Strengthening Physical Unclonable Functions using Composition

by

Zhuanhao Wu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

Author's Declaration

Abstract

We explore the idea of composing PUFs with the intent that the resultant PUF is stronger than the constituent PUFs. Prior work has proposed a construction, which subsequent work has shown to be weak. We revisit this prior construction and observe that it is actually weaker than previously thought when the constituent PUFs are arbiter PUFs. This weakness is demonstrated via our adaptation of the previously proposed Logistic Regression (LR) attack. We then propose new constructions called PUFs-composed-with-PUFs ($P \circ P$). In particular, we retain a two-layer construction, but allow the same input to the composite PUF to be input to more than one constituent PUF at the first layer. We explore this family of constructions, with arbiter PUFs serving as the constituent PUFs. In particular, we identify several axes which we can vary, and empirically study the resilience of our constructions compared to the prior construction and one another from the standpoint of LR attacks. As insight in to why our family of constructions is stronger, we prove, under some idealized conditions, that the lower-bound on an attacker is indeed higher under our constructions than the upper-bound on an attacker for the prior construction. As such, our work suggests that composition can be a promising approach to strengthening PUFs, contrary to what prior work suggests.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor Professor Hiren Patel for his patience, guidance, and training over the past two years. The door to the office of Prof. Patel was always open whenever I ran into a trouble spot or had a question about my research.

I would also like to thank my reviewers, Professor Mahesh Tripunitara, and Professor Manoj Sachdev. Without their passionate participation and kind support, this work could not have been carried out.

I thank my colleagues and friends Anirudh M. Kaushik, Paulos Tegegn in the Computer Architecture and Embedded Systems Research group at University of Waterloo for the guidance, support, and expert advice.

I thank Yang Shen for her kind encouragement and company.

Finally, I thank my family for the unfailing support and encouragement throughout my years of study.

Dedication

For my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the advent of Internet-of-Things (IoT) revolution, the number of distributed and unsupervised mobile computing devices continues to increase, and experts believe there will be approximately 100 billion connected devices by 2020 [8]. For such wide ranging devices, authentication for counterfeit prevention and secure communication is an important consideration.

1.1 Physical Unclonable Functions

A Physical Unclonable Function (PUF) is a physical one-way function where the mapping takes a constant time given the input. However, reverse-engineering the behavior of such a function given the output is supposed to be computationally difficult. In [4], different types of PUFs have been implemented using FPGAs. In [18], different types of PUFs are implemented and evaluated using both FPGAs and 45nm SOI CMOS ASICs. The silicon-based PUF leverages the process variation during the manufacturing process. Such process variation uniquely characterizes each of the chips, which appear physically identical at design time. PUFs have recently been proposed as replacements for non-volatile memories and on-die fuses that are prone to physical attacks for storing chip identifying digital signatures and seed generators to other cryptographic functions [13, 14].

Each of the PUF, $f : C \to R$ can be regarded as a black-box function. The black-box function f maps input from C, called *challenge*, to output in R, called *response*. The black-box might accept one or more challenges and produce corresponding responses, which form a set of *challenge-response pairs* (CRPs). The set of CRPs is also called the CRP space. PUFs can be characterized as strong or weak. The fundamental differences lie in the number of possible challenges the PUF supports. A strong PUF supports a large number of challenges so that enumerating all of the CRPs is hard within a limited time. The weak PUF, on the other hand, may only support a limited number of CRPs and can thus be used as a secret key without explicit storage [7].

PUF should bear a number of properties. This thesis focus on some of these properties which are listed as follows [11]:

Properties of PUFs

- Reproducibility: the PUF should ouput the same response for the same challenge.
- Unpredictability: the unobserved responses should be sufficiently random even after one has observed limited number of CRPs.
- Unclonability: given unlimited access to the PUF, an adversary cannot predict the response of an unobserved challenge with high probability.

Over the years, several strong PUF architectures have been proposed [7]. However, most of these PUFs have also been shown to be susceptible to modeling attacks. Through modeling attacks, an adversary can mimic the behavior of the strong PUF with a high prediction accuracy (around 95% or higher) rendering them ineffective [18, 23]. An interesting approach proposed in [20] was to compose PUFs such that the resultant PUF offered improved security, which they called composite PUF (CPUF). The central thesis underlying the approach was that compositions allowed increasing the CRP space while also preserving the performance properties of the resultant PUF. In a later work, the authors themselves identified that the CPUF was also susceptible to a two-phase modeling attack [19] called the cryptanalysis attack (CA-ATK). They showed that CA-ATK, although successful in modeling CPUF, required an enumeration of a large CRP to conduct the attack. This thesis aims to answer the following question: is the composition of PUFs still a good way of designing PUFs?

1.2 Contributions

The contributions of this thesis are as follows:

- Demonstration of the susceptibility of CPUF composed using ARB-PUFs to an enhancement of CA-ATK, which is a previously investigated attack on CPUF. The enhanced attack is called LR-CA-ATK.
- Demonstration of the effectiveness of LR-CA-ATK on CPUF composed of LWS-PUF and XOR-PUF.
- A simulation framework for investigating modeling attacks on PUF.
- A theory that characterizes an attacker's effort needed to fully charactrize a PUF.
- Emprical evaluation demonstrating the effectiveness of incoporating mapping functions in CPUF against machine learning modeling attacks.

Chapter 2

Background

For a reader to comprehend this work, this chapter provides necessary background information including logistic regression, evolutionary strategies, PUF architectures and modeling attacks.

2.1 Logistic Regression

With the rapid development of computing resources, we are in an age of data. Machine learning is a set of method that provides an automated way of analyze data. Specifically, With machine learning methods, a computing machine can *learn* the patterns in the data and later predict future data or make decisions based on learned patterns [15]. One type of machine learning method is the supervised learning. In supervised learning requires a *training set*, $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}$, of pairs of input $\mathbf{x}_i$ and output $\mathbf{y}_i$. This approach aims to learn the mapping between the input and output so that the prediction on the output of unseen input data is possible. The evaluation of the of the learning outcome is determined by the prediction accuracy on a *test set* $T = \{(\mathbf{x}_i, \mathbf{y}_i)\}$.

Logistic regression (LR) is a well studied supervised learning method for classification problem. In its simplest form of a binary classification problem with 2 classes $\{-1, 1\}$, the method aims to learn the probability distribution of the output conditioned on the input as described in Equation 2.1, where X is a random variable denoting the input $\mathbf{x} \in \mathbb{R}^n$ and Y is a random variable denoting the output $y \in \{-1, 1\}$. The model has one parameter to learn, namely $\mathbf{w} \in \mathbb{R}^n$. For conciseness, we assume that the last component of vector $\mathbf{x}$

satisfies $\mathbf{x}(n) \equiv 1$.

$$\ln \frac{Pr(Y = 1|X = \mathbf{x})}{Pr(Y = -1|X = \mathbf{x})} = \mathbf{w}^T\mathbf{x}. \tag{2.1}$$

Equation 2.1 is equivalent to another famous form as described in Equation 2.2:

$$Pr(Y = y|X = \mathbf{x}) = \sigma(y\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-(y\mathbf{w}^T\mathbf{x})}}. \tag{2.2}$$

In Equation 2.2, $\sigma : \mathbb{R} \to [0, 1]$ is the S-shaped sigmoid function, which evaluates to 0.5 when $x = 0$.

The model in Equation 2.1 can be fit by maximum likelihood estimation (MLE). Given a training set $S = \{(\mathbf{x}_i, y_i)\}$, the MLE problem is equivalent to the optimization problem in Equation 2.3

$$\begin{aligned}
\mathbf{w} &= \operatorname*{argmin}_{\mathbf{w}} L(S; \mathbf{w}) && (2.3)\\
&= \operatorname*{argmin}_{\mathbf{w}} \sum_{(\mathbf{x}_i, y_i) \in S} - \ln Pr(Y = y_i|X = \mathbf{x}_i) && (2.4)\\
&= \operatorname*{argmin}_{\mathbf{w}} \sum_{(\mathbf{x}_i, y_i) \in S} - \ln \sigma(y_i\mathbf{w}^T\mathbf{x}_i) && (2.5)
\end{aligned}$$

Equation 2.3 can be solved iteratively by adopting the gradient information of the log-likelihood in Equation 2.6:

$$\nabla L(S; \mathbf{w}) = \sum_{(\mathbf{x}_i, y_i) \in S} y_i(\sigma(y_i\mathbf{w}^T\mathbf{x}_i) - 1)\nabla \mathbf{w}^T\mathbf{x}_i. \tag{2.6}$$

The application of LR successfully breaks the security of serveral types of PUFs.

2.2 Evolutionary Strategies

Evolutionary strategy (ES) [3] is an optimization technique whose inspiration is from biology. The ES aims to optimize an objective function F, which takes as input a set of parameters $\mathbf{w}$. One instance of the objective function and parameter configurations can be: $\mathbf{w} \in \mathbb{R}^n$ and $F : \mathbb{R}^n \to \mathbb{R}$, which will be used through out the thesis. The process of ES introduces a set called *population* denoted as P, whose elements are called *individuals*. An individual $a_k \in P$ is a tuple: $a_k = \langle \mathbf{w}_k, \mathbf{s}, F(\mathbf{w}_k)\rangle$, in which:

- $\mathbf{w}_k$ is the parameter of a_k,
- $\mathbf{s}_k$ is the parameter that determines the mutation of a_k, and
- $F(\mathbf{w}_k)$ is the value of the objective function of a_k, called *fitness* in the context of an individual.

The ES goes through evolutionary cycles, or *generation step*. Within each of the generation step g, λ offspring individuals are generated from a set of μ individuals, called *parents*, from the population of last generation step, P_{g-1}. In the process of producing offspring individuals, μ of the individuals with the best fitness is selected, each is copied $\frac{\lambda}{\mu}$ times to form the new population P_g. Such selection strategy is denoted as (μ, λ). After that, a mutation operator will mutate each individual $a_k \in P_{g+1}$ based on $\mathbf{s}_k$ and the process proceeds to generation $g+1$. As the generation step grows, the individuals will have better fitness and thus moving to the optimal point of the objective function.

Since ES only requires a parametric model, it has been adopted to break the security of serveral types of PUFs, which are hard to break with LR.

2.3 Physical Unclonable Functions

Weak PUFs are often used for secret key generation where the CRP requirement is limited. On the other hand, strong PUFs are used for authentication purposes. As a security measure, each CRP is used once only. Consequently, when a PUF supports a large number of CRPs, i.e., is strong, adversaries cannot ascertain them under a constrained time frame [7]. Of course, the underlying function that the PUF realizes must be a random function, or some close approximation of it, for the PUF to be strong. Otherwise, even if the CRP space appears large, the PUF cannot be said to be strong, as it can be characterized fully with fewer CRPs than the size of its domain suggests.

2.3.1 Strong PUF Architectures

Arbiter PUF (ARB-PUF)

Figure 2.1 shows the ARB-PUF [7], which is one of the most investigated strong PUFs. ARB-PUF has two identical delay paths that race from left to right through n stages of multiplexers that are driven by the challenge bit. If the Data (D) arrives faster than the

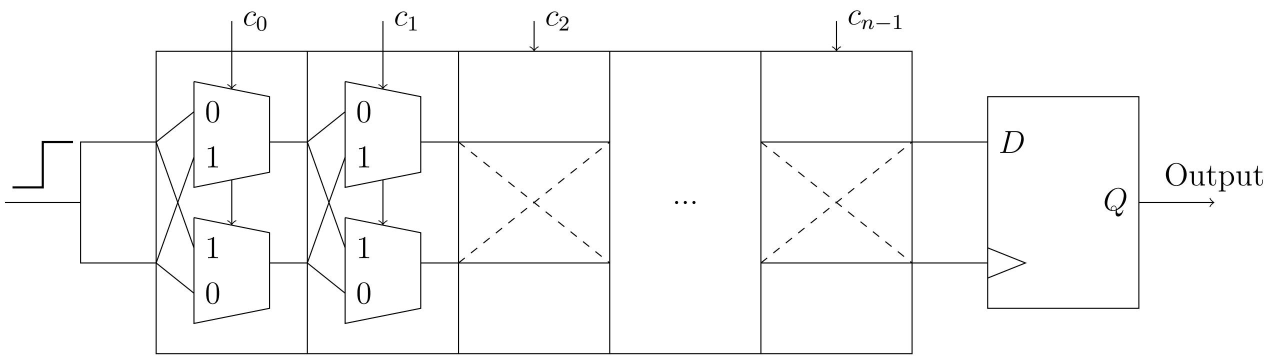


Figure 2.1: In the ARB-PUF architecture, each bit in the challenge controls one of two group of paths the rising edge goes through [7].

closing edge of the clock, the output (Y) makes a positive transition. An n bit challenge directs these paths through n multiplexers. The latch at the output acts as an arbiter selecting the edge arriving early [7]. The n bit challenges result in 2^n unique path pairs, which is also known as the architecture's CRP space. Throughout the thesis, ARB-PUF(n) represents an ARB-PUF with n-bit challenge.

XOR arbiter PUF (XOR-PUF)

As a variant of ARB-PUF, XOR-PUF deploys l ARB-PUF(n) in parallel as shown in Figure 2.2. The same challenge is applied to all ARB-PUFs, and their output is XORed to produce a one bit response [22]. Throughout the thesis, XOR-PUF(n, l) represents an XOR-PUF with n-bit challenge and l chains.

Lightweight Secure PUF (LWS-PUF)

Similarly, the light-weight secure PUF (LWS-PUF) [12] was introduced to make it difficult for an attacker to model its behavior. A light-weight secure PUF also features l ARB-PUF(n) as shown in Figure 2.3. However, when a challenge is applied, the challenge will go through an input network. The input network produces different challenges for each of the ARB-PUF to consume. Finally, the output of each ARB-PUF is XORed to produce the output of a LWS-PUF. Throughout the thesis, LWS-PUF(n, l) represents a LWS-PUF with n-bit challenge and l chains.

