UMBC

DISSERTATION APPROVAL SHEET

Title of Dissertation: ON THE RESILIENCY OF PHYSICAL UNCLONABLE FUNCTIONS AGAINST POWER ANALYSIS ATTACKS

Name of Candidate: Trevor Kroeger

Doctor of Philosophy, 2022

Graduate Program: Computer Engineering

Dissertation and Abstract Approved:

Naghmeh Karimi

Naghmeh Karimi

Ph.D.

Computer Engineering

4/25/2022 | 10:14:35 AM EDT

NOTE: *The Approval Sheet with the original signature must accompany the thesis or dissertation. No terminal punctuation is to be used.

ABSTRACT

Title of dissertation: ON THE RESILIENCY OF PHYSICAL UNCLONABLE FUNCTIONS AGAINST POWER ANALYSIS ATTACKS

Trevor Anthony Paul Kroeger
Candidate of Doctor of Philosophy, 2022

Dissertation directed by: Professor Naghmeh Karimi
Department of Computer Science
and Electrical Engineering

Integrated Circuits (ICs) have made their way into many critical systems that service transportation, medical, and military industries; areas that are targeted for maximum disruption of operations and daily life. Whether the motive is monetary or political, it is clear that ICs require protection from malicious intent. To aid in the protection of devices various techniques have been developed to identify, authenticate, and track them. One of the principal security primitives used in the aforementioned techniques are Physical Unclonable Functions (PUFs). PUFs produce unique signatures based on the uncontrollable physical variations which occur during the fabrication of ICs. A PUF's output is produced when the PUF is given an input. Together these inputs and outputs are known as Challenge Response Pairs (CRPs). A PUF's CRPs are used for authenticating devices or for IC metering purposes, aiding in the prevention of over-production and IC cloning. The arbiter-PUF is one of the most popular PUFs broadly adopted by industry because of its large number of CRPs. Owing to their usefulness in securing ICs, PUFs are also the focus of attacks. They are vulnerable to modeling attacks in which the adversary tries to model the PUF's behavior to predict its response for unseen challenges.

There are two forms of modeling attacks: CRP-based modeling attacks and power-based modeling attacks.

When attacking a PUF using its power side-channel, the PUF response is predicted based on the PUF's power consumption. This research focuses on the power-based modeling attacks perpetrated against the arbiter-PUF family, and presents PUFs that are resilient against power-based attacks via inserting the proposed countermeasures. First, investigations are performed on the resiliency of the state-of-the-art PUFs that were proposed in literature recently to counter modeling attacks such as analog variants and challenge obfuscation based PUFs. These PUFs are shown to be successfully compromised through their power side-channel. These investigations are taken one step further by performing Cross-PUF attacks, where the power traces of one PUF can be used to model another PUF fabricated from the same GDSII file. This research shows, for the first time, that such attacks are highly successful in exposing a previously unexplored vulnerability of PUFs. Further investigations of power-based modeling attacks are performed by characterizing the effects that temperature and aging have on both Self-PUF and Cross-PUF attacks. Exploration of modeling the power is extended to multi-bit response parallel PUFs to show their vulnerability against power-based attacks. The results of these investigations showed that the response could still be discerned from the power consumption of the device. Because of the phenomenon being exploited in the power-based modeling attacks this research shows that these attacks work not only on the arbiter-PUF, but also its derivatives. To further improve the understanding of the various power-based modeling attacks, this research uses the Signal-to-Noise Ratio (SNR) to characterize and assess the vulnerability of the target PUFs to modeling attacks. Finally, to enhance the resiliency of the targeted PUFs against power-based modeling attacks, a number of circuit-level countermeasures, based on reducing the SNR and/or confusing the model, are proposed. These countermeasures appear to be highly successful in protecting the PUF against the power-based modeling attack. The results have been extracted first using HSpice simulations, and then the experiments (the attacks and countermeasures) were performed on FPGA fabric to verify the findings in silicon.

ON THE RESILIENCY OF PHYSICALLY UNCLONABLE FUNCTIONS AGAINST POWER ANALYSIS ATTACKS

by

Trevor Anthony Paul Kroeger

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland - Baltimore County in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Dr. Naghmeh Karimi, Chair/Advisor
Dr. Tinoosh Mohsenin
Dr. Chintan Patel
Dr. Ryan Robucci
Dr. Ke Huang

Dedication

for me

"I think I can, I think I can, I think I can... I thought I could, I thought I could."
– Arnold Munk, The Little Engine that Could

for erika

"There are some things you can't share without ending up liking each other, and knocking out a twelve-foot mountain troll is one of them."
– J.K. Rowling, Harry Potter and the Sorcerer's Stone

for conrad and greta

"Whether you think you can, or you think you can't–you're right."
– Henry Ford

"Risk something or forever sit with your dreams."
– Herb Brooks

Acknowledgments

I want to give my profound thanks to everyone who offered me advice and support. I know there are those who I will unfortunately overlook in acknowledging for which I apologize.

