Learnability of Multiplexer PUF and S_N-PUF : A Fourier-based Approach

Durba Chatterjee, Debdeep Mukhopadhyay, and Aritra Hazra

Indian Institute of Technology Kharagpur, India
durba@iitkgp.ac.in, debdeep@iitkgp.ac.in, aritrah@cse.iitkgp.ac.in

Abstract

In this work, we prove that Multiplexer PUF (MPUF) and S_N-PUF are learnable in the PAC model. First, we show that both the designs can be represented as a function of Linear Threshold Functions. We show that the noise sensitivity of (n, k)-MPUF and S_N-PUF can be bounded by $O(2^k\sqrt{\epsilon})$ and $O(N\sqrt{\epsilon})$ respectively. Finally, we show that as a result of bounded noise sensitivity, both the designs can be accurately approximated using low degree algorithm. Also, the number of labelled examples (challenge-response pairs) required by the algorithm is polynomial in the input length and PAC model parameters.

1 Introduction

Ever since the emergence of Physically Unclonable Functions, the primitive has been subjected to an array of attacks starting from invasive to non-invasive attacks. One of the commonly applied non-invasive attacks on PUFs is model building attack. In this attack, an adversary uses empirical machine learning algorithms to create an accurate mathematical simulation of an instance. However, these algorithms do not allow a formal analysis of the primitive. The first formal mathematical framework for modelling of PUFs was proposed in [2]. It proves the learnability of several PUFs such as Arbiter PUF (APUF), XOR Arbiter PUF (XOR PUF), Ring Oscillator PUF (ROPUF), Bistable Ring PUF (BR-PUF) in the Provably Approximately Correct (PAC) model. An automated framework for the learnability assessment of PUFs was proposed in [1]. It proves the PAC learnability of Interpose PUF (IPUF), Double Arbiter PUF (DAPUF), Multiplexer PUF (MPUF), Feed-forward PUF (FF-APUF) and Feed-forward XOR PUF (FF-XOR PUF).

In this work, we provide a corrigendum for PAC learnability of MPUF (given in [1]) along with the learnability proof for a new PUF construction, named S_N-PUF [11]. It is to be noted that we use a Fourier based approach for this analysis, as in [3]. We provide a brief background on PUFs and Fourier Analysis of Boolean functions and then proceed to prove the PAC learnability of both designs.

2 Background

This section briefly describes the concept of PUFs, LTF, Low degree algorithm and PAC model.

2.1 Physically Unclonable Functions

In this section, we give a brief description of some of the silicon PUFs, namely Arbiter PUF (APUF), Multiplexer PUF (MUX PUF) and the recently proposed S_N-PUF.

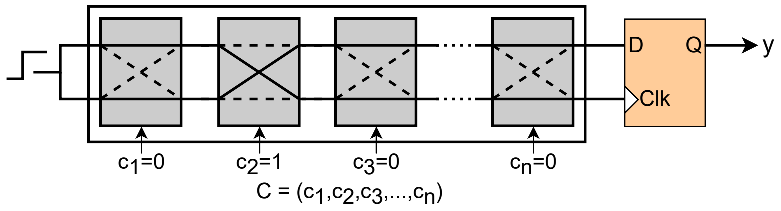

Figure 1: Block diagram of an Arbiter PUF

Arbiter PUF

Arbiter PUF is one of the first PUFs developed and henceforth has been used as a fundamental component is various other PUFs. It consists of a series of multiplexers followed by an arbiter. A signal passing in two identical paths through the multiplexers controlled by the challenge bits, reach the arbiter, which decides the output of the PUF, depending on whether the signal reaches the top or bottom input first. An n-bit APUF, consists of n multiplexers, each of which is controlled by a challenge bit. The schematic of an Arbiter PUF is shown in Figure 1.