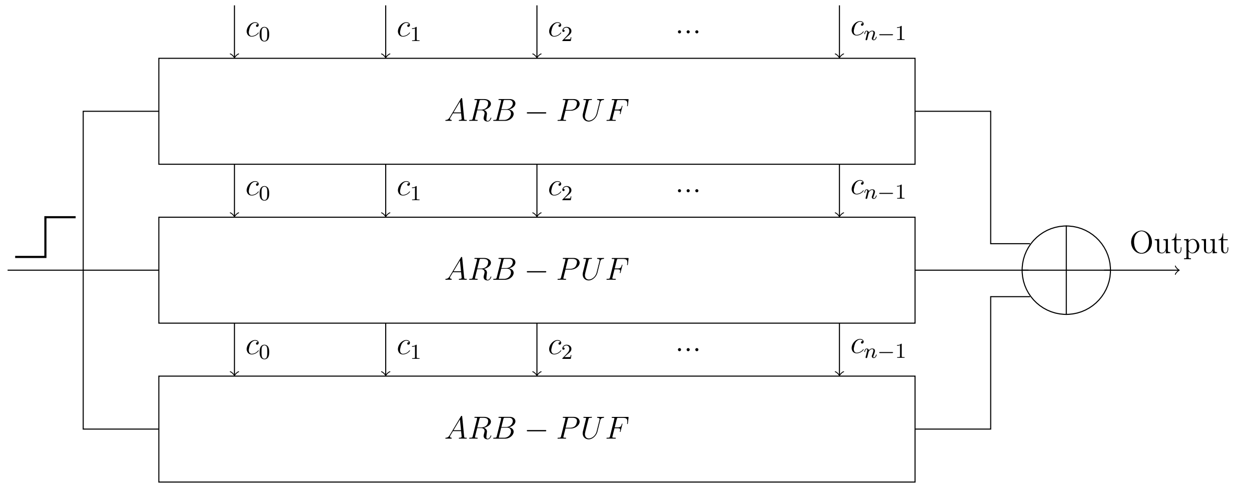


Figure 2.2: The XOR-PUF has 3 ARB-PUFs, accepting same challenge.

Feed-forward PUF (FF-PUF)

In feed-forward PUF (FF-PUF) [5], differential output of a multiplexer are fed into another arbiter which controls a subsequent multiplexer as shown in Figure 2.4.

Throughout the thesis, FF-PUF(n, l) represents a LWS-PUF with n-bit challenge and l loops.

Composite PUF (CPUF)

Composite PUF (CPUF) [20] presented an approach to design strong PUFs using composition of PUFs. An important observation in their work was that PUF compositions increase the CRP space while preserving important performance properties [20]. Their work focused on a two-layer composition where the outputs of the first layer are fed as challenge inputs to the second layer PUF [19]. While their compositions were successful in increasing the CRP space, the proposers of this approach themselves identified a crypt-analysis attack (CA-ATK) that successfully modeled the composite PUF [19]. However, CA-ATK required an enumeration of the entire CRP space to be successful. Throughout the thesis, CPUF(n, m) represents a CPUF, where Layer 1 PUFs take as input m-bit challenges and the Layer 2 PUF takes as input n-bit challenges.

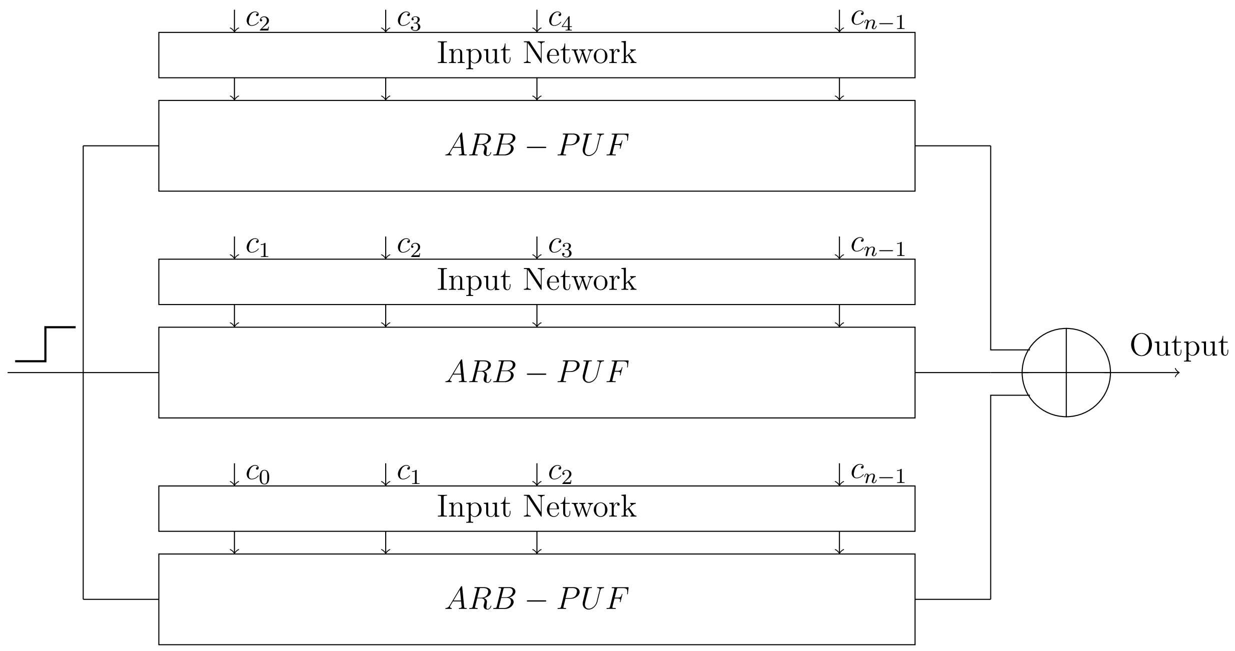


Figure 2.3: The LWS-PUF has 3 ARB-PUFs, accepting different challenges.

Multi-PUF (MPUF)

The multi-PUF (MPUF) [10] is one instantiation of CPUF of 2 layers. In its first layer, the MPUF uses a weak PUF called PicoPUF [6] to generate a key and XORes it with a challenge bit. The XORed bits are fed into the Layer 2 PUF, which is an ARB-PUF. And the output of the ARB-PUF serves as the output of the MPUF. Figure 2.6 shows the architecture of MPUF.

2.4 Modeling Attacks on PUF

A modeling attack on a PUF begins with an attacker collecting a subset of CRPs of that PUF. The attacker uses these CRPs to derive a model that predicts the responses of the PUF given any challenge. The model's effectiveness depends on the likelihood of producing a correct response for a challenge.

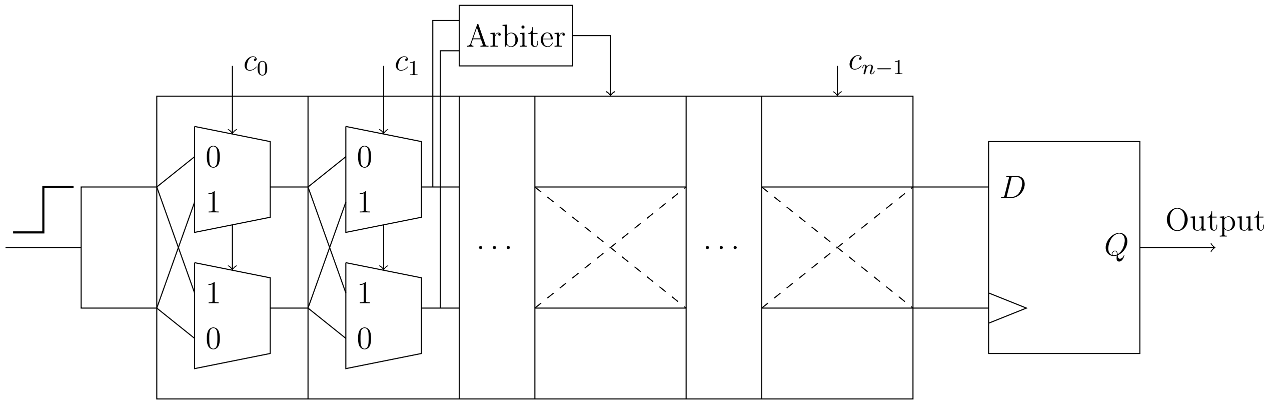


Figure 2.4: The FF-PUF employs an FF-loop to generate the control signal for a subsequent multiplexer group.

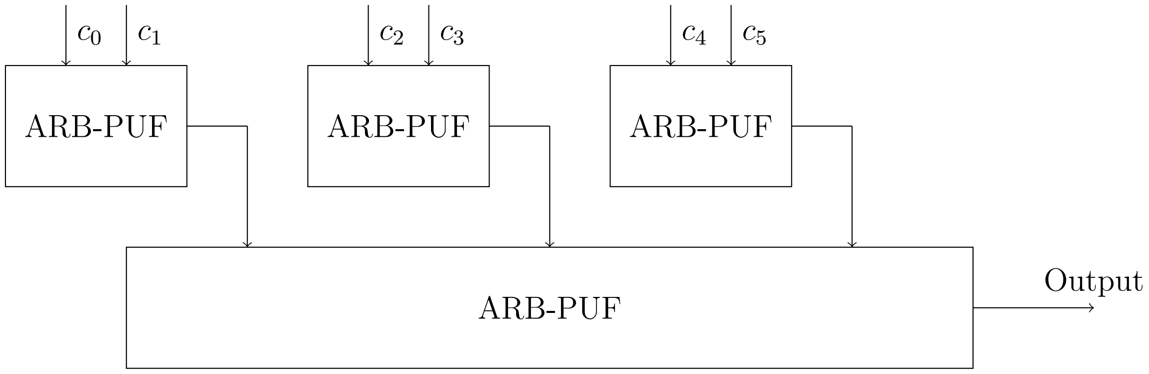


Figure 2.5: A CPUF$(3, 2)$ has 3 Layer 1 PUFs and 1 Layr 2 PUFs.

2.4.1 Machine Learning Modeling Attacks

ML algorithms are powerful tools which are naturally suitable for deriving such a model. Reference [17] introduces a successful way of modeling ARB-PUF, XOR-PUF with 1 output bit, LWS-PUF with 1 output bit and FF-PUF using ML methods.

Linear Additive Model (LAM)

The cornerstone of the ML modeling attacks is the LAM [9], which models the delay values of the ARB-PUF. In this section, we introduce the LAM for ARB-PUF(n). In the LAM,

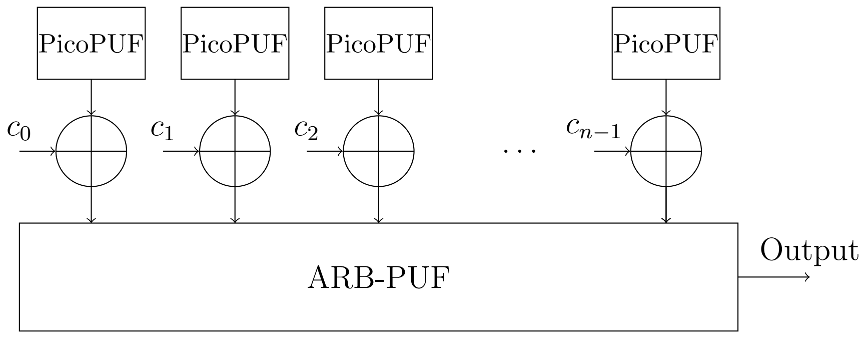


Figure 2.6: The MPUF architecture XORes the challenge with a secret key provided by a sequence of PicoPUFs and provides the XOR result as the input to the ARB-PUF [10].

the difference of delays of two racing paths, Δ, is modeled in Equation 2.7.

$$\Delta = \mathbf{w}^T \Phi, \tag{2.7}$$

where $\mathbf{w} \in \mathbb{R}^{n+1}$ are the parameters that encode delays and Φ is the encoded challenge defined in 2.8.

$$\Phi_i = F(\mathbf{c})(i) = \begin{cases} (-1)^{c_i}\Phi_{i+1} = \prod_{j=i}^{n}(-1)^{c_j}, & \text{if } 0 \le i \le n-1 \\ 1, & \text{if } i = n \end{cases}. \tag{2.8}$$

The response of the ARB-PUF is thus $t = sgn(\Delta)$, where $sgn(x)$ is sign function and returns 1 if x is non-negative and -1 otherwise.

The LAM can be trained with a training set composed of CRPs of the target ARB-PUF using LR described in Section 2.1.

Multiplicative Model

For XOR-PUF and LWS-PUF, which have more than one ARB-PUFs, it is possible to combine the LAMs of each of the ARB-PUFs, resulting in the multiplicative model described in Equation 2.9[17].

$$t_{XOR} = \prod_{i=0}^{l-1} sgn(\Delta_i) = \prod_{i=0}^{l-1} sgn(\mathbf{w}_i^T \Phi_i) = sgn\left(\prod_{i=0}^{l-1} \mathbf{w}_i^T \Phi_i\right). \tag{2.9}$$

In Equation 2.9, Φ_i represents the challenges defined in 2.8 for each of the ARB-PUF and $\mathbf{w}_i$ represents the encoded delays of the i-th ARB-PUF. Note that the last product term no longer represents the delay. The multiplicative model can be trained using LR in 2.1.

Evolutionary Strategies

ES successfully model FF-PUF, which is hard to model using LAM and multiplicative model. When modeling FF-PUFs, the ES trains an extended version of LAM. Equation 2.10 demonstrates the extended version of LAM for FF-PUF$(n, 1)$, which counts the difference of delays for the output arbiter. The loop starts at the end of stage i_1 and the output is forwarded to stage i_2.

$$\Delta = \mathbf{w}^T\Phi, \tag{2.10}$$

where Φ is defined as

$$\Phi_i = F(\mathbf{d})(i) = \begin{cases} (-1)^{d_i}\Phi_{i+1} = \prod_{j=i}^{n+1}(-1)^{d_j}, & \text{if } 0 \le i \le n \\ 1, & \text{if } i = n+1 \end{cases}$$

and

$$d_i = \begin{cases} c_i, & \text{if } 0 \le i < i_2 \\ f, & \text{if } i = i_2 \\ c_{i-1}, & \text{if } i > i_2 \end{cases} .$$

f is the output of the feedforward arbiter, defined as:

$$t_{ff} = \frac{1 - sgn(\mathbf{v}^T\Psi)}{2} \tag{2.11}$$

, where

$$\Psi_i = K(\mathbf{c})(i) = \begin{cases} (-1)^{c_i}\Psi_{i+1} = \prod_{j=i}^{i+1}(-1)^{c_j}, & \text{if } 0 \le i < i_1 \\ 1, & \text{if } i = i_1 \end{cases}$$

and

$$v_i = w_i, 0 \le i < i_1.$$

v_{i_1} is another free parameter.