I will start by thanking my advisor, Dr. Naghmeh Karimi who took me on as her first PhD student. She provided the spark and insights to the research that got me through the process. All of my research was performed through collaboration with persons at Télécom Paris, and I would like to thank Wei Cheng, Jean-Luc Danger, and Sylvain Guilley for their help, insights, and assistance in performing this work. Next, I would like to thank my committee. Their flexibility and assistance helped in getting me into candidacy and through my final defense.

I would be remiss without mentioning all of the assistance from the Johns Hopkins University Applied Physics Laboratory which provided the funding and flexibility to obtain this degree while working full time. There were many persons at APL which provided support for which I am grateful.

I would like to thank my family especially my in-laws Ted and Dr. Rebecca Hochradel for the support that they gave me and my wife through this process. I know this would've been much harder without them.

To my children, Conrad and Greta, I'm grateful that they will never remember this and I'm thankful for the much-needed joy and life they provided in times when it was greatly needed.

Finally, this would've never happened without my incredible wife, Erika. She stuck with me though this season of life. She provided meaningful support, opportune guidance, and overall belief in me. I want to thank her for getting me across the finish line, and I know I should've trusted the awesomeness more.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CRP(s)	Challenge and Response Pair(s)
FPGA	Field Programmable Gate Array
IC	Integrated Circuit
PUF	Physical Unclonable Function
SVM	Support Vector Machine
IP	Intellectual Property
RFID	Radio Frequency Identifier
IoT	Internet of Things
DRL	Dual Rail Logic
RIL	Randomized Initialization Logic
DRILL	Combined DRL and RIL Countermeasures
RRM	Random Response Masking
RAS	Randomized Arbiter Swapping
XOR	Exclusive OR
S-R	Set - Reset
CMOS	Complimentary Metal-Oxide Semiconductor
HCI	Hot Carrier Injection
BTI	Bias Temperature Instability
NBTI	Negative Bias Temperature Instability
PBTI	Positive Bias Temperature Instability
ML	Machine Learning
VTC	Voltage Transfer Characteristic
SNR	Signal-to-Noise
ASIC	Application Specific Integrated Circuit
LUT	Lookup Table
GDSII	Graphic Design System II (File Type)

Chapter 1: Introduction

Confidentiality, integrity, and availability are three main factors that need to be considered when the security of a system is taken into account. Confidentiality is the capacity to keep data/information from being shared with third parties. Integrity is the adeptness to keep data from being manipulated without authorization. Availability is the ability to keep systems/data accessible for use. It is through these principles that the security of systems is evaluated [76].

The core tenets of security can be compromised with the distributed manufacturing of Integrated Circuits (ICs) and the integration of Intellectual Property (IP) into larger and larger designs. In the beginning, end-to-end IC production was performed by a single entity which designed, manufactured, tested, and distributed the ICs. This process, however, has changed; chips are now developed through a highly distributed manufacturing process where one company will design the IC (likely with the integration of IP from other companies), then be fabricated by another company, who will outsource testing to another company, and then be distributed by yet another company [66]. This push towards a more distributed manufacturing process is driven by the increasing complexity of ICs owing to the decrease of transistors' feature size. Indeed, with increasing the complexity of integrated circuits,

it is not cost-effective for companies to perform all the steps needed for design, testing, and fabrication of these circuitries locally. Therefore, the fabrication is performed in facilities that serve many fabless companies for an appropriate economy of scale [22]. With this distributed manufacturing process, it is easy to see how the aforementioned security principles can be compromised. Indeed, in any step of realizing a chip (design, manufacturing, etc.) there is an increase in the possibility of malicious activity [61] as can be seen in Fig. 1.1.

One of the reasons for an adversary to perform malicious actions on ICs is avarice [12]. It is easy to comprehend how much an entity has to gain in the means of profit from malevolent IC sales since semiconductors are projected to be a $460 billion industry in 2021, not including the markets that the industry feeds [72]. Other than monetary gains, an adversary's objective may be to compromise assets and access secure systems. There is increasing concern in vulnerable hardware being integrated into military systems. If they were successful, an adversary could gain access to sensitive/secret data or render defense systems inoperable jeopardizing national security [56, 61].

There are many actions an adversary can perform at each level of the production process (see Figure 1.2) including creating counterfeit devices through cloning the design, recycling ICs from used devices, shipping ICs that did not meet the testing specifications and would otherwise be discarded, and overproduction of the stated IC yield [38]. These actions may undermine the overall quality of the part and decrease the lifetime of the part causing early failure of the device [39], or result in financial loss for the designer in the case of overproduction or cloning. Devices

