$$\begin{aligned} & \lambda_1(\{\mathbf{a}^1,\ldots,\mathbf{a}^k\}^{\perp}) \\ \approx\ & \sqrt{4n-k}\ \left(\det(\{\mathbf{a}^1,\ldots,\mathbf{a}^k\}^{\perp})\right)^{1/(4n-k)} \\ =\ & O\left(\sqrt{4n-k}\left(M\sqrt{n}\right)^{k/(4n-k)}\right). \end{aligned}$$

Next, regarding the construction of the lattice $\{\mathbf{b}^1,\ldots,\mathbf{b}^k\}^{\perp}$ we know that it also contains the lattice $(L_{\mathbf{X}})^{\perp}$ of rank $2m-4n$. According to the Gaussian volume heuristic, the vectors of any reduced basis of $(L_{\mathbf{X}})^{\perp}$ have an expected length of

$$\sqrt{2m-4n}\left(\left(\sqrt{O(m)}\right)^{4n}\right)^{1/(2m-4n)}.$$

Now we can follow the procedure in [23]. Namely, the first $2m-4n$ basis vectors of any reduced basis of $\{\mathbf{b}^1,\ldots,\mathbf{b}^k\}^{\perp}$ are with a high likelihood short lattice points of $(L_{\mathbf{X}})^{\perp}$, if the following condition is fulfilled

$$\begin{aligned} & \sqrt{O(m)}\cdot\sqrt{2m-4n}\left(\left(\sqrt{O(m)}\right)^{4n}\right)^{1/(2m-4n)} \\ \ll\ & \lambda_1(\{\mathbf{a}^1,\ldots,\mathbf{a}^k\}^{\perp}). \end{aligned}$$

Letting now m and n be sufficiently large, we can get the simpler expression

$$O\left(\frac{(\sqrt{m})^{(4n-k)/k}}{\sqrt{n}}\right) \ll M.$$

The setting $k=n$ gives us a very small bound for M, namely

$$M \gg O(m^{1.5}).$$

Note that several constants are removed and the Gaussian volume heuristic is taken into account particularly. Hence, this result on the bound for M should be interpreted only as an asymptotic value, indicating that the lattice basis reduction approach can be applied for very small M. Our experimental results strongly confirm this.

4.2.1 Non-Uniformity of $\mathbf{X}$

Now we describe how we address the issue of having the non-uniform matrix $\mathbf{X}$. We have learned from our experimental results that the fixed structure of the matrix $\mathbf{X}$ can be a serious issue since it cannot provide enough randomness to confirm all our assumptions made previously. To provide "somehow" enough randomness in the matrix of mapped challenges $\mathbf{X}$, we applied the following strategy— inspired by our specific physical side channel analysis.

For each original n-bit challenge, we define 3 smaller random sub-challenges with expected length $n/4$, symmetrically centered on $\lceil n/4\rceil$, $\lceil 2n/4\rceil$, and $\lceil 3n/4\rceil$ stages, respectively. Afterwards, as explained in Section 2, we can map each sub-challenge with expected length $n/4$, assuming that the other remaining bits are simply set to 0. Hence, we obtain three different randomly chosen mapped challenges (out of each real random challenge) with very small Hamming weight, which we call centering groups. Note that this random grouping strategy can be implemented by our data acquisition methodology as explained before.

5. EXPERIMENTAL SETUP AND RESULTS

Our proposed lattice basis reduction attack is implemented in Magma (V2.17) [1,4], installed on a virtual AMD64 server having 16 cores and 32 GB of RAM. The specific single physical core is an Intel Core 2 Duo (Penryn) running at 2,4 GHz. The Magma software package provides substantial support for computing reduced bases of lattices. Of crucial importance for our purposes is the implementation of the Block Korkine-Zolotarev (BKZ) algorithm provided in this package [31]. In this implementation, in addition to the block size, the parameter δ that controls the quality of the basis reduction algorithm can be set. A drawback of this implementation in Magma is that parallel computation is impossible. Hence, the computation is performed only on a single core of our virtual AMD64 server.

We launched our attack on simulated CRPs, generated in Magma. However, it is worth noting that our XOR arbiter PUF simulator implemented in Magma adopts only real experimental data obtained in [35]. Regarding those results and the proof provided in [10], the delay values are mapped to integer values, lying within a limited interval (see Section 2). Our XOR arbiter PUF simulator first generates the matrix $\mathbf{A}$ composed of integer-valued delays, which are uniformly distributed within the interval $[0,M]$. Although it is known that the delays are normally distributed in reality, uniformly distributed delays can be thought of being more complicated, and thus present a harder practical case. The matrix $\mathbf{B}$ is then calculated according to Eq. (7) and fed into the algorithm, whose main steps are depicted in Algorithm 2.