It is worth mentioning that ES is also able to model ARB-PUF, XOR-PUF and LWS-PUF, with the numerical models in previous sections.

2.4.2 Cryptanalysis Attacks

In Reference [19], a cryptanalysis attack, CA-ATK, successfully models the 2 layer CPUF. The CA-ATK is composed of two phases and each phase targets a different layer.

Algorithm 1: Set-Construction(K, P, C_i)

```
Input  : Repeat number $K$, CPUF $P$ and the challenge space $C_i$ of $P_i$
Output: Set $S_{i,0}$ and $S_{i,1}$, the partition of $C_i$
1 $S_{i,0} = \{\mathbf{a}\}, \mathbf{a} \in C_i$ is arbitrarily chosen;
2 $S_{i,1} = \emptyset$;
3 Choose a set of $K$ random different challenge of $P$: $X = \{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^K\}$;
4 for $\mathbf{b} \in C_i$ do
5     $s = 0$;
6     for $\mathbf{x} \in \mathbf{X}$ do
7         replace the $\mathbf{x}_i$ with $\mathbf{a}$ and $\mathbf{b}$ to obtain $\mathbf{u}$ and $\mathbf{v}$;
8         $s = 1$ if $P(\mathbf{u}) \neq P(\mathbf{v})$;
9     $S_{i,s} = S_{i,s} \cup \{\mathbf{b}\}$;
```

Phase 1

In the first phase, for each Layer 1 PUF p_i, the attacker partitions the challenge space $C_i = \{0,1\}^m$ of p_i into two sets $S_{i,0}$ and $S_{i,1}$. Let us assume that $\mathbf{a}$ and $\mathbf{b}$ are two challenges in C_i. If $\mathbf{a} \in S_{i,0}$ and $\mathbf{b} \in S_{i,1}$ then $p_i(\mathbf{a}) \neq p_i(\mathbf{b})$. This means that the two challenges produce different outputs for p_i. Similarly, if $\mathbf{a}, \mathbf{b} \in S_{i,j}, j \in \{0,1\}$, then $p_i(\mathbf{a}) = p_i(\mathbf{b})$ with high likelihood, which means the two challenges are likely to produce the same output for p_i. To construct the sets $S_{i,0}$ and $S_{i,1}$, one selects two m-bit challenges from C_i, and extends them to mn-bit challenges $\mathbf{x}$ and $\mathbf{y}$ by simply keeping the extended bits the same in both extended versions. If $P(\mathbf{x}) \neq P(\mathbf{y})$, then the attacker can be sure that the corresponding m-bit challenges are not in the same set. If $P(\mathbf{x}) = P(\mathbf{y})$, then the attacker can not be sure that they are in the same set. However, one can repeat this procedure multiple times by changing the extended bits.

Algorithm 1 shows this procedure.

As an example, consider a PUF P as a CPUF$(3,2)$, which takes as input 6-bit challenges. Table 2.1 lists the ground truth responses for each of the constituent PUFs in the CPUF$(3,2)$. Note that the *Response* column is marked as gray as the attacker cannot directly observe this information.

In Phase 1, the attacker may choose $K = 3$ and obtain a set of random challenges of

$$X = \{111001, 110110, 100011\} \tag{2.12}$$

To construct the set for p_0: $S_{0,0}$ and $S_{0,1}$, the attacker replaces the first two bits of each challenge in X with all possible 2-bit strings in $C_0 = \{00, 01, 10, 11\}$, that is:

$$X_{00} = \{111\mathbf{00}, 110\mathbf{00}, 100\mathbf{00}\} \tag{2.13}$$
$$X_{01} = \{111\mathbf{01}, 110\mathbf{01}, 100\mathbf{01}\} \tag{2.14}$$
$$X_{10} = \{111\mathbf{10}, 110\mathbf{10}, 100\mathbf{10}\} \tag{2.15}$$
$$X_{11} = \{111\mathbf{11}, 110\mathbf{11}, 100\mathbf{11}\} \tag{2.16}$$

Then, to determine whether p_0 responses the same for two challenges $\mathbf{a}, \mathbf{b} \in C_0$, the attacker applies challenges in $X_\mathbf{a}$ and $X_\mathbf{b}$, whose only difference is the first two bits, and check the result of the PUF P. For example, if $\mathbf{a} = 00, \mathbf{b} = 01$, the attacker first check the responses, $P(111\mathbf{00})$ and $P(111\mathbf{01})$. On one hand, If the responses are different, that is, $P(111\mathbf{00}) \neq P(111\mathbf{01})$, the attacker is sure that $p_0(\mathbf{00})$ and $p_0(\mathbf{01})$ are different and thus 00 and 01 belong to different sets. The reason is that p_1 in this case takes as input 10 and p_2 takes as input 11 for the two applied challenges. According to Table 2.1, the Layer 2 PUF will receive 11**0** and 11**1** as challenges. Note that due to the reproducibility of the PUF, p_1, as well as p_2, will produce the same response for the same challenge. If the responses of P are different, the only possibility is that bit-0 output of Layer 1 PUFs are different for bit 0, which is 0 and 1 in this case. On the other hand, if the responses are the same, the attacker cannot be sure that $p_0(\mathbf{00})$ and $p_0(\mathbf{01})$ are the same because in the Layer 2 PUF, different challenges can be paired with the same response. This corresponds to the case where the Layer 2 PUF responses the same for the two challenges 11**0** and 11**1**. In this case, the attacker has to try more challenges to increase his or her belief of p_0 giving the same response for 00 and 01 or encounter a case where P responds differently. After the check for each Layer 1 PUF, the attacker can obtain the *Guess* column in Table 2.1.

Phase 2

In this phase, the attacker uses the information from the first phase to enumerate the challenge space of the Layer 2 PUF. Algorithm 2 shows the steps in the second phase. Notice that the collected information in $S_{i,0}$ and $S_{i,1}$ is used to construct an input challenge to the composed PUF. It does this by selecting an n-bit string $\mathbf{u}$, and replacing one of the sub-strings u_i with an m-bit string from the first phase. The resulting string $\mathbf{c}$ is applied to the composed PUF P, and the challenge response pair is saved. Notice that this approach requires enumerating the n-bit challenge space.

For the same CPUF$(3, 2)$ P discussed in Section 2.4.2, the attacker will iterate through all strings in $\{0, 1\}^3$. For each string, the attacker derives a challenge that corresponds to

Algorithm 2: Class-Construction(S, P)

```
Input  : $S = \{S_{i,0}, S_{i,1} | 0 \leq i < n\}$ and CPUF $P$
Output: Set of special CRPs $Y$ and the response vector $\mathbf{y} \in \{0,1\}^n$
1 $Y = \emptyset$;
2 $\mathbf{y} = \mathbf{0}$;
3 for $\mathbf{u} = (u_0, u_1, ..., u_{n-1}) \in \{0,1\}^n$ do
4     $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_{n-1}), \mathbf{c}_i \in S_{i,u_i}$;
5     $y_{\mathbf{u}} = P(\mathbf{c})$;
6     $Y = Y \cup \{(\mathbf{c}, y_{\mathbf{u}})\}$;
```

the guess. For example, for the string 110, the attacker can choose from the challenges from the blue cells in Table 2.1 to replace each bit of the string. One of such possible challenges is $\mathbf{c} = 100100$. The attacker then applies challenge $\mathbf{c}$ to P and saves $(\mathbf{c}, P(\mathbf{c}))$ in Y.

Layer 1 PUF	Challenge	Response	Guess
p_0	00	0	$S_{0,0}$
	01	1	$S_{0,1}$
	10	0	$S_{0,0}$
	11	1	$S_{0,1}$
p_1	00	0	$S_{1,1}$
	01	0	$S_{1,1}$
	10	1	$S_{1,0}$
	11	1	$S_{1,0}$
p_2	00	1	$S_{2,0}$
	01	0	$S_{2,1}$
	10	0	$S_{2,1}$
	11	1	$S_{2,0}$

Table 2.1: In the Layer 1 PUFs truth table, grep column highlights the responses that is unknown to the attacker during the process of CA-ATK.

Forgery

To complete the attack, an attacker needs to forge the response of the CPUF given an unseen challenge $\mathbf{c}$ using information from Phase 1 and Phase 2. For example, for a challenge $\mathbf{y} = 010110$, the attacker can match $\mathbf{y}$ with $\mathbf{c} = 100100$ in Y and simple returns $P(\mathbf{c})$ from Y.

Chapter 3

System Model

3.1 Graphical Representation

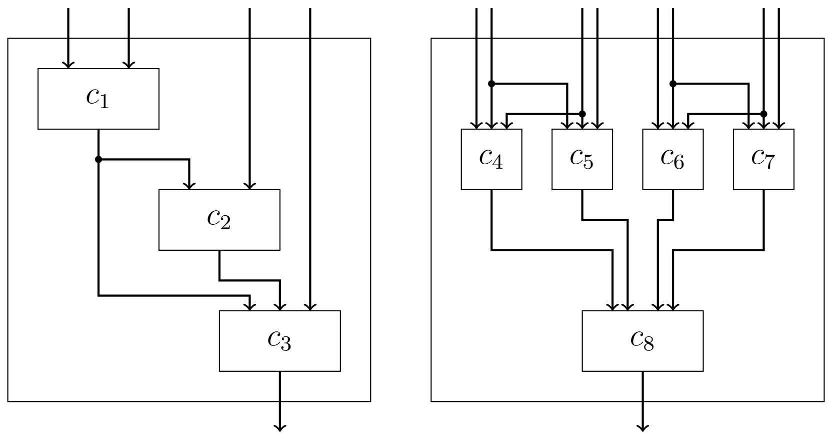


Figure 3.1: Examples of PUFs from composition of other PUFs. To the left, the constituent PUFs are in three levels to yield a composition whose domain is $\{0,1\}^4$. To the right is a composition with domain $\{0,1\}^8$, with the constituent PUFs in two layers. The PUFs at the first layer, c_4 and c_5, are each 3-input PUFs, and share two inputs to the composition.

We now introduce a general model for how PUFs can be composed to yield other PUFs. Then, we discuss the family of PUFs within that model on which we focus in this paper.

That family includes, as special cases, certain constructions from prior work [20].

A PUF is a physical realization of a function, $p\colon \{0,1\}^i \to \{0,1\}$, i.e., it maps an i bit input to a one bit output. Its intent is to serve as a random function, i.e., a function chosen uniformly from the set of all functions that map i bits to one bit. We perceive a PUF that results from composition as a directed graph, $P = \langle V_P, E_P\rangle$; we show two examples in Figure 3.1. Each vertex $u \in V_P$ is a PUF. Each edge, $e \in E_P$, maps a PUF to an input of another PUF. Thus, E_P can be perceived as a relation, $E_P \subseteq V_P \times V_P \times \mathbb{Z}^+$, where $\mathbb{Z}^+$ is the set of positive integers, where $\langle u, v, j\rangle \in E_P$ means that the output of the PUF u is provided as the j^{th} input of the PUF v. Our constraints are that given an edge $\langle u, v, j\rangle$, where $v\colon \{0,1\}^{i_v} \to \{0,1\}$, (i) $1 \leq j \leq i_v$, and, (ii) for every $\langle v, j\rangle$, there is exactly one edge incident on it, i.e., there is exactly one $u \in V_P$ such that $\langle u, v, j\rangle \in E_P$. In Figure 3.1, in the PUF to the left, the output of the constituent PUF c_1 is the first input to each of c_2 and c_3. So we have edges $\langle c_1, c_2, 1\rangle$ and $\langle c_1, c_3, 1\rangle$.

To represent inputs to and outputs from the composed PUF as a whole, we assume that we have two distinguished sets of edges, $\mathit{inp}, \mathit{outp} \subseteq E_P$, where each edge in *inp* has no source vertex, and each edge in *outp* has no destination PUF input. That is, the former is of the form $\langle \cdot, v, j\rangle$, and the latter is of the form $\langle u, \cdot, \cdot\rangle$. In Figure 3.1, the two inputs to the far left are input to the constituent PUF c_1. The output of the constituent PUF c_3 is the output of the composition.

Restrictions We now consider restrictions to the above rather general model for composition. As Section 3.2 establishes, notwithstanding such restrictions, we can realize PUFs which, asymptotically, yield the maximum possible attack-resistance. Futhermore, such restrictions yield more feasible constructions in practice. We present increasing restrictions that culminate in the family that captures prior work on compositions, and on which we focus.

As a first restriction, we require that the graph $P = \langle V_P, E_P\rangle$ is acyclic. Both PUFs in Figure 3.1 are acyclic. By acyclic, we mean that the conventional directed graph $\langle V_P, F_P\rangle$, where $F_P = \{\langle u, v\rangle \in V_P^2 \mid \langle u, v, j\rangle \in E_P \text{ for some } j\}$, is acyclic. Given such a PUF that is acyclic, we can consider its topologically sorted version, i.e., one in which all edges go from left to right only. This allows us to associate a *level* with each constituent PUF. The PUF to the left in Figure 3.1 is a composition of three PUFs, each at a level. We could further constrain the levels to be the stricter *layers*. A constituent PUF is at Layer 1 if the only edges incident on it are those from *inp*, the set of inputs to the composition as a whole. Then, for a PUF at layer $l > 1$, all edges incident on its inputs are those from PUFs at layer $l - 1$. The PUF to the right in Figure 3.1 is such a layered construction.