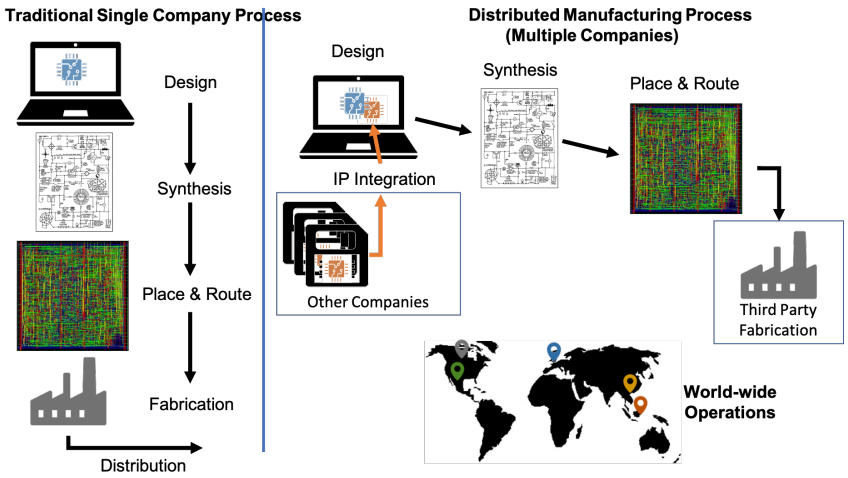


Figure 1.1: Distributed Manufacturing of Integrated Circuits

failing before they are expected has long-term effects on industry, government expenditure, and public health and safety [12]. These malicious actions call for provision of secure countermeasures for protecting ICs from adversaries.

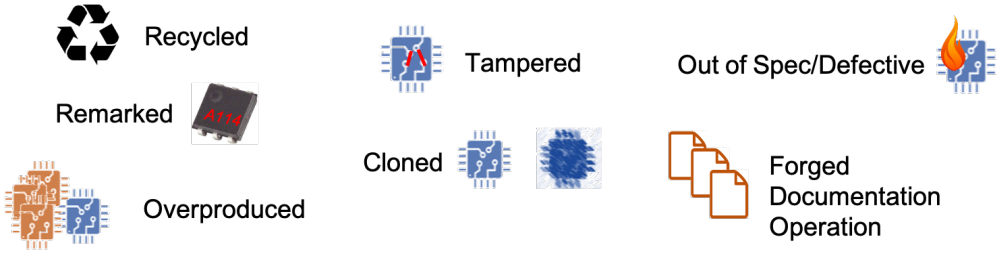


Figure 1.2: Types of Counterfeit Devices [38]

1.1 Problem Statement

Extensive research is being performed on securing ICs from the vulnerabilities inherent to the distributed design and manufacturing process [47]. The mechanisms employed include IC Metering, recycled IC detection and various techniques for unique device identification/authentication. The basis for many of these techniques

is a security primitive known as a Physically Unclonable Function or PUF [41, 48].

PUFs are incredibly useful devices that benefit from natural variations in the IC manufacturing process to produce unique signatures for each particular realized IC [32, 70]. The ability to produce these unique values make PUFs the hardware equivalent of one way functions; they were aptly proposed in 2002 almost simultaneously by Pappu et. al. [70] and Gassend et. al. [32] as Silicon Physical Random Functions. In most instances, a PUF is given a challenge for which it produces a response, together known as Challenge Response Pairs or CRPs. Since being proposed, PUFs have been integrated into a myriad of different devices for simple chip authentication like RFID cards [24] and Smartcards [28]. They are used in systems of systems, for instance devices in Internet of Things (IoT) networks [19] as well as more cutting edge technologies such as autonomous vehicles [8], and creating the basis of cryptocurrencies [64]. PUFs are increasingly being considered for "master key" generation. A number of companies have included PUFs in their portfolios, most notably one of the largest FPGA manufacturers, Xilinx, has begun integrating PUFs into their devices for consumer use [65]. Similarly, Microsemi has also integrated PUFs into their reprogrammable devices [79].

To diminish the security, PUFs themselves are natural targets for attackers and unfortunately the implementation of these primitives can themselves be compromised. The attacks that are of increasing concern are of the non-invasive nature. Modeling of a PUF through its CRPs is one method of compromising the PUF structures. For this attack, an adversary collects a number of CRPs for creating a model then uses that model to predict the PUF's response for the unseen chal-

lenges [30]. Another PUF cloning attack method (although a PUF is supposed to be unclonable) is through its Side-Channel characteristics, namely it's power. Similar to modeling through CRPs, the power consumption of the PUF is recorded during the PUF's operation, rather than its challenge and paired with the response. This power trace response pair is then used to train the model and is deployed to predict the response for unseen challenges [15, 57]. To be clear, in power-based modeling attacks, a machine learning model is trained by using the PUF's power traces and related response bits. This model is then utilized to infer the response of a PUF for previously unseen traces.

Previous literature indicates countermeasures for preventing CRP-based modeling attacks employing the use of fuses to prevent access to the critical data lines for the CRPs [83], employing challenge obfuscation techniques essentially encrypting the challenge before use [31, 60, 85, 91], or attempting to use analog PUF primitives [77, 78, 84].

Currently there is seemingly a lack of analysis of the PUFs' Side-Channel based Modeling vulnerabilities and countermeasures. It is important to state that the reason Power Side-Channel based modeling works is that the components create leakages within the circuit, as each component draws different levels of current in its operation. The models characterize the leakage from the components and the operation of the circuit can be discerned accordingly [16].