The challenge-response behaviour of an APUF can be represented using a linear delay model [5]. Let $c \in \{-1, 1\}^n$ be a challenge and $y \in \{-1, 1\}$ be the response where n be the number of switches in the APUF. Let $c_i \in \{-1, 1\}$ be the i^{th} challenge bit. The total delay difference at the end of $(n)^{th}$ stage can be given by :

$$\Delta(n) = \langle \mathbf{w}, \phi \rangle \tag{1}$$

where $\mathbf{w}$ denotes the weight vector comprising of the propagation delays and ϕ is the parity vector or the transformed challenge vector. The APUF output is given by $y = sign(\Delta(n))$.

Multiplexer PUF (MPUF)

Multiplexer PUF (MPUF) as a variant of APUF, which was proposed with the objective of enhancing modelling resistance without compromising on reliability [10]. It combines $2^k + k$ APUFs (2^k APUFs connected to data lines and k APUFs connected to select lines) using a $2^k \times 1$ MUX as shown in Fig. 2. The $2^k \times 1$ MUX is realized using $(2^k - 1)$ 2×1-MUXes. Each of the APUFs are provided with the same challenge and the output of the MUX is the MPUF response. For a given challenge, the k-bit select input only one of the APUF output gets propagated to the final response. Thus, MPUF response depends on only $k + 1$ APUF outputs, which explains the improved reliability.

S-PUF and S_N-PUF

S-PUF [11] is an arbiter based PUF construction consisting of two n-stage APUFs, whose outputs are combined using an XOR gate. The input $C = \{c_1, c_2, \cdots, c_n\}$ to the first APUF is shifted by $\frac{n}{2}$ bits and fed to the second APUF. Thus the input to the second APUF is $\widetilde{C} = \{c_{\frac{n}{2}}, c_{\frac{n}{2}+1}, \cdots, c_n,\ c_1, \cdots, c_{\frac{n}{2}-1}\}$. The schematic of S-PUF with two arbiters is given in Figure 3. The rationale behind this construction is to reduce the response bias, thereby improving the SAC property.

S_N-PUF [11] is a composition of N S-PUFs, whose outputs $o_1, o_2, \cdots, o_N$ are combined using a Maiorana-McFarland (M-M) Bent [7] function to produce a 1-bit response. The schematic

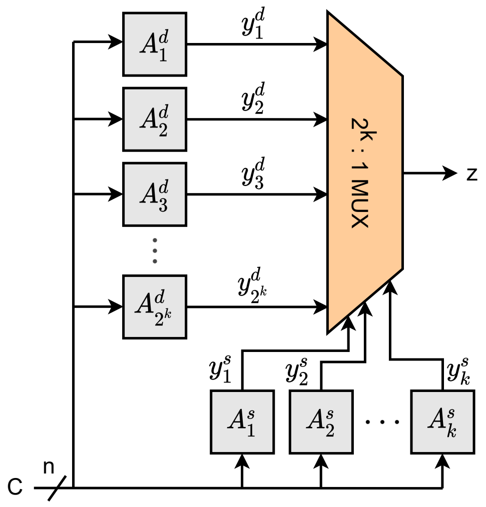

Figure 2: Block diagram of Multiplexer PUF (MPUF)

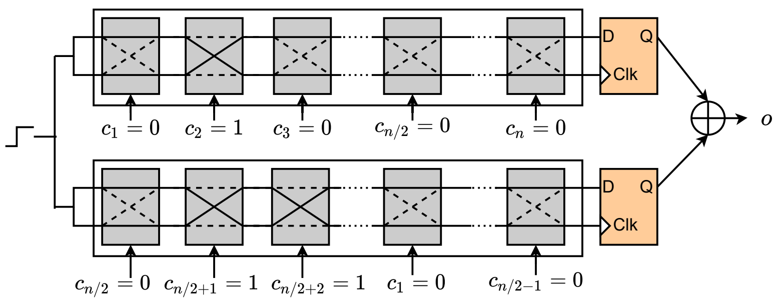

Figure 3: Block diagram of S-PUF with two APUFs

of S_N-PUF is given in Figure 4. Maiorana-McFarland (M-M) Bent function is given by,