Family on which we focus The family of PUFs from composition on which we focus are layered constructions. The inputs to the composition are inputs to the constituent PUFs at Layer 1. The outputs from the PUFs at Layer 1 serve as inputs to a single PUF at Layer 2 and so on. The output from the constituent PUF at the last layer is the output to the composed PUF as a whole. An example composition with 2 layers is to the right in Figure 3.1.

More specifically, for the composition with 2 layers, we associate our compositions with four parameters. (i) The number of inputs to the composition as a whole, i. In the PUF to the right in Figure 3.1, $i = 8$. (ii) The number of inputs to each constituent PUF that belongs to Layer 1 PUFs, denoted as m. We adopt the restriction that each PUF that is not in Layer k takes the same number of inputs. In the PUF to the right in Figure 3.1, $m = 3$. (iii) The number of *partitions*, r, on the inputs, where the inputs in a partition serve as input to only a subset of the Layer 1 constituent PUFs. In the PUF to the right in Figure 3.1, we have $r = 2$ partitions. The first four inputs are in one partition, and the others are in the other. This partition on the inputs induces the partition $\{c_4, c_5\}, \{c_6, c_7\}$ on the Layer 1 PUFs. (iv) the number of input bits that each PUF at Layer 1 has in common with another PUF at Layer 1, denoted s. We also investigate PUFs whose compositions have more than 2 layers. In this case, the composition is characterized by the number of layers, denoted as k and how the output of each layer is mapped to another layer. In the PUF to the right in Figure 3.1, we have $k = 2$ layers.

In the PUF to the right of Figure 3.1, $s = 2$, because each of $c_4, \ldots, c_7$ shares 2 bits of input with another PUF at Layer 1. We adopt the restriction that the s is the same for all Layer 1 PUFs.

Notation The notation we adopt to denote a PUF in our family is [**i, m, r, s**]. For example, the PUF to the right in Figure 3.1 is denoted $[\mathbf{i} = 8, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 2]$.

Hardware Cost The strength of the PUFs comes at a price. We identify the hardware cost for our generalized model of PUFs.

Definition 3.1.1. The hardware cost of a PUF P is the number of constituent PUFs.

3.2 Analysis of Resistance to Attack

We now establish, analytically, the level of resistance of an instance from our family of PUFs to attack. We assume an attacker that seeks to fully characterize a PUF. By that we mean the following. Given a PUF $p\colon \{0,1\}^i \to \{0,1\}$, we say that p is fully characterized by a function $f\colon \{0,1\}^i \to \{0,1\}$ if $f(x) = p(x)$ for all $x \in \{0,1\}^i$ with probability 1. In practice, we typically relax this probability, e.g., to 95% only. We say that an attacker has fully characterized p when he is in possession of such an f. This is the same attacker characterization as has been adopted in the literature.

An attacker is provided the following capabilities. (i) Black-box access to p. That is, the attacker is allowed to exercise p with inputs, and observe the corresponding outputs. And, (ii) the attacker knows the design of the composition. That is, in our case, he knows the graph $P = \langle V_P, E_P \rangle$ of the PUF under attack.Thus, the only thing the attacker does not know are the actual functions that the constituent PUFs realize.

We quantify the strength of a PUF's resistance to attack as the number of queries the attacker needs to perform to the black-box. If for a PUF p, the attacker must perform n_p queries to the black-box, and for another PUF q the attacker must perform n_q queries, and $n_q < n_p$, then we deem the PUF p to be strictly more resistant to attack than the PUF q.

In our analysis below, we make the following idealization assumption. We assume that each constituent PUF is a random function. The reason we do this is that, our analysis pertains really to the manner in which we compose, rather than some artifact of the constituent PUFs.

Notation Our focus is the layered PUFs in the composition, however, the analysis applies to any PUF, whose graph is acyclic. Given a composition denoted by a graph $P = \langle V_P, E_P \rangle$, a partition on V_P into two set of vertices, namely, S_P and T_P, results in a cut of the graph. We denote the cut as $Q = \langle S_P, T_P \rangle$. We collect the edges crossing the cut in an set $E(Q)$. For all edge $e = \langle u, v, j \rangle \in E(Q)$, the following condition holds: $u \in S_P$ and $v \in T_P$, indicating a connection from the output of a constituent PUF to another constituent PUF.

We denote as $N(Q)$ the unique bits crossing the cut Q. That is

$$N(Q) = \{u | e = \langle u, v, j \rangle \in E(Q), u \in S_P, v \in T_P, 1 \le j \le i_v\}$$

Furthermore, we denote the cut with maximal $|N(Q)|$ as $Q^\star = \langle S_P, T_P \rangle^\star$, named *max-cut*. And the unique bits of a max-cut $Q^\star$ is denoted as $N^\star(Q^\star)$. Note that for a PUF composition, there could be more than one max-cut $Q^\star$.

Assuming $q = |N^\star(Q^\star)|$, we denote as $u_1, u_2, \ldots, u_q$ the bits in $N^\star(Q^\star)$. We denote as $C_{Q^\star} : \{0,1\}^i \to \{0,1\}^q$ the mapping that maps an input of the PUF as a whole to the concatenation of unique bits in $N^\star(Q^\star)$. For a set $X \subseteq \{0,1\}^i$, we denote as $C_{Q^\star}[X]$ the set $\{C_{Q^\star}(y) \in \{0,1\}^q \mid y \in X\}$, i.e., all possible values of bits in $|N^\star(Q^\star)|$ on inputs from X. Thus, $C_{Q^\star}\left[\{0,1\}^i\right]$ is the range of $C_{Q^\star}$.

Suppose the set of queries the attacker issues to the black-box is $B \subseteq \{0,1\}^i$. The corresponding set of outputs of cut $Q^\star$ is $C_{Q^\star}[B]$. Then, we have the following claim.

Claim 1. *If a PUF from composition P has been fully characterized after queries from $B \subseteq \{0,1\}^i$, then for all cut Q on P, $C_{Q^\star}[B] = C_{Q^\star}\left[\{0,1\}^i\right]$.*

Proof. We can prove the above claim by contradiction. If there exists a max-cut $Q^\star = \langle S_P, T_P\rangle^\star$ on P and there exists some input $y \in \{0,1\}^i$, such that $C_{Q^\star}(y) \notin C_{Q^\star}[B]$, then given that the PUFs in T_P form a random function, the attacker has at best a 1/2 probability of guessing the output of the PUF P on input y. □

Claim 2. *There exists a PUF for which $\left|C_{Q^\star}\left[\{0,1\}^i\right]\right| \geq 2^q$, where $q = |N^\star(Q^\star)| \leq i$.*

Proof. The proof for the above claim is by observing a PUF with 2 layers. Note that in this composition, the max-cut $Q^\star = \langle S_P, T_P\rangle^\star$ is obtained by setting S_P as the set of Layer 1 PUFs and T_P as the set of Layer 2 PUF. In Layer 2, we use a PUF with q-bit inputs. And in Layer 1, we use the xor, $\oplus$, of the input bits of PUF as a whole, as the Layer 1 PUF. Then we have a sequence $u_1, u_2, \ldots, u_q$ of S_P bits, where $q = |N^\star(Q^\star)|$, such that u_j has an input that is not input to any u_k where $k < j$. That is, every PUF as we go forward in that sequence has a "new" input. Thus, the concatenation of outputs of that sequence of Layer 1 PUFs is every bit string from $\{0,1\}^q$ when those Layer 1 PUFs are xor of their respective inputs. □

Consequence Claim 2 establishes a lower bound on the number of possible outputs from S_P of the PUFs to T_P. Claim 1 establishes that a lower bound on the attacker is that the number of black-box queries she issues must be at least the number of possible outputs from S_P. Together then, Claims 1 and 2 establish that a lower bound on the number of queries to the black-box that an attacker must issue to be confident that she has fully characterized a PUF P with max-cut $Q^\star$ is 2^q, where $q = |N^\star(Q^\star)|$.

Consider a 2 layer composition in which $s = \Theta(m), r = \Theta(1)$ and $m = O(i)$, then $q = \Theta(i)$. That is, under the assumption that the constituent PUFs are random functions, there exist PUFs that result from the composition whose strength is asymptotically bounded tightly by exactly the maximum possible CRP space, 2^i. Example values for s, r and m that meet the sufficient condition for this to be achieved are $s = m-1$, $r = 1$ and $m = i/16$.

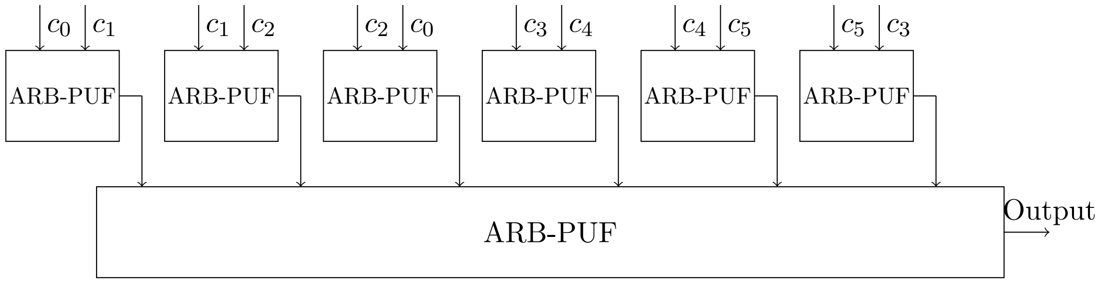


Figure 3.2: The 2-Layer $P \circ P$ instance has a mapping function $[\mathbf{i}=6, \mathbf{m}=2, \mathbf{r}=2, \mathbf{s}=1]$

We can compare the above lower-bound to the upper-bound to successfully attack a CPUF [19]. There, notwithstanding what the constituent PUFs are, an upper-bound number of queries to the black-box for a successful attack is $\frac{i}{m}2^m + 2^{\frac{i}{m}}$. Measured as bits, this strength is $O\big(\max\{m, \frac{i}{m}\}\big)$. This suggests that our broader admittance of ways to compose PUFs can be effective in yielding PUFs whose strength matches that of the maximum possible for a particular input size. The CPUF merely happens to be a weak member of the family.

3.3 Example Architectures

3.3.1 2 Layer Architectures

Figure 3.2 showcases a model that we used to represent $[\mathbf{i}=6, \mathbf{m}=2, \mathbf{r}=2, \mathbf{s}=1]$. Note that in the configuration, c_0, c_1, c_2 belongs to one partition and c_3, c_4, c_5 belongs to another partition.

3.3.2 Multi-Layer Architectures

Figure 3.3 shows a $P \circ P$ that has homogeneous mapping functions for Layer 1 PUFs and Layer 2 PUFs. In both layers, the $[\mathbf{i}=6, \mathbf{m}=2, \mathbf{r}=1, \mathbf{s}=2]$ partitions the input of the layer into 2 partitions. We denote this composition as:

$$\langle [\mathbf{i}=6, \mathbf{m}=2, \mathbf{r}=1, \mathbf{s}=2], [\mathbf{i}=6, \mathbf{m}=2, \mathbf{r}=1, \mathbf{s}=2] \rangle.$$

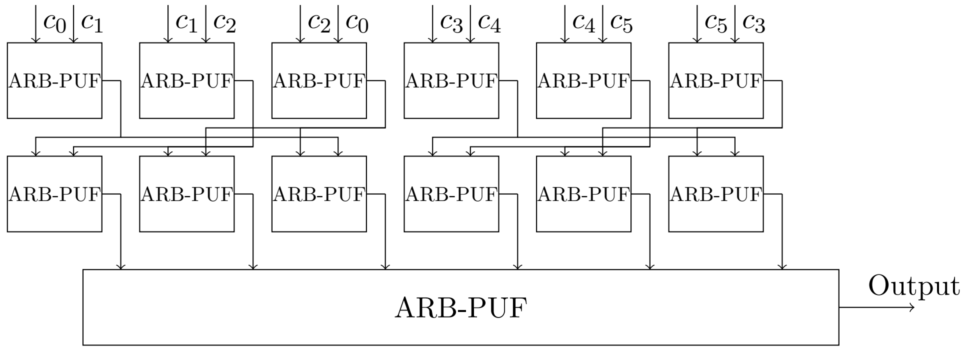


Figure 3.3: The 3-Layer $P \circ P$ uses $[\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 2]$ as the mapping functions for each of the layers.

Figure 3.4 shows a $P \circ P$ that has different mapping functions for Layer 1 PUFs and Layer 2 PUFs. For Layer 1 PUFs, the $[\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 2]$ partitions the input of the layer into 2 partitions. For Layer 2 PUFs, the $[\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$ mapping maps the output of the Layer 1 PUFs to Layer 2 PUFs. We denote this composition as:

$$\langle [\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 2], [\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1] \rangle.$$

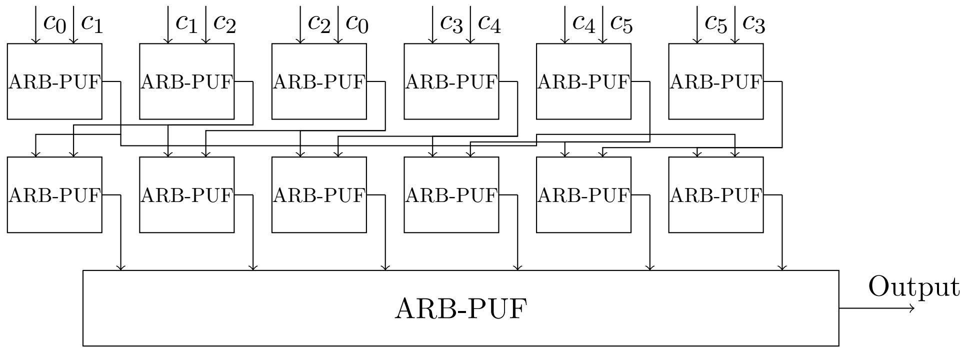


Figure 3.4: The 3 layer $P \circ P$ has heterogeneous mapping functions, where the mapping function in Layer 1 is $[\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 1]$ and the mapping function in Layer 2 is $[\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$.