Another interesting topic that is performed in this research is the possibility of attacking one PUF using the power traces of another PUF manufactured from the same GDSII File. This attack, known as a Cross-PUF attack, is a newly discovered

vulnerability which severely undermines the security of PUFs. This threat makes the security primitive even more vulnerable since to target a PUF one can build the model from another PUF.

There exist inherent countermeasures such as limited physical access to ICs and the perception of the cost of resources to attack these devices. There are also seemingly simple implementation countermeasures, like increased PUF size, in both the challenge space and the amount of bits for the PUF response (i.e. multi-bit PUFs). Increasing the PUF size and creating multi-bit PUFs becomes costly in terms of the footprint required for the circuit and in terms of the power that is used in its operation, which might not be feasible for low-cost devices. This research involves an investigation of whether these countermeasures and assumptions are enough to preserve security of PUFs when faced with power-based modeling attacks.

1.2 Proposed Solutions

The focus of this research is on the resiliency of PUFs against power analysis attacks. This research includes the analysis of the power side-channel based modeling vulnerability of PUFs and developing appropriate countermeasures. There are two metrics that are important for predicting responses for power-based modeling attacks: the Signal-to-Noise Ratio (SNR) between the power and the observed noise and the distinguishability in the response (shown in Figure 1.3).

For a countermeasure to be successful, it must decrease the SNR of the collected power trace observing a response of a '0' or a '1' within the observed noise

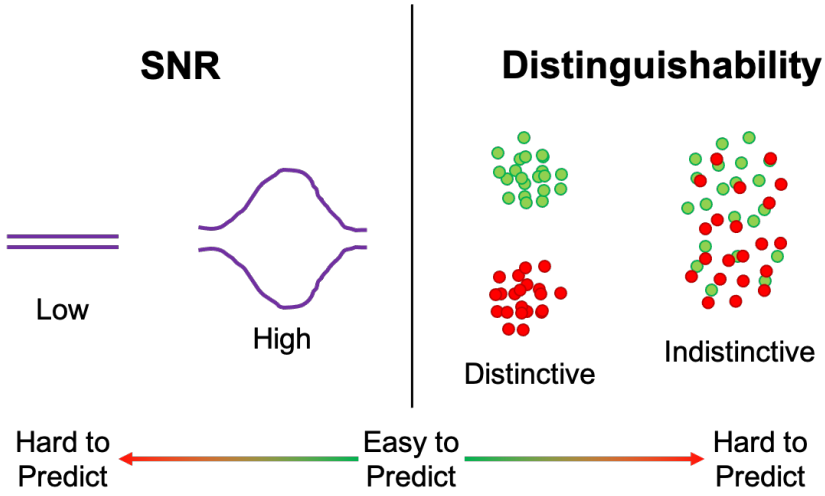


Figure 1.3: SNR and distinguishability metrics for PUF attackability. Countermeasures need to adjust these metrics as shown to reduce attack viability.

in the circuit. Decreasing the SNR insinuates that it will be difficult to predict the response via the model. Methods for doing this are through Dual Rail Logic (DRL) and Randomized Initialization Logic (RIL) mechanisms and will be presented in this research in combination as the DRILL countermeasure. DRL uses balanced Flip-Flops on the PUF output for the response and its inversion so that the combined power output is ideally uniform regardless of the response value. For RIL a Flip-Flop with an initialization port is used with the initialization port fed by a random value thereby providing more possible transition characterizations for the PUF output. Of course DRILL is the utilization of both techniques at the same time.

To decrease distinguishability, or the discrepancy between the power traces and the response values, one can add logic to confuse or poison the modeling algorithm. We develop two methods to achieve this: Random Response Masking (RRM) and Randomized Arbiter Swapping (RAS). Both of the proposed methods make use of

a random number generator. In RRM, an additional logic circuit is used to XOR the output from a random number generator and the PUF's response to randomize the output. RAS is used in the Multi-bit parallel PUFs for swapping the output bits generated from the PUF instances. The goal of both RAS and RRM is to create illogical responses for the PUF and poison the modeling algorithm. The randomization effects of both of these methods are reversed in software to utilize the correct response bits as the PUF response.

1.3 Contributions

The contributions of this research are as below:

- Show the vulnerability of PUFs (in particular the arbiter-PUF and its family) against power side-channel based modeling attacks;
- Compare the effectiveness of the power side-channel based modeling attacks and the CRP-based modeling attacks;
- Show the effect of environmental conditions (e..g, temperature) as well as aging on the power side-channel based modeling attacks;
- Show the ineffectiveness of countermeasures tailored against CRP-based modeling in preventing power-based modeling attacks and develop appropriate countermeasures against such attacks;
- Develop Cross-PUF attacks for the first time and show their efficiency in modeling the PUFs behavior;

- Investigate the effect of temperature and aging misalignments in the efficiency of the Cross-PUF attacks for both unprotected and protected variants;
- Launch successful power side-channel based modeling attacks on multi-bit parallel PUFs;
- Launch power-based attacks on the unprotected variants of single-PUFs, Cross-PUFs, and parallel PUFs realized in FPGA as well as their protected counterparts equipped with the proposed countermeasures, and show the success of the attacks in the unprotected implementations and the resiliency of the protected counterparts against power-based modeling attacks.