Comparing Algorithm 1 and Algorithm 2, the major modifications made to improve the results quantitatively and qualitatively are the following:

1. Instead of applying the BKZ function with a constant block size, we perform this function by increasing the block size in an iterative manner. Indeed, it has been proved that the iterative reduction with respective block size $2,4,\ldots,2^t$ performs twice as fast as the direct reduction with block size 2^t, cf. [32, 33]. The parameter t is set to 5, resulting in a maximal block size of 32.
2. Before each reduction step the bases are randomly permuted to guarantee a better quality of the reduced bases, cf. [33].
3. In order to further improve the quality of the reduced bases in terms of their overall length, the step corresponding to the final computation of the hidden coefficients is repeated five times for randomly permuted bases.

Finally, when constructing the lattice L'_X, the required coefficient is also changed from 2 to 16 to account for the shorter average length of the mapped challenge vectors (for more details see Section 2.2 and [8, 23]).

Our experimental results for different settings are shown in Table 3. In addition to the setting of the experiments, we report the rank of the matrices composed of the measurements that are centered on three different stages of the XOR arbiter PUF (three centering groups). Moreover, the number

of hidden coefficients (i.e., columns of the matrix $\mathbf{X}$) disclosed by applying our algorithm on each of these matrices is shown as well. The total number of disclosed coefficients is counted separately in order to avoid having common disclosed vectors in the three centering groups. However, we have observed that no common disclosed vector is found in the three groups.

The number of arbiter chains is set to $k = n$, although our approach also works for $k \geq \Omega(n)$ and even works provably better in this case. However, we aim to further elaborate on this specific setting, since all known machine learning attacks cannot be launched in polynomial time, if $k \geq \omega(\ln n)$. An initial and empirical study on XOR arbiter PUFs with $n = 512$ and $k = 16$ has been performed by [30], but no solid and fundamental research has been conducted in the case of $\omega(\ln n) < k \leq O(n)$. Although by applying the Gaussian volume heuristic we have proved that our algorithm can be applied to disclose the hidden coefficients in the case of $k = O(n)$, it might also be useful in the more simplified scenario, where $\ln n \ll k \ll n$. Given all that, we decided to focus on disclosing the matrices $\mathbf{X}$ and $\mathbf{A}$ in the complicated case of $k = n$. We stress that once $\mathbf{X}$ is known, a simple standard Gaussian elimination will also reveal $\mathbf{A}$.

The setting corresponding to $n = 11$ represents a virtually similar example as shown by Nguyen et al. [23]. They have demonstrated that their attack can be launched successfully against a random HSS, with the number of hidden coefficients equal to 45, $m = 90$, and $M = 2^{160}$. Our attack can disclose the hidden coefficients even when $m = 78$ and $n = 11$ (i.e., the number of hidden coefficients is 44), but of course for a significantly smaller number of bits, $M = 2^6$ in our case. This is in line with our former proof that the input to our algorithm is the matrix $\mathbf{B}$, whereas only a single vector $\mathbf{b}$ is fed into the algorithm proposed by [23] (see Section 2.5.1 for a detailed explanation).

We conducted further experiments on XOR arbiter PUFs for $n = 8$, $n = 16$, $n = 24$, and $n = 32$. For $n = 8$ and $n = 16$, our algorithm can entirely disclose the matrix $\mathbf{X}$, and thereby $\mathbf{A}$. An increase in the number of XOR arbiter PUF stages naturally leads to a slight increase in the number of measurements as well as the time required for launching the attack, as shown for $n = 24$ and $n = 32$. The impact of the improvements that we make to Algorithm 1 proposed by Nguyen et al. [23] can be clearly seen in the results for $n = 24$ and $n = 32$. Whereas their algorithm cannot disclose more than 90 hidden coefficients, our algorithm can disclose 123 hidden coefficients for significantly smaller M, corresponding to the case of $n = 32$ in an XOR arbiter PUF with 32 parallel arbiter chains. Note also that the computation times required to conduct the experiments are ranging from only a few minutes to a few hours.

Although we performed our superior lattice basis reduction approach only against XOR arbiter PUFs with the maximum $n = 32$ and $k = 32$, it is clear that these values are not its final limit. Indeed, comparing with other approaches, e.g., the very recent ones [30] and [37], we are only limited by the computational power of our server having an old CPU and only 32 GB of RAM. Namely, the authors of [30] used a more modern system providing 48 GB of RAM, while a system with 256 GB of RAM has been employed in [37]. Specifically, the authors of [37] used their system to prove that XOR arbiter PUFs with $n = 128$ and $k = 8$ are indeed "learnable", which has been already implied in [10].