Label	Architecture	$\lvert N^\star(Q^\star)\rvert$
A	CPUF$(3, 2)$ (Figure 3.3.3)	2^3
B	$[\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 1]$ (Figure 3.2)	2^6
C	$\langle[\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 2], [\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 2]\rangle$ (Figure 3.3)	2^6
D	$\langle[\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 2], [\mathbf{i} = 6, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]\rangle$ (Figure 3.4)	2^6

Table 3.1: Strengths of different PUF compositions.

3.3.3 Strength Analysis

In this section, we analyze the strengths of the PUFs discussed in Figure 3.2, Figure 3.3, Figure 3.4 and CPUF in Figure and list the result in Table 3.1. All of the compositions take as input 6-bit challenges. Table 3.1 shows that, adding multiple layers will not necessarily enhance the strength of the PUF, represented as $|N^\star(Q^\star)|$. The reason is that, for arbitrary number of layers, the max-cut divides the constituent PUFs into two sets, where each set as a whole can be regarded as a PUF.

Chapter 4

LR-CA-ATK: An Enhanced Modeling Attack on CPUF

4.1 Basics

We propose an enhanced attack on ARB-PUF based CPUF which exploits the property of outputs of Layer 1 PUFs and LAM for ARB-PUF. In phase 1 of CA-ATK, we make a guess about the Layer 1 PUFs by assigning their challenges to $S_{i,0}$ or $S_{i,1}$. We show that the guess can be characterized by a vector $\mathbf{x} \in \{0,1\}^n$ and this can be exploited to construct the LR-CA-ATK. In this section, we focus on constructing the model for a PUF $[\mathsf{i} = nm, \mathsf{m} = m, \mathsf{r} = n, \mathsf{s} = 0]$.

We denote the attacker's guess for the i-th Layer 1 PUF as $\tilde{p}_i$, which is constructed in phase 1. Thus, $\tilde{p}_i(\mathbf{a}) = j$, for $\mathbf{a} \in S_{i,j}$. The number of possible guesses an attacker makes about the output of the Layer 1 PUFs is large, however, we show that this can be fully characterized by an n-bit string. Specifically, we show that this applies to every Layer 1 PUF p_i in Theorem 1.

Theorem 1. *For all $0 \leq i < n$, $\mathbf{c} \in \{0,1\}^m$, $\tilde{p}_i(\mathbf{c}) = p_i(\mathbf{c}) \oplus x_i$, where $x_i \in \{0,1\}$ and $\oplus$ is binary exclusive-or.*

Proof. We choose an arbitrary challenge of p_i: $\mathbf{d} \in \{0,1\}^m$. If $p_i(\mathbf{c}) = p_i(\mathbf{d})$, then $\mathbf{c}$ and $\mathbf{d}$ belong to the same set, and thus, $\tilde{p}_i(\mathbf{c}) = \tilde{p}_i(\mathbf{d})$. If $p_i(\mathbf{c}) \neq p_i(\mathbf{d})$, then $\mathbf{c}$ and $\mathbf{d}$ belong to different sets, and thus, $\tilde{p}_i(\mathbf{c}) = 1 \oplus \tilde{p}_i(\mathbf{d})$. In both cases, $\tilde{p}_i(\mathbf{c}) = p_i(\mathbf{c}) \oplus \tilde{p}_i(\mathbf{d}) \oplus p_i(\mathbf{d})$ holds. $\square$

Algorithm 3: Enhanced-Class-Construction(S, P, D)

```
Input  : $S = \{S_{i,0}, S_{i,1} | 0 \leq i < n\}$, CPUF $P$ and $D \subseteq C = \{0,1\}^{nm}$ is a set of $N$
         challenges of the CPUF $P$.
Output: A LAM for $P_n$
1 $E = \emptyset$;
2 for c $\in D$ do
3     $\mathbf{u} = (u_0, u_1, ..., u_{n-1})$, where $\mathbf{c}_i \in S_{i,u_i}$;
4     $E = E \cup \{(\mathbf{u}, P(\mathbf{c}))\}$;
5 Compute the model $P_n$ with Logistic Regression, given the CRP set $E$;
```

According to Theorem 1, the attacker's guess of p_i is determined by a secret bit $x_i = \tilde{p_i}(\mathbf{d}) \oplus p_i(\mathbf{d})$.

Another fact we utilize is the LAM used for attacking ARB-PUF. In [17], the ARB-PUF P with n-bit challenge can be modeled as

$$P(\mathbf{c}) = sgn(\mathbf{w}\phi) = sgn(\mathbf{w}F(\mathbf{c})),$$

where $\mathbf{w} \in \mathbb{R}^{n+1}$ and $\phi \in \{-1,1\}^{n+1}$ is defined as:

$$\phi_i = F(\mathbf{c})(i) = \begin{cases} (-1)^{c_i}\phi_{i+1} = \prod_{j=i}^{n}(-1)^{c_j}, & \text{if } 1 \leq i \leq n \\ 1, & \text{if } i = n+1 \end{cases}. \tag{4.1}$$

We then show in Theorem 2 that if the challenge of an n-bit ARB-PUF P is XOR'ed with a secret key $\mathbf{x} \in \{0,1\}^n$, it can still be modeled with a LAM. The theorem shows that an ARB-PUF with XOR operation between a secret vector and its challenge is equivalent to another instance of ARB-PUF. Also, this theorem allows us to exploit the property of the attacker's guess in Theorem 1.

Theorem 2. *$P(\mathbf{c} \oplus \mathbf{x}) = sgn(\mathbf{w}F(\mathbf{c} \oplus \mathbf{x})) = sgn(\mathbf{v}F(\mathbf{c}))$, where $\mathbf{w}, \mathbf{v} \in \mathbb{R}^{n+1}$ and P is an n-bit ARB-PUF.*

Proof. Let $\phi = F(\mathbf{c} \oplus \mathbf{x})$, then

$$\begin{aligned}
\mathbf{w}F(\mathbf{c} \oplus \mathbf{x}) &= \sum_{i=1}^{n+1} w_i \phi_i && (4.2)\\
&= w_{n+1} + \sum_{i=1}^{n} w_i \prod_{j=i}^{n+1} (-1)^{c_j \oplus x_j} && (4.3)\\
&= w_{n+1} + \sum_{i=1}^{n} \left[\prod_{j=i}^{n+1} (-1)^{x_j} w_i\right] \prod_{j=i}^{n+1} (-1)^{c_j} && (4.4)\\
&= [\mathbf{w} \odot F(\mathbf{x})]F(\mathbf{c}) && (4.5)\\
&= \mathbf{v}F(\mathbf{c}), && (4.6)
\end{aligned}$$

where $\odot$ is element-wise product between vectors. Note that from Equation 4.3 to Equation 4.4, we apply the fact that $(-1)^{c_j \oplus x_j} = (-1)^{c_j}(-1)^{x_j}$ given that $c_j, x_j \in \{0, 1\}$. □

Now let us combine Theorem 1 and Theorem 2 to illustrate the weakness of CPUF. According to Theorem 1, the guess made in phase 1 of the CA-ATK about the output Layer 1 PUFs is actually not *far* from the actual output of these PUFs. If one knows about the secret string $\mathbf{x}$, this person could derive the actual output of the Layer 1 PUFs. And Theorem 2 shows that if the Layer 2 PUF is an ARB-PUF, this ARB-PUF is equivalent to another ARB-PUF, whose CRP space is defined by the guessed output and the original responses. These observations imply that an attacker does not need to explicitly obtain the secret string $\mathbf{x}$ to build the model for the Layer 2 ARB-PUF. Previous work has shown that building models for ARB-PUF with LR can be done efficiently [17], which we exploit here. We formalize these observations in Theorem 3.

Theorem 3. *Let P be a $[\mathsf{i} = nm, \mathsf{m} = m, \mathsf{r} = n, \mathsf{s} = 0]$. $\mathbf{c} \in \{0,1\}^{nm}$ is a challenge of P, $\mathbf{r} = (p_0(\mathbf{c_0}), p_1(\mathbf{c}_1), ..., p_{n-1}(\mathbf{c}_{n-1}))$ is the response vector of Layer 1 PUFs, $\tilde{\mathbf{r}} = (\tilde{p}_0(\mathbf{c_0}), \tilde{p}_1(\mathbf{c}_1), ..., \tilde{p}_{n-1}(\mathbf{c}_{n-1}))$ is the response vector constructed in phase 1 of the CA-ATK and $\mathbf{x}$ is a vector in $\{0,1\}^n$ that determines the guess, then*

$$P(\mathbf{c}) = p_n(\mathbf{r}) = p_n(\tilde{\mathbf{r}} \oplus \mathbf{x}) = sgn(\mathbf{v}F(\tilde{\mathbf{r}})),$$

where p_n is the Layer 2 PUF.

Algorithm 3 shows the procedure of using Theorem 3 to model the Layer 2 ARB-PUF. The attacker first randomly chooses a set D of nm-bit challenges. For each challenge in

w_0	w_1	w_2	w_3
0.1	0.3	0.2	-0.5

Table 4.1: Intrinsic paramters of the ARB-PUF in Layer 2 cannot be measured by the attacker without changing these values.

D, the attacker applies the reverse transformation as is done in Algorithm 2 to get a set of challenges of n-bit. This reversed transformation replaces every m-bit component with a single bit, based on whether the m-bit component is in $S_{\cdot,0}$ or $S_{\cdot,1}$. These challenges, combined with their corresponding responses, are fed into the LR attack to train a model for the CPUF. Compared to the class construction algorithm in CA-ATK, Algorithm 3 does not require the enumeration of all strings in $\{0,1\}^n$. The size of CRP set required is $N = |D|$, which can be much less than 2^n.

In summary, for an n-bit $P \circ P$, the attacker needs to enumerate 2^n CRPs to build the model. This is impractical for large values of n. Thus, an attacker cannot efficiently attack $P \circ P$ with this technique. Furthermore, an attacker could attack ARB-PUF based CPUF more efficiently yet $P \circ P$ is not subject to this.

4.2 An Example

In this section, an example demonstrates how an attacker can break the CPUF$(3,2)$ instance P, which is composed of ARB-PUF. Recall from Section 2.4.2, Table 2.1 shows the truth table of each of the Layer 1 PUFs obtained from Phase 1 of the LR-CA-ATK. In LR-CA-ATK, the same technique in Phase 1 is used to obtain Table 2.1. Also note that the secret key $\mathbf{x}$ in Theorem 2 is 110. Table 4.2 shows the intrinsic delay values of the Layer 2 PUF. Note that an attacker cannot measure the values in Table 4.1 without changing these parameters and thus these values are in gray background. Table 4.2 shows the challenges and responses that can be controlled or observed by the attacker and the output of Layer 1 PUFs which cannot be observed by the attacker. Table 4.3 shows the parameters of another ARB-PUF the attacker will actually model, based the third and forth columns in Table 4.2. Note that in this ARB-PUF, its response to challenge $\mathbf{c} \oplus \mathbf{x}$ is the same as the response of Layer 2 PUF to challenge $\mathbf{c}$.

Challenges Applied to the CPUF$(3, 2)$	Challenges to the Layer 2 ARB-PUF	Challenges used in Algorithm 3	Responses of P
00 00 00	1 0 0	0 1 0	1
00 00 01	1 0 1	0 1 1	1
00 10 00	1 1 0	0 0 0	1
01 00 00	0 0 0	1 1 0	0

Table 4.2: Attacker attempts to model the Layer 2 PUF using challenges from the first and third columns and responses from the forth column, without knowledge of the second gray column.

v_0	v_1	v_2	v_3
0.1	0.3	-0.2	-0.5

Table 4.3: Paramters of the ARB-PUF in Layer 2 are learned with challenges XORed with secret vector $\mathbf{x}$.

4.3 Case Study

Instead of using ARB-PUF as Layer 2 PUF, the designer may use other types of PUFs as the Layer 2 PUF in a CPUF. In this section, we study the effect of using XOR-PUF and LWS-PUF in a CPUF and show that the CPUF using XOR-PUF or LWS-PUF is also subject to LR-CA-ATK.

4.3.1 XOR-PUF as last level PUF

XOR-PUF features a combination of multiple chains of ARB-PUF where a final xor gate combines the result. In [17], the XOR-PUF is successfully attacked with an extension of the LAM. The XOR-PUF(n, c) P with n-bit challenge and c chains can be modeled as:

$$P(\mathbf{c}) = \prod_{i=1}^{c} sgn(\mathbf{w}_i \phi_i) = \prod_{i=1}^{c} sgn(\mathbf{w}_i F(\mathbf{c}))$$

where F is defined in equation 4.1. Note that all ARB-PUF in XOR-PUF takes the same input.

Following a similar approach in Theorem 2, we conclude in Theorem 4 that an XOR-PUF with XOR operation between a secret vector and its challenge is equivalent to another instance of XOR-PUF, where each of the component ARB-PUFs can be represented by another instance of ARB-PUF.

Theorem 4.
$$P(\mathbf{c} \oplus \mathbf{x}) = \prod_{i=1}^{c} sgn(\mathbf{w}_i F(\mathbf{c} \oplus \mathbf{x})) = \prod_{i=1}^{c} sgn(\mathbf{v}_i F(\mathbf{c})),$$
where $\mathbf{w}_i, \mathbf{v}_i \in \mathbb{R}^{n+1}$ *and* P *is an XOR-PUF(n, c).*

Proof. We can proof the equation by applying Theorem 2 for each inner-most term. □

Thus, if an XOR-PUF is used as the last level PUF in the CPUF configuration, we can leverage LR-CA-ATK and model the PUF.