1.4 Research Outline

This research is composed of 9 chapters outlined in Figure 1.4. Chapter 1 is the Introduction. Chapter 2 contains relevant background material on the subject. Related work is discussed in Chapter 3. A compendium of Completed Research comprises Chapters 4 through 8 where Chapter 4 discusses Self-PUF attacks, Chapter 5 investigates attacks on multi-bit parallel response PUFs, Cross-PUF attacks are discussed in Chapter 6, the countermeasures to the attacks are reviewed in Chapter 7, and the effectiveness of the attacks and proposed countermeasures evaluated on FPGA fabrics are discussed in Chapter 8. Finally the Conclusion makes up Chapter 9.

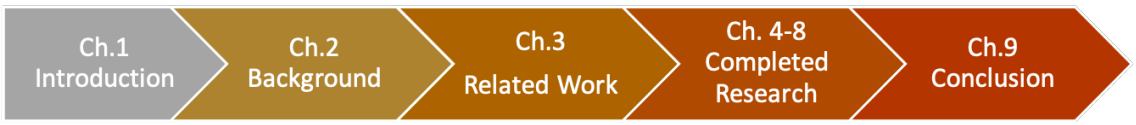


Figure 1.4: Pictorial Representation of Paper Outline

Chapter 2: Research Background

This chapter presents relevant background information to the work being presented. Pertinent background on PUFs is reviewed with a focus on the arbiter-PUF, a principal primitive used in this work. Machine Learning is also discussed since it is the basis for relevant modeling algorithms.

2.1 Physically Unclonable Functions

Physically Unclonable Functions, or PUFs as they are commonly called, are the hardware equivalent of a one-way function, i.e., they take an input and produce a unique output in a way that it is infeasible to determine the input from the output [32, 70]. Unlike the algorithmic one-way functions, which can be reproduced through the proper implementation of their algorithm, PUFs are unclonable which means that the same design will not produce the same output in another chip. A PUF's unclonablity is caused by the imperfections of the IC's manufacturing process resulting in random variations in the transistors specification. When an IC is manufactured, process variations are unavoidable and unpredictable. PUFs are carefully constructed to make use of these variations in a way to produce an output unique with respect to those variations [32, 70]. The security requirements of PUFs

are outlined in ISO/IEC 20897-1:2020 [42]. Equation 2.1 shows the function (f) of a PUF, it takes an input C, so-called challenge, and produces the response R. A challenge and its corresponding response is referred to as Challenge Response Pair (CRP). This function (f) is unique to each implementation (i) of a PUF such that none of their functions are the same. Although this equation seems very simple it is in fact enormously complicated.

$$R = f_i(C) \tag{2.1}$$

2.1.1 PUF Advantages and Applications

There are three key advantages of using PUFs in ICs. The first advantage is that they do not store sensitive information in memory, rather it is generated on-demand. If sensitive information is stored then an adversary can read said information out of the memory; this on-demand nature is important as it requires more effort for an attacker to gain access to the sensitive data. The second advantage is that since the sensitive information from a PUF is generated within the device itself, invasive attacks (those attacks that physically manipulated the IC) become much more difficult. Manipulation of the circuit may result in a change in the behavior of the PUF which will prevent the correct information from being generated and may destroy the circuit entirely. The third and final advantage is that PUFs require less hardware than traditional cryptography implementations (both can be used for device authentication) which reduces the space and cost of implementation [41].

PUFs take an active role in securing devices. To protect the devices, in a distributed manufacturing process, companies employ IC metering techniques where one of the key components used in these techniques are PUFs. IC metering, sometimes referred to as hardware metering, is the methodology employed by designers to enable the tracking, identification, and/or control of individual chips [47]. Figure 2.1 shows the various different categories related to IC metering with the types closely associated with PUFs highlighted. In fact, PUFs can play a role in all types of IC metering with the exception for passive reproducible nonfunctional identification techniques which consist of assigning a chip an ID value (e.g., a serial number) that is imprinted on the device or stored accessibly in the chip.

For passive unclonable types of IC metering, PUFs are used quite naturally, producing values in which an individual device can be verified. It is easy to extrapolate how IC overproduction is prevented by using PUFs. This protection is provided by the fact that if the unique identifier provided by a PUF, embedded in a non-legitimate IC, is being validated against the registry of devices held by the designer/distributor, it is easily flagged [47].

In active IC metering methodologies, PUFs are used more creatively. A common method for active internally controlled types is through state transition obfuscation, which utilizes a PUF to initialize a state machine. When implemented, the device starts in a NULL state (initialized by the PUF) where the designer needs to verify the operation and provide the proper state transitions to get the IC into a functioning state [10]. PUFs can also be used for locking (obfuscating) an IC so that each IC requires a unique identifier (or key) specifically generated for each individual

chip. This key is used to unlock the internal portions of the chip responsible for its proper functionality [47].