Algorithm 2 Our algorithm for disclosing the hidden bits of the challenge vectors

Require: Matrix $\mathbf{B}$, $2m$, and $4n$
Ensure: Matrix $\mathbf{X}$

Calculate the orthogonal lattice $\mathbf{B}^{\perp}$
$L^* = \mathbf{B}^{\perp}$
$blocksize = 2$
while $blocksize \leq 32$ **do**
  $(\mathbf{u}_1, \mathbf{u}_2, \ldots) := BKZ(L^*, blocksize)$
  Randomly permute the current basis $(\mathbf{u}_1, \mathbf{u}_2, \ldots)$ of L^*
  $blocksize = 2 * blocksize$
end while
/* Compute the hidden lattice */
$L^* = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{2m-4n})^{\perp}$
Apply steps 4-8 on lattice L^*
$\overline{L}_{\mathbf{X}} = L^*$
/* Compute the hidden coefficients */
$L'_{\mathbf{X}} = 16\overline{L}_{\mathbf{X}} + \mathbb{Z} \cdot (1, \ldots, 1)$
$L^* = L'_{\mathbf{X}}$
$z = 1$
while $z \leq 5$ **do**
  Apply steps 4-8 on lattice L^*
  Find the basis vectors of L^* with the form $\pm\left(16\mathbf{x}_j^i - (1, \ldots, 1)\right),\ 1 \leq j \leq 2,\ 1 \leq i \leq 2m$
  Randomly permute the current basis of L^*
  $z = z + 1$
end while
return X

Thus, ignoring certain O-constants, it is obvious that when having an 8-fold increase in our RAM to achieve the computational power virtually similar to [37], our lattice method should easily break the security of XOR arbiter PUFs with $n = 4 \cdot 32$ and $k = 4 \cdot 32$ — particularly in the case of controlled PUFs.

Last but not least, we would like to mention that we have just applied standard lattice basis reduction algorithms provided by Magma, namely the old but most appropriate and available BKZ reduction. Considering now that in the lattice basis reduction field several new breakthroughs have been provided by faster and better theoretical algorithms, cf. [7, 32, 33], we expect also new, strong, and practical reduction algorithms. For instance, the newly developed "BKZ 2.0" basis reduction algorithm will dramatically improve the effectiveness and the efficiency of our attack against XOR arbiter PUFs for larger n and k and most likely also for smaller k in such cases where the machine learning algorithms fail.

6. CONCLUSION

In this article, we proved the vulnerability of XOR arbiter PUFs to a new attack, which is a combination of a lattice basis reduction attack and a photonic side channel analysis. The success of previous modeling attacks primarily depends on the limitation in the number of arbiter chains. Moreover, popular semi-invasive attacks against PUFs often rely on the direct access to challenges and/or responses. We demonstrated that neither an increase in the number of arbiter chains nor the limitation of the access to challenges and responses qualifies as a secure countermeasure against our novel hybrid attack. Therefore, our study contributes to bridging the gap between these two types of attacks, and characterizing an XOR arbiter PUF featuring an arbitrarily large number of arbiter chains, even when the challenges and responses are not accessible.

Table 3: Results of the lattice basis reduction attack on XOR arbiter PUFs with the different number of stages and arbiter chains