4.3.2 LWS-PUF as last level PUF

LWS-PUF is very similar to XOR-PUF. An LWS-PUF that outputs 1-bit has an XOR-PUF in its heart and its challenge will go through an input network so that each ARB-PUF receive different challenges. In [17], an extension model of LAM similar to the XOR-PUF model is used:
$$P(\mathbf{c}) = \prod_{i=1}^{c} sgn(\mathbf{w}_i \phi_i) = \prod_{i=1}^{c} sgn(\mathbf{w}_i F(G_i(H_i(\mathbf{c}))))$$
where H_i circularly shifts the challenges by $i-1$ bits to the right, and G_i is defined as follows:

$$G_i(\mathbf{c})(\frac{n+j+1}{2}) = c_j, \text{for } j = 1 \tag{4.7}$$
$$G_i(\mathbf{c})(\frac{j+1}{2}) = c_j \oplus c_{j+1}, \text{for } j = 1, 3, 5, \ldots, n-1 \tag{4.8}$$
$$G_i(\mathbf{c})(\frac{n+j+2}{2}) = c_j \oplus c_{j+1}, \text{for } j = 2, 4, 6, \ldots, n-2 \tag{4.9}$$

.

Theorem 5 shows that for a LWS-PUF that output 1 bit, LWS-PUF with XOR operation between a secret vector and its challenge is equivalent to another instance of LWS-PUF.

Theorem 5.

$$P(\mathbf{c} \oplus \mathbf{x}) = \prod_{i=1}^{c} sgn(\mathbf{w}_i F(G_i(H_i(\mathbf{c} \oplus \mathbf{x})))) \tag{4.10}$$

$$= \prod_{i=1}^{c} sgn(\mathbf{v}_i F(G_i(H_i(\mathbf{c})))) \tag{4.11}$$

Proof. We prove the theorem by showing that

$$\mathbf{w}_i F(G_i(H_i(\mathbf{c} \oplus \mathbf{x}))) = (\mathbf{w}_i \odot F(G_i(H_i(\mathbf{x})))) F(G_i(H_i(\mathbf{c}))). \tag{4.12}$$

$\odot$ represents element-wise vector product.

Since H_i is a shift operation, it is obvious that the Equation 4.13 holds:

$$H_i(\mathbf{c} \oplus \mathbf{x}) = H_i(\mathbf{x}) \oplus H_i(\mathbf{c}). \tag{4.13}$$

Next, according to Equation 4.7, Equation 4.8 and Equation 4.9,

$$\begin{aligned}
G_i(\mathbf{c} \oplus \mathbf{x})(\frac{n+j+1}{2}) &= c_j \oplus x_j, \text{for } j = 1 \\
G_i(\mathbf{c} \oplus \mathbf{x})(\frac{j+1}{2}) &= (c_j \oplus x_j) \oplus (c_{j+1} \oplus x_{j+1}) \\
&= (c_j \oplus c_{j+1}) \oplus (x_j \oplus x_{j+1}), \text{for } j = 1, 3, 5, \ldots, n-1 \\
G_i(\mathbf{c} \oplus \mathbf{x})(\frac{n+j+2}{2}) &= (c_j \oplus x_j) \oplus (c_{j+1} \oplus x_{j+1}) \\
&= (c_j \oplus c_{j+1}) \oplus (x_j \oplus x_{j+1}), \text{for } j = 2, 4, 6, \ldots, n-2
\end{aligned}$$

, which implies Equation 4.14:

$$G_i(\mathbf{c} \oplus \mathbf{x}) = G_i(\mathbf{x}) \oplus G_i(\mathbf{c}) \tag{4.14}$$

Combining Equation 4.13, Equation 4.14 and Theorem 2, we can move $\mathbf{x}$ out of the $F(G_i(H_i(\cdot)))$ composition:

$$\begin{aligned}
\mathbf{w}_i F(G_i(H_i(\mathbf{c} \oplus \mathbf{x}))) &= \mathbf{w}_i F(G_i(H_i(\mathbf{x}) \oplus H_i(\mathbf{c}))) \\
&= \mathbf{w}_i F(G_i(H_i(\mathbf{x})) \oplus G_i(H_i(\mathbf{c}))) \\
&= (\mathbf{w}_i \odot F(G_i(H_i(\mathbf{x})))) F(G_i(H_i(\mathbf{c})))
\end{aligned}$$

Thus, if we choose $\mathbf{v}_i = F(G_i(H_i(\mathbf{x}))) \odot \mathbf{w}_i$ then Theorem 5 holds. □

Chapter 5

Evaluation

We separate our evaluation into five parts. The first part shows how $P \circ P$ with 2 layers compares with other state-of-the-art PUFs including CPUF composed using ARB-PUFs using an evolutionary strategy (ES) attack [17]. The second part shows that the LR-CA-ATK successfully models CPUF, but it is unable to model $P \circ P$ with various mapping functions. Using these results, we provide insights into properties that make a mapping function resilient to the LR-CA-ATK. We accomplish this by carefully selecting a few different mapping functions, and comparing them. The third part evaluates the statistical performance metrics of $P \circ P$ including the training times for the CA-ATK and LR-CA-ATK on the CPUF. It also includes our observations on the incurred hardware cost with $P \circ P$ and CPUF. Given that a $P \circ P$ with 2 layers is able to defense against ES modeling attacks and LR-CA-ATK, we extend our scope in the forth part, in which we evaluate the $P \circ P$ with more than 2 layers and different mapping function configurations. Our result shows the ability of $P \circ P$ with more than 2 layers can defense against the ES attack. Finally, we empirically show the ability of LR-CA-ATK to model CPUF, where the Layer 2 PUF is either XOR-PUF or LWS-PUF.

5.1 Framework

The experiments are performed on a simulation framework for exploring modeling attacks on various PUF architectures. The framework can be divided into two parts: simulation and modeling. The simulation framework features a detailed simulation at component level. For example, when simulating ARB-PUF, the framework can simulate the behaviour of individual multiplexer stages, including delays and the parameters of the distribution for generating the delays. The modeling framework features two backends: Tensorflow [2] and PyBrain [21]. Not only does the Tensorflow backend allows exploration on complicated models such as neural networks, but also it allows scalable modeling attacks if more computational resources, such as GPUs, are available. Also, extension to the framework can be easily made. The PyBrain, on the other hand, allows modeling PUFs using population-based method, for example, ES, and provides the framework with the ability to model PUFs that do not have a model with clear advantage such as LAM for ARB-PUFs. The simulation framework is publicly available for download [24].

5.2 Experimental Setup

For LR, we implement RProp [1], and for ES, we use the open-source implementation provided by PyBrain [21]. We use the meta-parameters for ES from [18] that have resulted in successful attacks on a variety of PUF architectures. Without loss of generality, our results use PUF instances with an 18-bit challenge. The 18-bit challenge allows us to explore different mapping functions while the attacks can still be practically conducted.

5.3 Results

We begin by showing that CPUF is vulnerable to LR-CA-ATK, but $P \circ P$ remains resilient. This establishes that $P \circ P$ with varying mapping functions results in a strengthened architecture with respect to the LR-CA-ATK. We use this result to discover properties of $P \circ P$ that contribute to its resilience. In particular, we investigate varying the input size of the Layer 1 PUFs, and the effect of mapping functions on the amount of sharing across Layer 1 PUFs.

5.3.1 Comparison against other state-of-the-art PUF architectures

We use the notation described in Table 5.1 to refer to PUFs, and their configurations. We compare four $P \circ P$ configurations: $[\mathsf{i} = 64, \mathsf{m} = 2, \mathsf{r} = 1, \mathsf{s} = 1]$, $[\mathsf{i} = 64, \mathsf{m} = 4, \mathsf{r} = 1, \mathsf{s} = 3]$, $[\mathsf{i} = 64, \mathsf{m} = 2, \mathsf{r} = 2, \mathsf{s} = 1]$ and $[\mathsf{i} = 64, \mathsf{m} = 4, \mathsf{r} = 2, \mathsf{s} = 3]$ against two CPUF architectures CPUF$(64, 2)$ and CPUF$(64, 4)$, FF-PUF$(64, 1)$, FF-PUF$(64, 6)$, MPUF$(64, 1)$, XOR-PUF$(64, 2)$, ARB-PUF, LWS-PUF$(64, 2)$ as shown in

Figure 5.1. We use ES to attack the aforementioned PUFs as ES only requires a parametric model. For all the PUFs in Figure 5.1, the training set and test set include 50,000 and 10,000 randomly generated CRPs, respectively. We observe that $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$, $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$, $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 1]$ and $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 2, \mathbf{s} = 3]$ sustain a prediction accuracy of 60.48%, 55.87%, 61.57% and 57.70% even after performing 2×10^6 evaluations. Other PUFs have the following prediction accuracy: FF-PUF with 6 loops has 97.95%, XOR-PUF has 97.6%, LWS-PUF has 97.47%, ARB-PUF has 99.14% and MPUF has 99.3%. Note that all alternative PUF architectures suffer a rise in their prediction accuracy to more than 97%. This result shows that a larger effort is required to attack $P{\circ}P$ and CPUF than the alternatives with ES. For instances of $P{\circ}P$, the architecture with the same number of partitions, the ones with more Layer 1 PUF stages show lower prediction accuracy. The difference is within 5%. For $P{\circ}P$ instances with the same Layer 1 PUF stages, the ones with fewer partition numbers show lower prediction accuracy. The difference is within 2%. Hence, our experiments confirm that $P{\circ}P$ provides additional resistance to ML modeling attacks with ES than other state-of-the-art PUF architectures.

5.3.2 Comparing Resilience of CPUF against $P{\circ}P$ using LR-CA-ATK.

Figure 5.2 shows the prediction accuracy of the LR-CA-ATK when applied to CPUF and $P{\circ}P$ with a varying number of CRPs. The prediction accuracy is a metric to describe a PUF's resistance to modeling attacks. A high prediction accuracy signifies less effort is required to model the PUF compared to one with lower prediction accuracy. Recall there are two phases in LR-CA-ATK. The experiment in Figure 5.2 varies the number of CRPs used in Phase 2 of the LR-CA-ATK. When we evaluate the model, we exclude the CRPs used in Phase 1 and Phase 2. The reason for this is that any CRP used for training the model is stored by the attacker since the PUF is actually exercised. We consider a

Table 5.1: The notation of different PUF architectures.

Notation	Architecture
FF-PUF(n, l)	n-bit FF-PUF with l loops
XOR-PUF(n, l)	n-bit XOR-PUF with l chains
LWS-PUF(n, l)	n-bit LWS-PUF with l chains
MPUF(n)	n-bit MPUF.
ARB-PUF(n)	n-bit ARB-PUF.

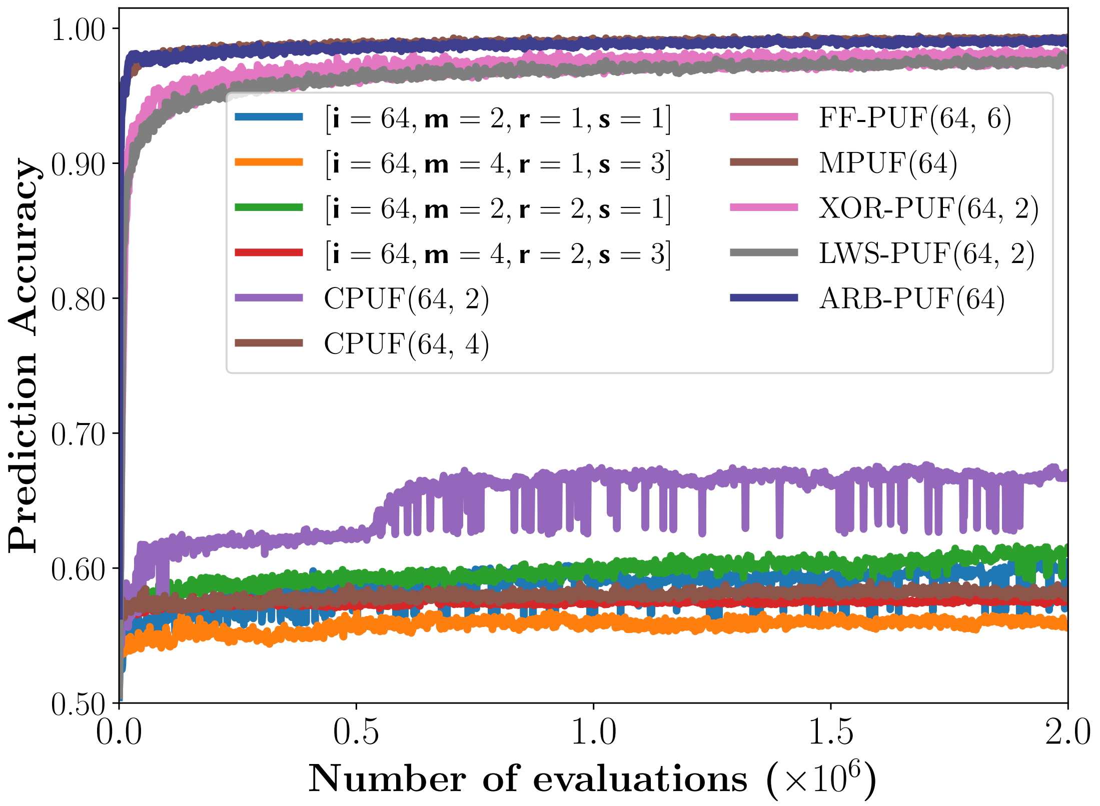


Figure 5.1: $P \circ P$ versus state-of-the-art PUFs under ES modeling attack.

PUF to be modeled when the prediction accuracy is around 95%. We show in Figure 5.2 that CPUF is modeled with more than 95% prediction accuracy when the number of the training CRPs is 2^7. However, with the training CRPs being 2^{13}, the LR-CA-ATK cannot model our proposed architectures with 95% prediction accuracy. Even if we use half of the CRP space, our results show that we cannot model $P \circ P$ with more than 95% prediction accuracy. Since increasing the number of training CRPs does not help us model $P \circ P$, we choose an in-between value of 2^{13} as the number of training CRPs in Phase 2 for the remaining experiments in Section 5.3.3.