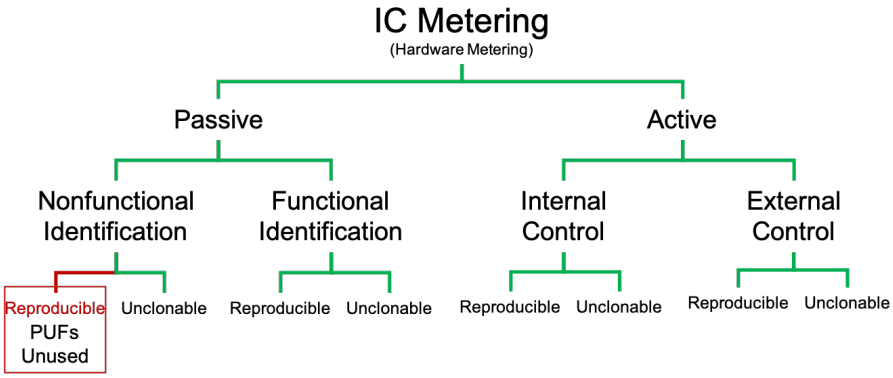


Figure 2.1: IC metering categorization with PUF aided categories highlighted [47].

Other than IC metering, PUFs are used mainly for authentication purposes or for generating keys for cryptographic modules. For example, RFIDs and Smartcards make use of PUFs for authentication. RFIDs and Smartcards are very simple unpowered devices that are used to authenticate transactions. This authentication could be as simple as providing building access or as important as providing access to bank accounts. PUFs are well-suited for this purpose since they have a lightweight design, are cheap to manufacture, and quickly produce their unique values [94].

Thanks to their lost-cost, a PUF can also be used to produce values that can be employed for secure communications either between devices or between the device and a server. It can be also used for device authentication, in particular for Internet of Thing (IoT) devices as they are low cost primitives [27, 55, 88]. An example of device authentication in a network of systems can be seen in the research presented by Chatterjee et. al. [19]. Here, the authors create a distributed public-private key exchange methodology that utilizes a distribution of verifier and prover nodes to

provide verification of node to node communications using PUFs.

To summarize, PUFs find their usefulness in two primary functions: key generation and authentication [41].

2.1.2 PUF Evaluation Metrics

According to the PUF's related standards, these primitives should meet the following requirements:

- Randomness - The indication that the response bits conform to a uniform distribution i.e. how random are the output bits.
- Uniqueness - There is significant variation across implementations of the PUF such that they are distinguishable from one another.
- Uniformity - There is an equal balance of '0's and '1's in the response set.
- Reliability/Stability - The response to the same challenge is not changed by the change of environmental conditions (e.g. temperature) or over time (due to device aging mechanisms).

To be able to use a PUF in authentication and other purposes, it is important to ensure that the above metrics follow the requirements [17, 81, 89]. Specifically the stated requirements for randomness for a PUF are evaluated using the benchmarks provided by NIST in Bassham et. al. [14].

2.1.3 Weak and Strong PUFs

There are two broad categories of PUFs: Weak and Strong PUFs. The distinction between Weak and Strong PUFs has to do with their challenge space. Weak PUFs have a small challenge space containing few or sometimes a single challenge. Strong PUFs have an exponentially large challenge space [41]. Another difference between the two is the number of response bits generated from the PUF. Typically Weak PUFs produce many response bits per challenge and Strong PUFs produce only a few. There exists a dichotomy between the number of challenges and responses; as the challenge space increases the amount of responses that are yielded from the PUF decreases and so the PUF transitions from being a Weak PUF to a Strong PUF. The general conservation rule for the CRP duality is:

$$\#\text{challenges} * \#\text{responses} = 2^n$$

or to represent this adequately in terms of n:

$$n = log_2(\#\text{challenge}) + log_2(\#\text{responses})$$

This equation makes visualization of the relationship easier to comprehend and is shown in Figure 2.2 [17].

Communities of practice use entropy to standardize the metrics by which PUFs are evaluated. Entropy for a PUF relates to its predictably. In relation to the previous discussions on duality, the predictability of PUFs with a small challenge space (i.e., Weak PUFs) are relatively high, meaning that these sorts of PUFs have a low entropy. The SRAM PUF, for example, takes a single challenge (the request for

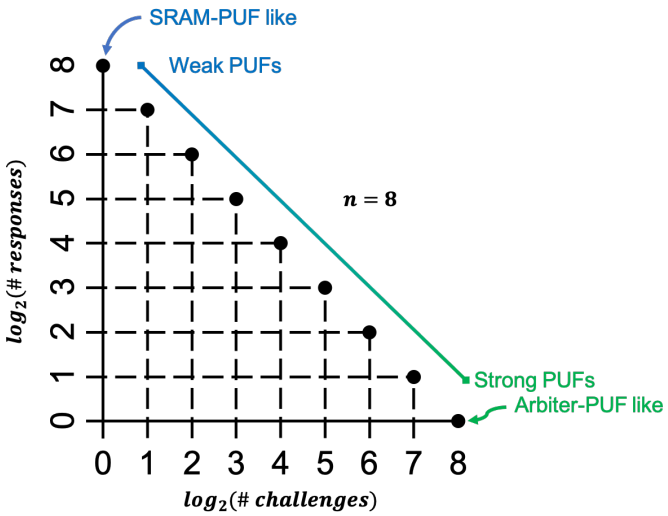


Figure 2.2: CRP size tradeoff and transition from Weak PUFs to/from Strong PUFs [17].