Setting	Results		
$n = 8$, $k = 8$, $m = 60$, $M = 2^5$	Rank of the matrix $\mathbf{B}$ centered on	$\lceil n/4 \rceil$	22
		$\lceil n/2 \rceil$	32
		$\lfloor 3n/4 \rfloor$	24
	Number of disclosed vectors from the group centered on	$\lfloor n/4 \rfloor$	6
		$\lceil n/2 \rceil$	19
		$\lfloor 3n/4 \rfloor$	7
	Total number of disclosed vectors		32
$n = 11$, $k = 11$, $m = 78$, $M = 2^6$	Rank of the matrix $\mathbf{B}$ centered on	$\lceil n/4 \rceil$	36
		$\lceil n/2 \rceil$	44
		$\lfloor 3n/4 \rfloor$	32
	Number of disclosed vectors from the group centered on	$\lfloor n/4 \rfloor$	10
		$\lceil n/2 \rceil$	26
		$\lfloor 3n/4 \rfloor$	8
	Total number of disclosed vectors		44
$n = 16$, $k = 16$, $m = 128$, $M = 2^8$	Rank of the matrix $\mathbf{B}$ centered on	$\lceil n/4 \rceil$	48
		$\lceil n/2 \rceil$	64
		$\lfloor 3n/4 \rfloor$	46
	Number of disclosed vectors from the group centered on	$\lfloor n/4 \rfloor$	17
		$\lceil n/2 \rceil$	32
		$\lfloor 3n/4 \rfloor$	15
	Total number of disclosed vectors		64
$n = 24$, $k = 24$, $m = 280$, $M = 2^{12}$	Rank of the matrix $\mathbf{B}$ centered on	$\lceil n/4 \rceil$	72
		$\lceil n/2 \rceil$	96
		$\lfloor 3n/4 \rfloor$	70
	Number of disclosed vectors from the group centered on	$\lfloor n/4 \rfloor$	23
		$\lceil n/2 \rceil$	50
		$\lfloor 3n/4 \rfloor$	21
	Total number of disclosed vectors		94
$n = 32$, $k = 32$, $m = 370$, $M = 2^{15}$	Rank of the matrix $\mathbf{B}$ centered on	$\lceil n/4 \rceil$	94
		$\lceil n/2 \rceil$	128
		$\lfloor 3n/4 \rfloor$	96
	Number of disclosed vectors from the group centered on	$\lfloor n/4 \rfloor$	26
		$\lceil n/2 \rceil$	67
		$\lfloor 3n/4 \rfloor$	30
	Total number of disclosed vectors		123

In the realm of XOR arbiter PUFs we present a novel application of the concept of lattice basis reduction, which is a very powerful tool providing a firm basis for the assessment of the security of these PUFs. Our methodology of collecting physically measured data required for establishing a lattice-based representation of an XOR arbiter PUF is nowadays very well developed. We further extended a celebrated lattice basis reduction attack to the multidimensional case in order to launch it for higher densities, in contrast to its original very low density. Therefore, we were able to adapt the maximum variation of delays in the PUF (M) to the realistic and practical values, typically found in arbiter PUFs implemented on chips.

The importance of defining this maximum variation has been recognized and emphasized in a recent work [10]. It has been demonstrated that the maximum variation is one of the critical parameters for launching successful machine learning attacks against arbiter PUFs in general. More interestingly, afterward this has been observed and implicitly re-confirmed in a careful practical study [37]. Hence, regarding the results from [10] and [37], it may seem tempting to drastically increase M (by increasing the intrinsic delay variations and/or the final arbiter precision) to impair the effectiveness of machine learning attacks. While this provably protects a PUF against machine learning attacks, having a larger M even improves the effectiveness of our novel lattice basis reduction attack against XOR arbiter PUFs. As the "knapsack-density" eventually turned out to be *the* security-critical parameter for all knapsack-based cryptographic constructions, cf. [8, 24], it seems that M and its canonical density definition is also the Achilles heel of arbiter PUFs.

7. ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of the German Federal Ministry of Education and Research in the project Photon FX2 and the Helmholtz Research School on Security Technologies.

8. REFERENCES

[1] Magma Computational Algebra System. `http://magma.maths.usyd.edu.au/magma/`.

[2] F. Armknecht, R. Maes, A. Sadeghi, O.-X. Standaert, and C. Wachsmann. A Formalization of the Security Features of Physical Functions. In *Security and Privacy (SP), IEEE Symp. on*, pages 397–412, 2011.

[3] G. T. Becker and R. Kumar. Active and Passive Side-Channel Attacks on Delay Based PUF Designs. *IACR Cryptology ePrint Archive*, 2014:287, 2014.

[4] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational Algebra and Number Theory.

[5] V. Boyko, M. Peinado, and R. Venkatesan. Speeding up Discrete Log and Factoring Based Schemes via Precomputations. In *Advances in Cryptology - EUROCRYPT '98, Int. Conf. on the Theory and Application of Cryptographic Techniques*, pages 221–235, 1998.

[6] J. W. S. Cassels. *An Introduction to the Geometry of Numbers*, volume 99. Springer, 1996.

[7] Y. Chen and P. Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. In *Advances in Cryptology–ASIACRYPT 2011*, pages 1–20. Springer, 2011.

[8] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr, and J. Stern. Improved Low-Density Subset Sum Algorithms. *Computational Complexity*, 2(2):111–128, 1992.

[9] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede. Secure Lightweight Entity Authentication with Strong PUFs: Mission Impossible? In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 451–475. Springer, 2014.

[10] F. Ganji, S. Tajik, and J.-P. Seifert. PAC Learning of Arbiter PUFs, Security Proofs for Embedded Systems–PROOFS, 2014. `https://eprint.iacr.org/2015/378.pdf`.