$$\begin{aligned} f &: \mathbb{F}_2^{\frac{N}{2}} \times \mathbb{F}_2^{\frac{N}{2}} \to \mathbb{F}_2 \\ f(x,y) &= x.\pi(y) + g(y), \quad \text{for all } (x,y) \in \mathbb{F}_2^{\frac{N}{2}} \times \mathbb{F}_2^{\frac{N}{2}} \end{aligned} \tag{2}$$

where $x = (o_1, o_2, \cdots, o_{\frac{N}{2}})$ and $y = (o_{\frac{n}{2}+1}, o_{\frac{N}{2}+2}, \cdots, o_N)$ and $\pi(y)$ is a permutation function and g is a Bent function. Let $z = (z_1, z_2, \cdots, z_{\frac{N}{2}})$ be the $\frac{N}{2}$-bit product where $z_i = x_i y_{\pi(i)}$. In [11], the authors have considered $g(y) = 0$ for better reliability. The rationale behind this construction is to boost modelling robustness, without hampering the reliability.

2.2 Fourier Analysis of Boolean Functions

A Linear Threshold Function $h : \mathbb{R}^n \to \{0,1\}$ is given by:

$$h = \begin{cases} 1, & \text{if } \sum_{i=1}^{n}(w[i].\phi[i]) \geq \theta \\ 0, & \text{if } \sum_{i=1}^{n}(w[i].\phi[i]) < \theta \end{cases} \tag{3}$$

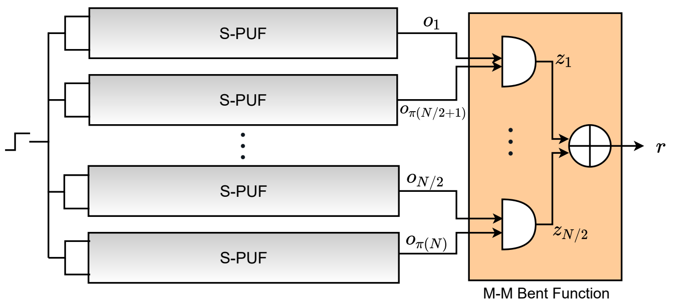

Figure 4: Block diagram of S_N-PUF where N is the number of constituent S-PUFs

where $\phi \in \mathbb{R}^n$ is the input vector and $\mathbf{w}$ represents the weight vector. The sets of positive and negative examples of h form two halfspaces S^0 and S^1 where $S^1 = \{\phi \in \mathbb{R}^n | \sum_{i=1}^{n}(w[i].\phi[i]) \geq \theta\}$ and $S^0 = \{\phi \in \mathbb{R}^n | \sum_{i=1}^{n}(w[i].\phi[i]) < \theta\}$. Mapping $\{0,1\} \rightarrow \{1,-1\}$, including the constant in the weight vector $\mathbf{w}$ and appending 1 to the input vector ϕ, we get $h = sign(\mathbf{w}.\phi)$ where $\mathbf{w} = (w_1, w_2, \cdots, w_n, \theta)$ and $\phi = (\phi[1], \cdots, \phi[n], 1)$. Decision hyperplane is given by $\mathcal{P} : \mathbf{w}.\phi = 0$

Noise sensitivity of Boolean functions The noise sensitivity of a Boolean function f is given by

$$\mathrm{NS}_\epsilon(f) = \mathrm{Pr}_{x \in X}[f(x) \neq f(x')] \tag{4}$$

where x' is obtained by flipping each bit of x independently with a probability of ϵ.