the response) and produces its response. Conversely, a PUF with a large challenge space (i.e., Strong PUFs) has a high entropy, with very low predictability, because its challenge space is exponential. The arbiter-PUF displays this exponential challenge space and it is generally considered to be infeasible to check the responses of all the challenges [17].

Weak PUFs are primarily used for key generation. Accordingly, they are also commonly known as Physically Obfuscated Keys (POKs). Two types of Weak PUFs that are mainly used are memory-based PUFs (i.e. SRAM PUFs), and Ring-Oscillator variants (as shown in Figure 2.3) [43]. Since their challenge space is so small their responses must not be disclosed. The key advantage of Weak PUFs is secure key storage, where the key is generated reliably by the PUF and not stored within a memory structure and therefore more difficult to extract [41].

The other category of PUFs are Strong PUFs which are useful for identification and authentication purposes (as well as key generation) [41]. Strong PUFs benefit

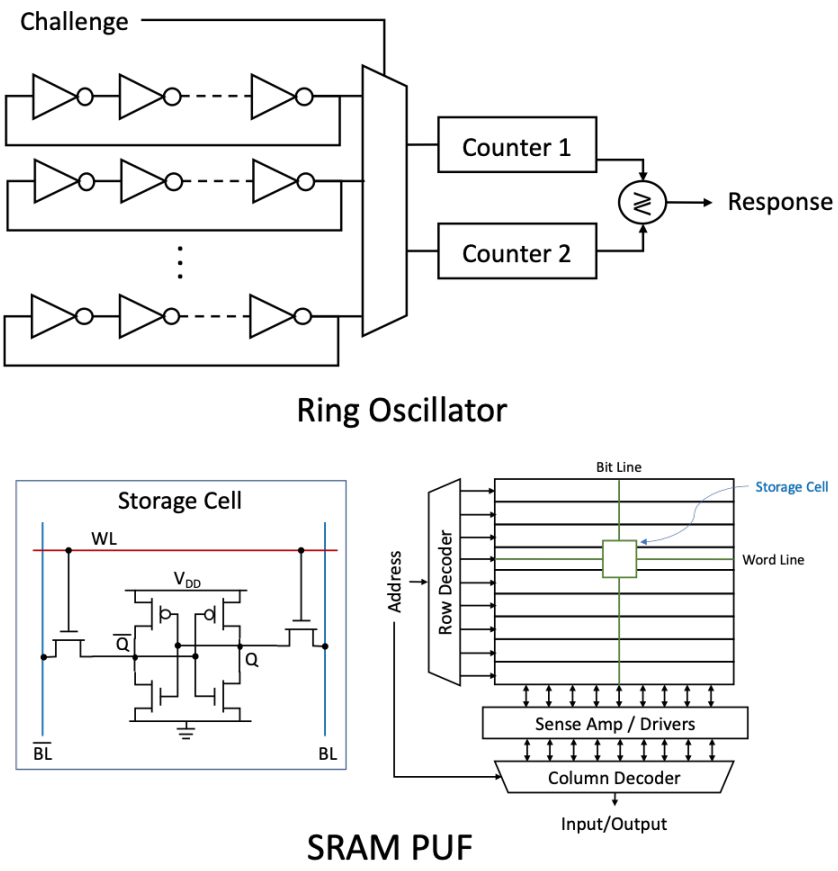


Figure 2.3: The SRAM PUF [36, 71] and Ring-Oscillator PUF [63]; examples of Weak PUFs.

from having a vast challenge space and therefore do not need to have complete secrecy in their CRP generation. Types of Strong PUFs (as shown in Figure 2.4) include physical structures (i.e., Optical PUFs [70]), and delay-based PUFs such as the arbiter-PUF and its derivatives (XOR-PUF, Interpose PUF, Feed Forward PUF), and loop-variant PUFs (Loop PUF, Bistable Ring PUF) [7]. To increase the reliability of the PUF, error correction techniques may need to be employed. This error correction may take place in extra implementation logic within the device or, depending on the application, error correction processes can occur off of the device in which the PUF is implemented [41].