[11] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Controlled Physical Random Functions. In *Computer Security Applications Conf., 2002. Proc. 18th Annual*, pages 149–160. IEEE, 2002.

[12] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Silicon Physical Random Functions. In *Proc. of the 9th ACM Conf. on Computer and Communications Security*, pages 148–160. ACM, 2002.

[13] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert. Cloning Physically Unclonable Functions. In *Hardware-Oriented Security and Trust (HOST), IEEE Intl. Symp. on*, pages 1–6, 2013.

[14] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert. Breaking and Entering through the Silicon. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 733–744. ACM, 2013.

[15] S. Katzenbeisser, Ü. Kocabaş, V. Van Der Leest, A.-R. Sadeghi, G.-J. Schrijen, and C. Wachsmann. Recyclable PUFs: Logically Reconfigurable PUFs. *Journal of Cryptographic Engineering*, 1(3):177–186, 2011.

[16] O. Kömmerling and M. Kuhn. Design Principles for Tamper-Resistant Security Processors. In *USENIX Workshop on Smartcard Technology*, 1999.

[17] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas. A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications. In *VLSI Circuits, 2004. Digest of Technical Papers. Symp. on*, pages 176–179, 2004.

[18] R. Maes. *Physically Unclonable Functions: Constructions, Properties and Applications*. Springer, 2013.

[19] R. Maes and I. Verbauwhede. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In *Towards Hardware-Intrinsic Security*, pages 3–37. Springer, 2010.

[20] M. Majzoobi, F. Koushanfar, and S. Devadas. FPGA PUF using Programmable Delay Lines. In *Information Forensics and Security (WIFS), IEEE Intl. Workshop on*, pages 1–6, 2010.

[21] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas. Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching. In *Security and Privacy Workshops (SPW), IEEE Symp. on*, pages 33–44, 2012.

[22] J. Martinet. *Perfect Lattices in Euclidean Spaces*, volume 327. Springer Science & Business Media, 2003.

[23] P. Nguyen and J. Stern. The Hardness of the Hidden Subset Sum Problem and Its Cryptographic Implications. In *Advances in Cryptology—CRYPTO'99*, pages 31–46. Springer, 1999.

[24] P. Q. Nguyen and B. Vallée. *The LLL Algorithm*. Springer, 2010.

[25] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical One-way Functions. *Science*, 297(5589):2026–2030, 2002.

[26] U. Rührmair, H. Busch, and S. Katzenbeisser. Strong PUFs: Models, Constructions, and Security Proofs. In *Towards hardware-intrinsic security*, pages 79–96. Springer, 2010.

[27] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber. Modeling Attacks on Physical Unclonable Functions. In *Proc. of the 17th ACM Conf. on Computer and Communications Security*, pages 237–249. ACM, 2010.

[28] U. Rührmair, J. Sölter, and F. Sehnke. On the Foundations of Physical Unclonable Functions. *IACR Cryptology ePrint Archive*, 2009:277, 2009.

[29] U. Ruhrmair, J. Solter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas. PUF Modeling Attacks on Simulated and Silicon Data. *Information Forensics and Security, IEEE Trans. on*, 8(11):1876–1891, 2013.

[30] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. Burleson. Efficient Power and Timing Side Channels for Physical Unclonable Functions. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 476–492. Springer, 2014.

[31] C.-P. Schnorr. A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms. *Theoretical Computer Science*, 53(2):201–224, 1987.

[32] C. P. Schnorr. Accelerated Slide- and LLL-Reduction. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:50, 2011. `http://eccc.hpi-web.de/report/2011/050`.

[33] C. P. Schnorr and T. Shevchenko. Solving Subset Sum Problems of Density close to 1 by "randomized " BKZ-reduction, 2012. `http://www.math.uni-frankfurt.de/~dmst/research/papers/Taras.pdf`.

[34] G. E. Suh and S. Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proc. of the 44th annual Design Automation Conf.*, pages 9–14. ACM, 2007.

[35] S. Tajik, E. Dietz, S. Frohmann, J.-P. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich. Physical Characterization of Arbiter PUFs. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 493–509. Springer, 2014.

[36] S. Tajik, D. Nedospasov, C. Helfmeier, J.-P. Seifert, and C. Boit. Emission Analysis of Hardware Implementations. In *Digital System Design (DSD), 17th Euromicro Conf. on*, pages 528–534. IEEE, 2014.

[37] J. Tobisch and G. T. Becker. On the Scaling of Machine Learning Attacks on PUFs with Application to Noise Bifurcation, 2015. `https://www.emsec.rub.de/research/publications/ScalingPUFCameraReady/`.