The Fourier expansion of a Boolean function is given by

$$f(c) = \sum_{S \subset [n]} \hat{f}(S)\chi_S(c) \tag{5}$$

where $\chi_S(c) = \prod_{i \in S} c_i$, $[n] = \{1, 2, \cdots, n\}$ and $\hat{f}(S) = \mathbb{E}_{c \in \mathcal{U}}[f(c)\chi_S(c)]$.

The relationship between noise sensitivity of a function $f : \{+1,-1\}^n \rightarrow \{+1,-1\}$ and its Fourier spectrum is given by [9]

$$NS_\epsilon(f) = \frac{1}{2} - \frac{1}{2} \sum_{S \subset [n]} (1-2\epsilon)^{|S|} f(S)^2 \tag{6}$$

For any function $f : \{+1,-1\}^n \rightarrow \{+1,-1\}$, $0 < \epsilon < \frac{1}{2}$

$$\sum_{|S| \geq \frac{1}{\epsilon}} \hat{f}(S)^2 \leq 2.32 NS_\epsilon(f) \tag{7}$$

Corollary 1: [8] Let $f : \{+1,-1\}^n \rightarrow \{+1,-1\}$ be any Boolean function and let $\alpha : [0, \frac{1}{2}] \rightarrow [0,1]$ be an increasing function such that $NS_\epsilon(f) \leq \alpha(\epsilon)$. Then,

$$\sum_{|S| \geq d} \hat{f}(S)^2 \leq \epsilon \qquad \text{for } d = \frac{1}{\alpha^{-1}(\frac{\epsilon}{2.32})} \tag{8}$$

The following result connects the Fourier spectrum analysis and learning theory.

Fact 1: [8] Let $\mathcal{C}$ be a concept class with a Fourier concentration bound of d. Then there exists a uniform distribution PAC learning algorithm for $\mathcal{C}$ which runs in time $n^{\mathcal{O}(d)}$. The sample complexity of the learning algorithm is $n^{\mathcal{O}(d)} ln(1/\delta)$.
The bound on noise sensitivity of an LTF and any function of constant number of LTFs is given as follows:
Corollary 2: [4] For any halfspace $h : \{+1, -1\}^n \rightarrow \{+1, -1\}$ we have,

$$NS_\epsilon(f) \leq 8.54\sqrt{\epsilon} \tag{9}$$

Theorem 1: [4] Let $f : \{+1, -1\}^k \rightarrow \{+1, -1\}$ be any function of k halfspaces. Then $NS_\epsilon(f) \leq \mathcal{O}(k\sqrt{\epsilon})$.

Applying Corollary 1, we get Fourier concentration bound $d = \mathcal{O}(k^2/\epsilon^2)$ for class of k halfspaces ($\epsilon < 1/k^2$).

2.3 PAC Model

The Probably Approximately Correct (PAC) model of learning is a general model which enables us to formally analyse machine learning algorithms [12]. It consists of a learning algorithm ($\mathcal{A}$), which is provided with a set of examples picked from the input space $\mathcal{X}$ as per distribution $\mathcal{D}$ and labelled using the target function f. The objective of the algorithm is to deliver an approximately correct hypothesis with high probability. It can be formally stated as follows:

Let $\mathcal{C}_n$ be a concept class defined over an instance space $\mathcal{X}_n = \{0, 1\}^n$ and let $\mathcal{X} = \cup_{n \geq 1} \mathcal{X}_n$ and $\mathcal{C} = \cup_{n \geq 1} \mathcal{C}_n$. Let f be the target function in $\mathcal{C}_n$. Let $\mathcal{H}_n$ be the hypothesis class and $\mathcal{H} = \cup_{n \geq 1} \mathcal{H}_n$. The concept class $\mathcal{C}_n$ is said to be PAC Learnable if there exists a learning algorithm $\mathcal{A}$, polynomial $p(\cdot, \cdot, \cdot)$ and values ϵ and δ with the following property: For every $\epsilon, \delta \in (0, 1)^2$, for every distribution $\mathcal{D}$ over $\mathcal{X}_n$ and every target concept $f \in \mathcal{C}_n$, when $\mathcal{A}$ is provided with $p(n, 1/\epsilon, 1/\delta)$ independent examples drawn with respect to $\mathcal{D}$ and labelled according to f, then with probability atleast $1 - \delta$ the algorithm $\mathcal{A}$ returns a hypothesis $h \in \mathcal{H}_n$ such that $error(h) \leq \epsilon$. The smallest polynomial p satisfying this condition is the sample complexity of $\mathcal{A}$. If the above condition holds only for uniform distribution over $\mathcal{X}$, A is known as a *uniform distribution* PAC Learning algorithm.