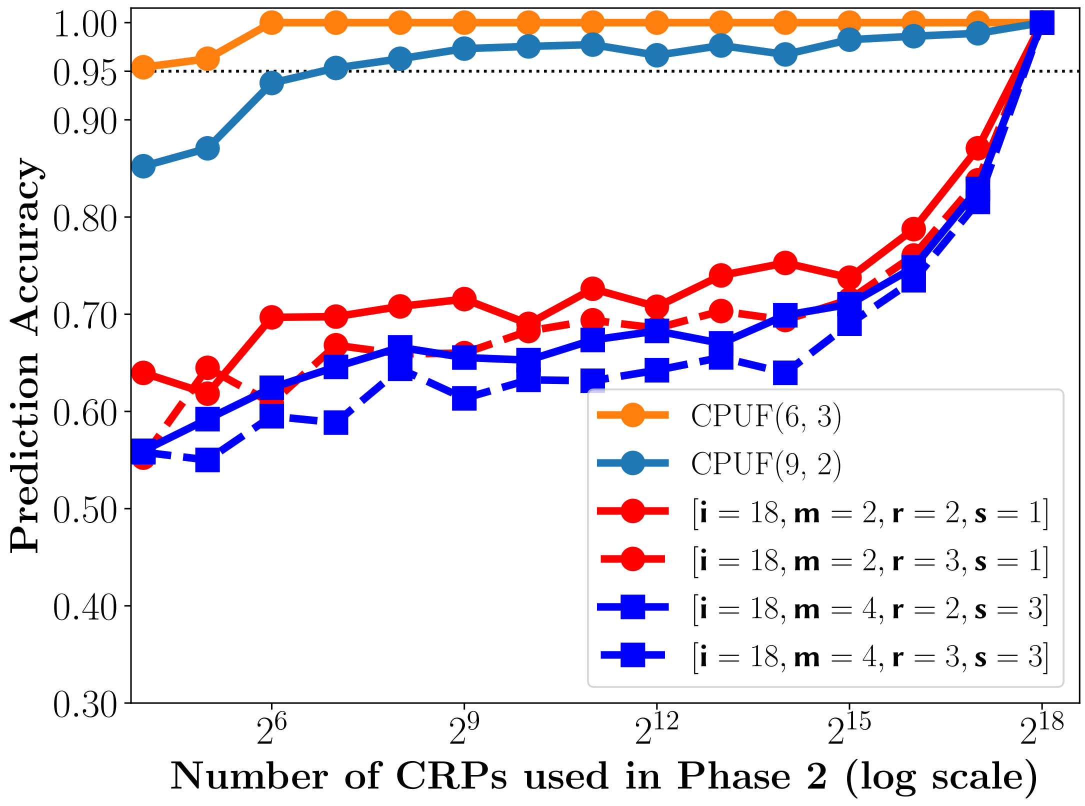


Figure 5.2: CPUF versus $P \circ P$ when varying the number of training CRPs in Phase 2 of the LR-CA-ATK.

5.3.3 Evaluating properties of mapping function

We evaluate two properties of mapping functions: 1) the resulting partition size and 2) number of partitions from mapping functions. Throughout this subsection, we assess the security with the *normalized prediction accuracy*. The normalized prediction accuracy is defined as the prediction accuracy divided by the number of CRPs used in Phase 1 and Phase 2. This metric measures for the prediction accuracy achieved per CRP used. Compared to a lower normalized prediction accuracy, a higher normalized prediction accuracy means that the CRP utilization is high and thus a PUF with a higher normalized prediction accuracy is less secure.

Partition size as a result of mapping functions. We investigate the effect of partition size resulting from the mapping function. In Figure 5.3a, we evaluate a class of mapping functions, which partition the Layer 1 PUFs into two partitions ($[\mathbf{i} = 18, \mathbf{m} = m, \mathbf{r} = 2, \mathbf{s} = 2]$). Here, the partition sizes are not necessarily the same. The lowest normalized prediction accuracy occurs when the size of the first partition is small or large compared to the other one. The normalized prediction accuracy peaks at the point where the partition sizes are equal and such a mapping function is less desirable as it offers reduced strength. This is because, if one of the partitions induced by the mapping function has a large size, the attacker must enumerate the larger partition in Phase 1 of the attack. However, if the partition does not have a large size, the attacker does not need to enumerate the large CRP space introduced by the partition. Thus, we propose using mapping functions where the partition for Layer 1 PUFs has a large size.

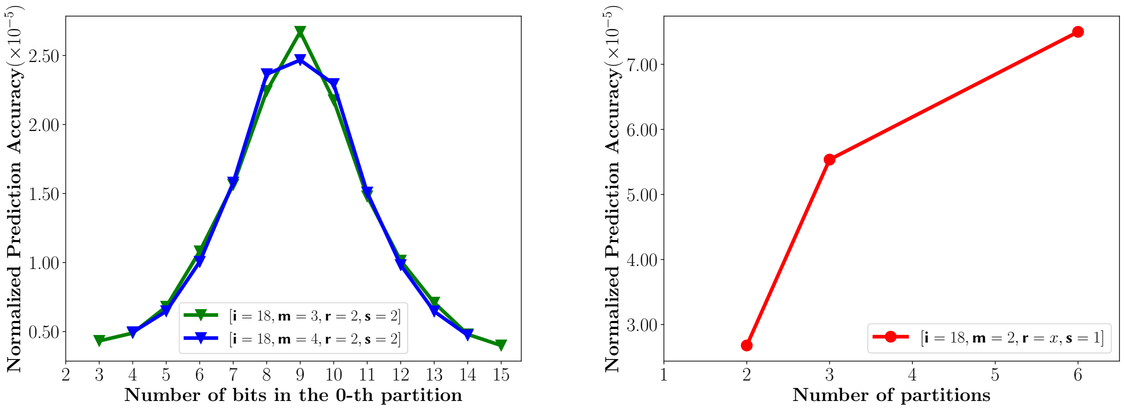
Number of partitions as a result of mapping functions. We also investigate the effect of number of partitions resulting from the mapping function. For this, we select mapping functions that divide Layer 1 PUFs into equally sized groups. Figure 5.3b shows that as the number of partitions increase, the normalized prediction accuracy increases and the attacker can model the PUF more efficiently. This means that a good choice for an mapping function is one that partitions the Layer 1 PUFs into a lower number of partition.

Table 5.2: LR-CA-ATK and CA-ATK on CPUF. Prediction rates and the training time are averaged over 5 trials.

Pred. Acc. LR-CA-ATK	Pred. Acc. CA-ATK	Train. Time LR-CA-ATK	Train. Time CA-ATK
	$\mathrm{CPUF}(20,4)$		
99.37%	99.92%	**31.38s**	162.91s
	$\mathrm{CPUF}(64,4)$		
99.62%	-	**33.94s**	-

5.3.4 Training time versus accuracy

Table 5.2 compares the training time and accuracy of LR-CA-ATK against CA-ATK. Note that it is impractical to apply CA-ATK to $\mathrm{CPUF}(64,4)$ as its Phase 2 requires the enumeration over a space of 2^{64} binary strings. The result clearly shows that LR-CA-ATK succeeds in modeling both CPUFs, but CA-ATK fails to complete for $\mathrm{CPUF}(64,4)$. In addition, LR-CA-ATK only takes 20% of the time CA-ATK takes for $\mathrm{CPUF}(20,4)$.



(a) Normalized prediction accuracy for $P \circ P$ as a function of the number of bits of the first partitions.

(b) Normalized prediction accuracy for $P \circ P$ as a function of the number of partitions.

Figure 5.3: Normalized prediction accuracy given different properties of mapping functions.

Table 5.3 shows the training time and the best achieved prediction rate for 50,000 CRPs. We observe that $[\mathbf{i}=64, \mathbf{m}=2, \mathbf{r}=1, \mathbf{s}=1]$ has a higher prediction accuracy by $\sim$4.6% over $[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3]$, but the training time for $[\mathbf{i}=64, \mathbf{m}=2, \mathbf{r}=1, \mathbf{s}=1]$ is larger due to the fact that the model with 4-stage Layer 1 PUFs has higher complexity. This result suggests that $[\mathbf{i}=64, \mathbf{m}=2, \mathbf{r}=1, \mathbf{s}=1]$ provides just as much resilience as $[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3]$. Also note that both $P \circ P$ configurations outperform FF-PUFs in terms of their prediction accuracy.

5.4 Performance Metrics

We also present the uniformity and uniqueness performance metrics in Table 5.4. Our results show that $P \circ P$ with the different mapping functions offer good uniqueness traits. However, we find that $P \circ P$s with a $\mathbf{r}=2$ the uniformity is biased.

5.5 Hardware Cost

In Table 5.5, we compare the hardware cost of $P \circ P$ and CPUF.

Table 5.3: Prediction accuracy for the best of 40 trials for $P \circ P$ and other PUFs. The training time is the averaged.

Architecture	Pred. Acc. Best Run	Training Time
$[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$	60.48%	1:57 hrs
$[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$	55.87%	2:29 hrs
$[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 1]$	61.57%	1:53 hrs
$[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 2, \mathbf{s} = 3]$	57.70%	2:31 hrs
CPUF$(64, 2)$	66.86%	1:35 hrs
CPUF$(64, 4)$	58.16%	1:55 hrs
FF-PUF$(64, 6)$	97.95%	5:59 hrs
XOR-PUF$(64, 2)$	97.60%	0:20 hrs
LWS-PUF$(64, 2)$	97.47%	0:21 hrs
MPUF(64)	99.30%	0:12 hrs
ARB-PUF(64)	99.14%	0:31 hrs

Table 5.4: Uniqueness and uniformity for $P \circ P$.

PUF Architectures	Uniqueness	Uniformity
$[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$	50.4%	47.6%
$[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 2, \mathbf{s} = 3]$	50.5%	61.7%
$[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$	49.5%	49.5%
$[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 2, \mathbf{s} = 1]$	50.4%	63.6%

$[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$ and $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$ are the $P \circ P$ instances that take 64-bit challenge. $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$ is slightly better than $[\mathbf{i} = 64, \mathbf{m} = 2, \mathbf{r} = 1, \mathbf{s} = 1]$ according to our evaluations with the ES attack. This enhanced security comes at the price of 66.7% larger number of stages. $[\mathbf{i} = 64, \mathbf{m} = 4, \mathbf{r} = 1, \mathbf{s} = 3]$ and CPUF$(64, 4)$ have the same hardware cost and CPUF$(64, 2)$ can take 128-bit challenge. However, we already showed that CPUF$(64, 4)$ can be modeled using LR-CA-ATK with far fewer CRPs than the 128-bit challenge space.

Table 5.5: The hardware cost of different PUF compositions.

PUF Architectures	ARB-PUF(2)	ARB-PUF(4)	ARB-PUF(64)	Total Number of Stages
$[\mathbf{i}=64, \mathbf{m}=2, \mathbf{r}=1, \mathbf{s}=1]$	64	–	1	192
$[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3]$	–	64	1	320
CPUF$(64, 2)$	64	–	1	192
CPUF$(64, 4)$	–	64	1	320
LWS-PUF$(64, 2)$/XOR-PUF$(64, 2)$	–	–	2	128

5.6 Evaluating Resilience of $P{\circ}P$ with Multiple Layers using ES

In this section, we evaluate the $P{\circ}P$ with more than two layers using ES. We use the notation listed in Table 5.6 to refer to PUFs and the configurations. The five $P{\circ}P$s with multiple layers feature different aspect of composition of multiple layers. C1 and C2 features the homogeneous setup of mapping functions between Layer 1 and Layer 2 PUFs. In C1, the challenge of the $P{\circ}P$ as a whole and the output of the Layer 1 PUFs are mapped using $[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3]$ to the next Layer. While in C2, we use $[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=2, \mathbf{s}=3]$ as the mapping function and thus, the challenge and the output are grouped into two partitions. C3 and C4 features the heterogeneous setup of mapping functions between Layer 1 and Layer 2 PUFs, in which the mapping functions are different for the challenge and the output of the Layer 1 PUF. In C5, the $P{\circ}P$ has 3 layers, and the challenge bits, the output of Layer 1 PUFs and the output of Layer 2 PUFs are mapped using $[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3]$. For all PUFs in Figure 5.4, the training set and the test set include 50,000 and 10,000 randomly generated CRPs respectively. After performing 2×10^6 evaluations, C1, C2, C3, C4 and C5 sustain a prediction accuracy of

Table 5.6: The notation of different multi-layer $P{\circ}P$ architectures.

Notation	Mapping functions
C1	$\langle[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3], [\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3]\rangle$
C2	$\langle[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=2, \mathbf{s}=3], [\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=2, \mathbf{s}=3]\rangle$
C3	$\langle[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=2, \mathbf{s}=3], [\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3]\rangle$
C4	$\langle[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3], [\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=2, \mathbf{s}=3]\rangle$
C5	$\langle[\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3], [\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3], [\mathbf{i}=64, \mathbf{m}=4, \mathbf{r}=1, \mathbf{s}=3]\rangle$

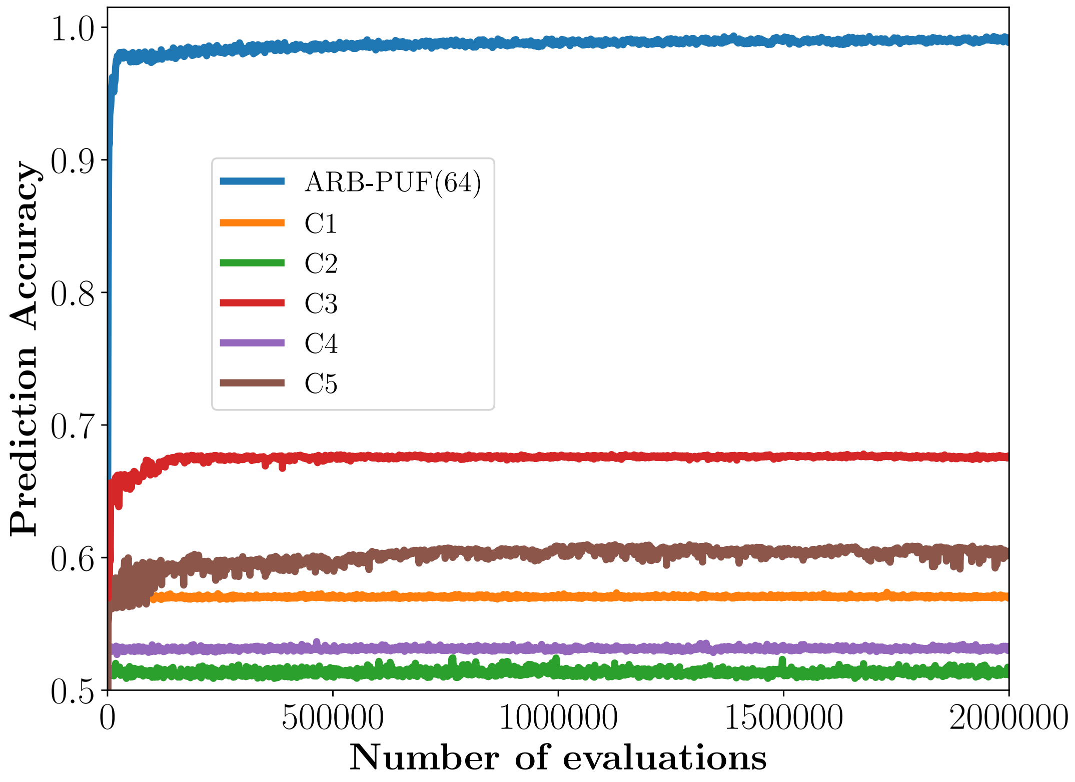


Figure 5.4: Prediction accuracy for $P \circ P$ as a function of evaluations.

57.12%, 51.15%, 67.58%, 53.07% and 60.49%, respectively. Compared to results of $P \circ P$ with 2 layers, having more than 2 layers does not necessarily enhance the resilience of the $P \circ P$ to ES attack, while in the meantime, more layers incur more hardware cost. Thus, we conclude that $P \circ P$ with 2 layers is resilient enough to defense against ES attack with less hardware compared to $P \circ P$ with more than 2 layers.

5.7 Case Study Results

In this section, we show the empirical results for studying whether the LR-CA-ATK can model CPUF architectures composed of other types of PUFs. According to Theorem 4

PUF Architecture	Prediction Accuracy	Training Time (s)	Number of ARB-PUFs in Layer 2 PUF
$CPUF(20, 4)-L$	97.59%	29	2
$CPUF(20, 4)-L$	97.69%	33	3
$CPUF(20, 4)-L$	97.71%	45	4
$CPUF(64, 4)-L$	97.75%	136	2
$CPUF(64, 4)-L$	97.7%	104	3
$CPUF(64, 4)-L$	97.68%	113	4
$CPUF(20, 4)-X$	97.68%	29	2
$CPUF(20, 4)-X$	97.67%	31	3
$CPUF(20, 4)-X$	97.68%	34	4
$CPUF(64, 4)-X$	97.66%	96	2
$CPUF(64, 4)-X$	97.68%	105	3
$CPUF(64, 4)-X$	97.69%	111	4

Table 5.7: LR-CA-ATK on $P \circ P$ composed of XOR-PUF and LWS-PUF

and Theorem 5, the LR-CA-ATK should be able to model CPUF with XOR-PUF or LWS-PUF as the Layer 2 PUF due to the fact that XORing the challenge of an XOR-PUF or a LWS-PUF is equivalent to another instance of the corresponding PUF. Table 5.7 validates the theory. In Table 5.7, LR-CA-ATK is able to model both CPUF with 80-bit challenges and 256-bit challenges with a prediction accuracy of more than 89% within 63 seconds. For 256-bit challenges, the attacker can achieve a prediction accuracy of 94.79%. Using CA-ATK, an attacker can model CPUF with 40-bit challenges with a prediction accuracy of 99%. However, in Table 5.8, the successful modeling requires more than 10$\times$ the time compared to that of the LR-CA-ATK. Also, LR-CA-ATK is unable to model 256-bit CPUF as it requires an enumeration of a CRP space of size 2^{64}.

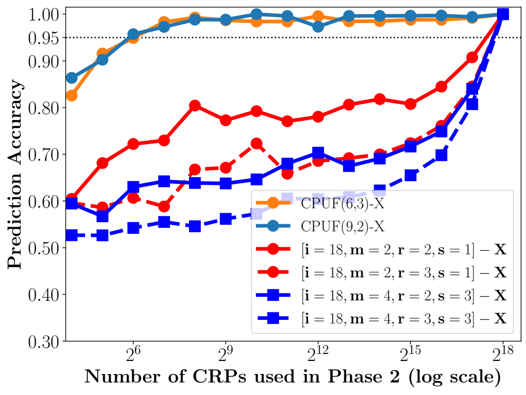
We then examine whether $P \circ P$ composed of XOR-PUF or LWS-PUF can be enhanced. Figure 5.5a and Figure 5.5b show the prediction accuracy of the LR-CA-ATK when the number of CRPs used in Phase 2 is varying. We append an X or an L to indicate that the data is collected for Layer 2 PUF being XOR-PUF or LWS-PUF, respectively. The result indicates that, for both types of CPUFs, it can be modeled with 95% prediction accuracy given 2^7 training CRPs in Phase 2. For $P \circ P$s, even if half of the CRP space is used in Phase 2, the results indicate that LR-CA-ATK is not able to model different configurations of $P \circ P$s with more than 95% of prediction accuracy.

Finally, we attempt to model $P \circ P$ composed of XOR-PUF and LWS-PUF with ES.

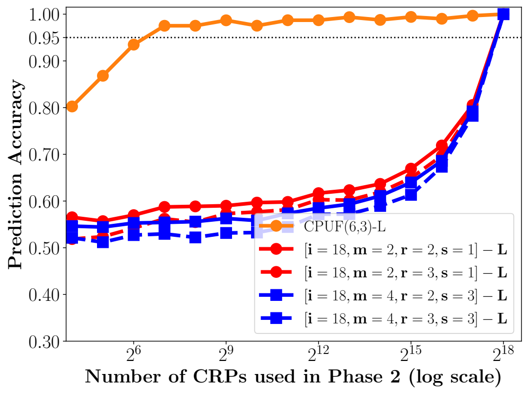
PUF Architecture	Prediction Accuracy	Training Time (s)	Number of ARB-PUFs in Layer 2 PUF
$CPUF(20,\ 4){-}X$	99.31%	1068	2
$CPUF(20,\ 4){-}X$	100.0%	1477	3
$CPUF(20,\ 4){-}X$	100.0%	1730	4
$CPUF(20,\ 4){-}L$	100.0%	1404	2
$CPUF(20,\ 4){-}L$	100.0%	1348	3
$CPUF(20,\ 4){-}L$	100.0%	1878	4

Table 5.8: CA-ATK on $P{\circ}P$ composed of XOR-PUF and LWS-PUF

Figure 5.6 shows that, equiped with mapping function, the LWS-PUF and XOR-PUF gained resilience to the ES attack. Specifically, when equipped with mapping function, XOR-PUF can achieve a prediction accuracy of 55.52% and LWS-PUF can achieve a prediction accuracy of 50.96%. In the meantime, XOR-PUF achieves a prediction accuracy of 97.6% and LWS-PUF achieves a prediction accuracy of 97.47%.



(a) XOR-PUF acts as the Layer 2 PUF.



(b) LWS-PUF acts as the Layer 2 PUF.

Figure 5.5: CPUF versus $P{\circ}P$ composed of different Layer 2 PUFs when varying the number of training CRPs in Phase 2 of LR-CA-ATK.

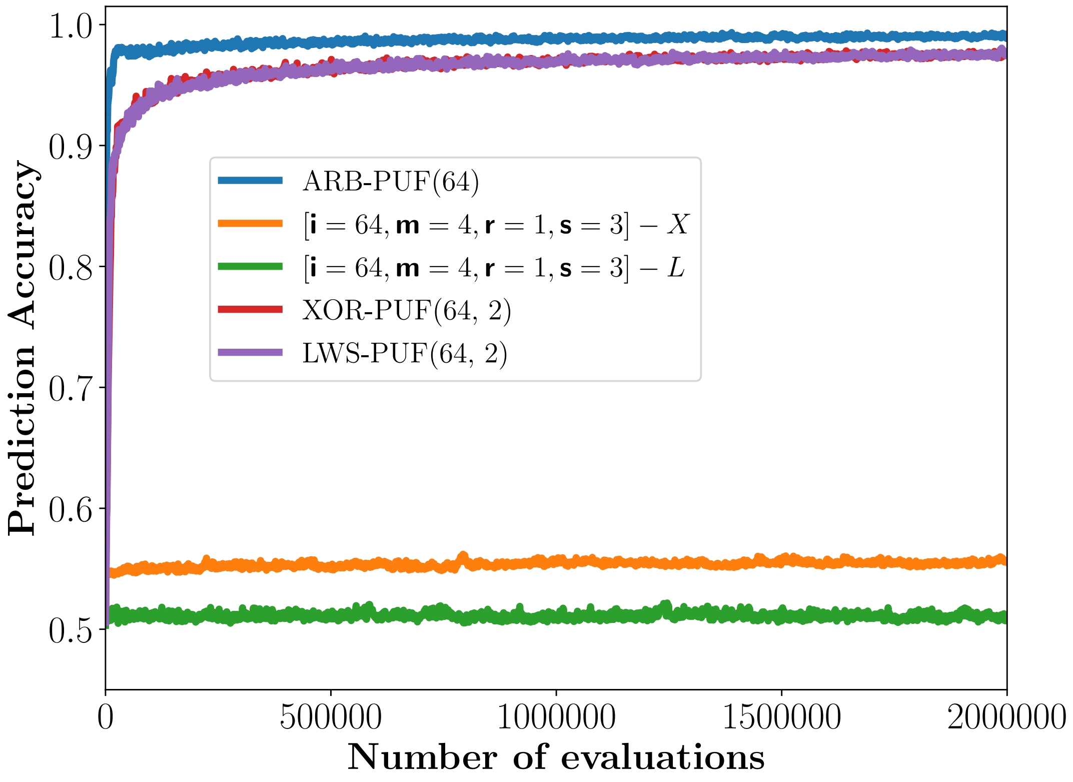


Figure 5.6: Prediction accuracy for $P \circ P$ composed of XOR-PUF and LWS-PUF as a function of evaluations.

Chapter 6

Conclusion

We have revisited the idea of strengthening PUFs by constructing PUFs that are compositions of other PUFs. Prior work has explored this idea, but the constructions there have not been promising. We have proposed a general model for composition, and considered a particular family in that model that admits new kinds of compositions, and also captures prior constructions. We have established analytically that even within this restricted family of constructions, there can exist PUFs whose strength, asymptotically, is the maximum possible for a particular input-size. We have then revisited state-of-the-art attacks on PUFs, and proposed an enhancement to the prior attack on such compositions. Via our extensive empirical assessments, we have confirmed that constructing PUFs by composition is indeed a promising approach to realizing strong PUFs. Our rather general model, and the manner in which PUFs are realized in practice, suggest a rich area of future work. A fuller exploration of other families in our model, for example, PUFs not limited to layers but also levels, is one possible futher work. Another perspective of the futher work may focus on the practical side of the PUF composition, for example, incorporation of the elements of practically realizable PUFs.

References

[1] http://www.pcp.in.tum.de/code/lr.zip, 2010.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, and Craig Citro et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[4] Blaise Gassend, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 148–160, New York, NY, USA, 2002. ACM.

[5] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Identification and authentication of integrated circuits: Research articles. *Concurr. Comput. : Pract. Exper.*, 16(11):1077–1098, September 2004.

[6] Chongyan Gu, Neil Hanley, and Máire O'neill. Improved reliability of fpga-based puf identification generator design. *ACM Trans. Reconfigurable Technol. Syst.*, 10(3):20:1–20:23, May 2017.

[7] C. Herder, M. Yu, F. Koushanfar, and S. Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, Aug 2014.

[8] Cliff Hou. A smart design paradigm for smart chips. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, feb 2017.

[9] Daihyun Lim. Extracting secret keys from integrated circuits. Master's thesis, MIT, 2004.

[10] Qingqing Ma, Chongyan Gu, Neil Hanley, Chenghua Wang, Weiqiang Liu, and Maire O'Neill. A machine learning attack resistant multi-PUF design on FPGA. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, jan 2018.

[11] Roel Maes. *Physically Unclonable Functions.* Springer Berlin Heidelberg, 2013.

[12] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure pufs. In *2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 670–673, Nov 2008.

[13] Sanu K. Mathew, Sudhir K. Satpathy, Mark A. Anders, Himanshu Kaul, Steven K. Hsu, Amit Agarwal, Gregory K. Chen, Rachael J. Parker, Ram K. Krishnamurthy, and Vivek De. A 0.19pj/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, feb 2014.

[14] D. Mukhopadhyay. Pufs as promising tools for security in internet of things. *IEEE Design Test*, 33(3):103–115, June 2016.

[15] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[16] Pappu Srinivasa Ravikanth. *Physical One-Way Functions.* PhD thesis, MIT, 2001.

[17] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[18] U. Rhrmair, J. Slter, 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. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, Nov 2013.

[19] D. P. Sahoo, P. H. Nguyen, D. Mukhopadhyay, and R. S. Chakraborty. A case of lightweight puf constructions: Cryptanalysis and machine learning attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1334–1343, Aug 2015.

[20] Durga Prasad Sahoo, Sayandeep Saha, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, and Hitesh Kapoor. Composite PUF: A new design paradigm for physically unclonable functions on FPGA. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*. IEEE, may 2014.

[21] Tom Schaul, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rckstie, and Jrgen Schmidhuber. Pybrain. *Journal of Machine Learning Research*, 11(Feb):743–746, 2010.

[22] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14, June 2007.

[23] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu. Machine learning resistant strong puf: Possible or a pipe dream? In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 19–24, May 2016.

[24] Zhuanhao Wu, Hiren Patel, Manoj Sachdev, and Mahesh V. Tripunitara. https://git.uwaterloo.ca/caesr-pub/pop-iccad-2019, 2019.