3 PAC Learning of PUFs using Fourier Approach

In this work, we prove that MPUF and S_N-PUF are learnable in the PAC model under uniform input distribution. To demonstrate this, we use the low degree algorithm proposed in [8] that generates a close approximation of a Boolean function given that the Fourier distribution of the function is concentrated to a small number of terms. The learning algorithm can approximate the function by focusing on only a small fraction of the Fourier coefficients. We leverage this algorithm and the relationship between noise sensitivity and Fourier spectrum of Boolean functions to provide a polynomial time learning algorithm for the above mentioned PUF designs. This attack methodology was first used in the PUF literature to learn APUF, ROPUF and BR-PUF constructions [3].

First, we present the Low Degree algorithm that generates a hypothesis function h which is ϵ-close to the function f after observing polynomial number of labelled examples.

Theorem 2: Low Degree Algorithm (LMN algorithm) [3, 8] Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function. Given a set $\mathcal{S} \subseteq 2^{[n]}$ consisting of subsets of $[n]$ such that $\sum_{s \in \mathcal{S}} \hat{f}(S)^2 \geq 1 - \epsilon$, a confidence level δ and access to a polynomial number of input output pairs of f chosen uniformly at random, the low degree algorithm delivers a Boolean function h with a probability $1-\delta$ that is an ϵ-approximator of f,

$$\sum_{S\subset[n]} \left(\hat{f}(S) - \hat{h}(S)\right)^2 \leq \epsilon$$

For more details on the algorithm, its sample complexity and proof of the theorem, we refer the readers to [6, 8].

3.1 PAC Learning of Multiplexer PUF (MPUF)

An (n,k)-MPUF comprises of 2^k n-stage APUFs whose outputs are connected to data lines and k n-stage APUFs whose outputs are connected to select lines as shown in Fig. 2. The output of the MUX network is the final response. LTF is a well known representation for APUF, owing to its linear delay model [5]. It has a bounded noise sensitivity and therefore can be PAC learned by the Low Degree algorithm. Here, we prove that a composition of APUFs using a MUX network is also PAC Learnable by virtue of the noise sensitivity bound given in [4]. Formally, we prove the following:

Theorem 3: For an (n,k)-MPUF represented by a function g, the Low degree algorithm can deliver a Boolean function h approximating g with a probability of $1-\delta$. The running time of the algorithm is $poly(n, 2^{2k}/\epsilon^2, log_2(1/\delta))$.

Proof: An APUF can be represented using a LTF, owing to its linear delay model [5]. The noise sensitivity of an LTF f is bounded by $NS_\epsilon(f) \leq 8.54\sqrt{\epsilon}$ [4].

An (n,k)-MPUF comprises of $2^k + k$ APUFs whose outputs are combined using a MUX network. For a given k, a MPUF can be considered as a function of constant number of LTFs (2^k). Thus, noise sensitivity of (n,k)-MPUF is bounded by $\mathcal{O}(2^k\sqrt{\epsilon})$ using Theorem 1. Using Corollary 1, we obtain that the running time of the low degree algorithm is polynomial in $\mathcal{O}(n^d)$ where $d = 1/\alpha^{-1}(\epsilon/2.32) = (2.32 \times 2^k)^2/\epsilon^2$. The sample complexity of the learning algorithm is $O(n^d ln(1/\delta))$. □

3.2 PAC Learning of S_N-PUF

In this section, we derive the learnability bounds of S_N-PUF. We show that the construction can be accurately modelled with a high probability using the Low degree algorithm. Formally, we prove the following:

Theorem 4: The low degree algorithm can deliver a Boolean function h approximating a function representing an S_N-PUF with a probability of $1-\delta$. The running time of the algorithm is $poly(n, N^2/\epsilon^2, log_2(1/\delta))$.

Proof: An S-PUF can be represented using a LTF, owing to its linear delay model [11]. Consequently, its noise sensitivity is bounded by $NS_\epsilon(f) \leq 8.54\sqrt{\epsilon}$ [4].

The outputs of N S-PUFs are combined using an MM Bent Function. Thus, an S_N-PUF can be considered as a function of N LTFs. The noise sensitivity of S_N-PUF is bounded by $\mathcal{O}(N\sqrt{\epsilon})$ (using Theorem 1). Using Corollary 1, we obtain that the running time of the low degree algorithm is polynomial in $\mathcal{O}(n^d)$ where $d = 1/\alpha^{-1}(\epsilon/2.32) = (2.32 \times N)^2/\epsilon^2$. The sample complexity of the learning algorithm is $O(n^d ln(1/\delta))$. □

An important point to note is that the Low degree algorithm can learn any function of k LTFs only when $k = O(1)$. For $k > \sqrt{ln\, n}$, the learning algorithm fails. This also provides a practical bound on the size of the PUF compositions.

4 Conclusion

We proved that Multiplexer PUF and S_N-PUF are learnable in the PAC model for various values of accuracy and confidence. To this end, we showed that the noise sensitivity of any function combining the outputs of a constant number of LTFs can be bounded by a constant factor, and consequently the construction can be accurately approximated by the Low degree algorithm. Finally we proved that the sample complexity of the learning algorithm is polynomial in the number of stages (n), Fourier coefficient bound (d) and the PAC model parameters.

References

[1] Durba Chatterjee, Debdeep Mukhopadhyay, and Aritra Hazra. PUF-G: A CAD framework for automated assessment of provable learnability from formal PUF representations. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

[2] Fatemeh Ganji. *On the Learnability of Physically Unclonable Functions.* Springer, 2018.

[3] Fatemeh Ganji, Shahin Tajik, and Jean-Pierre Seifert. A fourier analysis based attack against physically unclonable functions. In *International Conference on Financial Cryptography and Data Security*, pages 310–328. Springer, 2018.

[4] Adam R Klivans, Ryan O'Donnell, and Rocco A Servedio. Learning intersections and thresholds of halfspaces. *Journal of Computer and System Sciences*, 68(4):808–840, 2004.

[5] Daihyun Lim. Extracting secret keys from integrated circuits. 2005.

[6] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993.

[7] James A Maiorana. A class of bent functions. *R41 Technical paper*, 1971.

[8] Yishay Mansour. Learning boolean functions via the fourier transform. In *Theoretical advances in neural computation and learning*, pages 391–424. Springer, 1994.

[9] Ryan O'Donnell. Hardness amplification within np. *Journal of Computer and System Sciences*, 69(1):68–94, 2004.

[10] Durga Prasad Sahoo, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, and Phuong Ha Nguyen. A multiplexer-based arbiter puf composition with enhanced reliability and security. *IEEE Transactions on Computers*, 67(3):403–417, 2017.

[11] Akhilesh Anilkumar Siddhanti, Srinivasu Bodapati, Anupam Chattopadhyay, Subhamoy Maitra, Dibyendu Roy, and Pantelimon Stănică. Analysis of the strict avalanche criterion in variants of arbiter-based physically unclonable functions. In *International Conference on Cryptology in India*, pages 556–577. Springer, 2019.

[12] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.