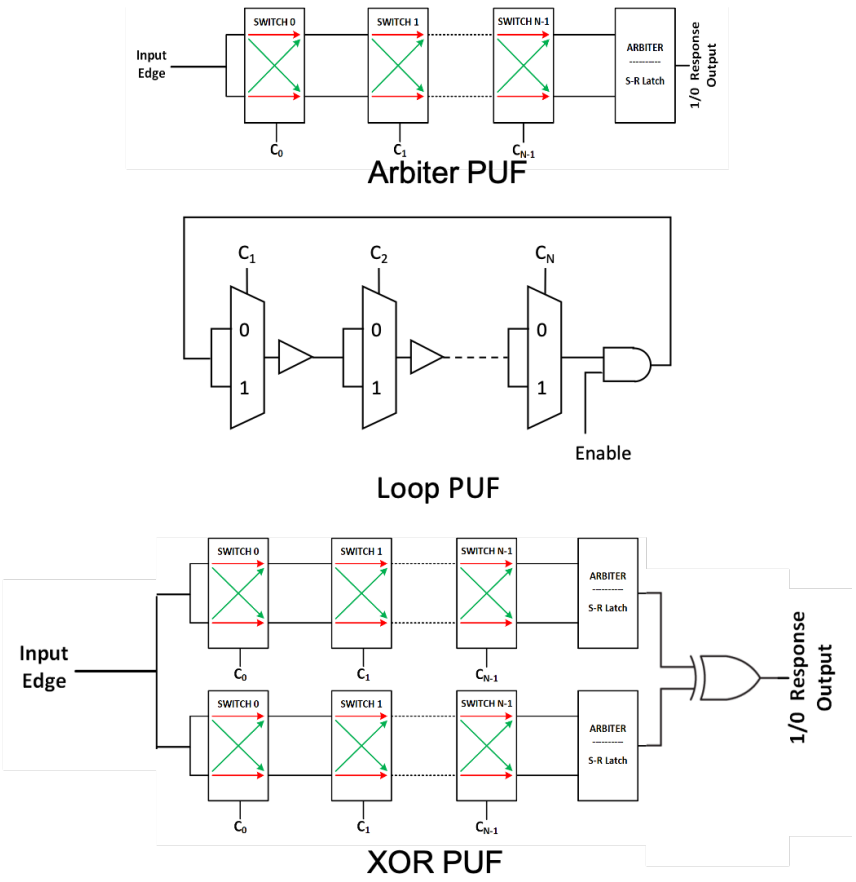


Figure 2.4: Arbiter-PUF [32], Loop-PUF [20], and XOR-PUF [93] examples of Strong PUFs.

2.1.4 Arbiter-PUF Family

The arbiter-PUF is one of the most popular PUFs because of its small size and large CRP range [15]. It is a delay-based PUF that forms the foundation of various other PUF variants such as the XOR-PUF, Feed-Forward PUF, and Interpose PUF (among others). Other reasons for its widespread adoption can be attributed to the relative simplicity in its implementation and its effectiveness in producing unique values. It utilizes a large challenge space, in fact the total number of challenges is 2^n (where n is the number of stages it includes) which makes it attractive for

implementation (note that this makes it a Strong PUF). Its structure was first proposed by Gassend et al. [32] as a Silicon Random Function; its design was one of the first PUFs proposed. The structure of the arbiter-PUF is shown in Figure 2.4.

The arbiter-PUF has a single input trigger which feeds a series of connected switches leading to an arbitration component. The top and bottom inputs to the initial switch are connected to the input trigger. The following switches are connected with each of the top outputs and each of the bottom outputs connected to the corresponding lines on the inputs to the next switch, creating a top and bottom path through the entire structure. Each switch is composed of two multiplexers which select the appropriate top and bottom lines to pass through to correct output. Each bit of the challenge is provided to one of the switches (these are the select lines to the internal multiplexers), if the challenge bit is a '0' then the top and bottom paths are not switched, however if the challenge is a '1' the paths are swapped. The switch component of the arbiter-PUF is shown in Figure 2.5.

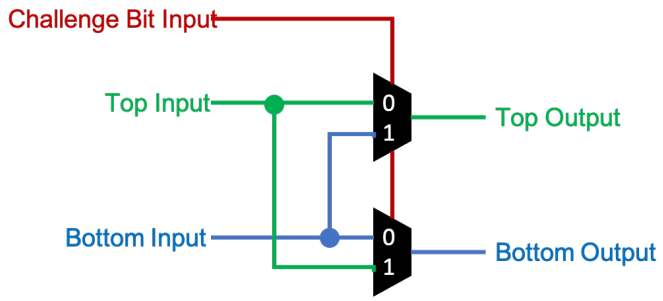


Figure 2.5: Internal structure of the PUF switch comprised of two multiplexers selecting the opposite inputs.

At the end of the switching chain the arbiter takes both the top and bottom paths as inputs and determines the final PUF response which is then sent to the

system components which include a register for storing the result. The arbiter can be realized with an S-R latch with the S and R inputs connected to the top and bottom paths, or a Flip-Flop with the input connected to the top and the bottom connected to the clock. During its operation, the arbiter-PUF is queried with a transition on input trigger *after the challenge is provided to the switches.* This edge propagates through all of the switches compounding the delays of the individual paths. The arbiter then utilizes the edge arrivals from the top and bottom paths to determine if the response is a 0 or a 1. In this sense, it is the sign of the delay difference between the top and bottom paths that builds the response of the PUF.

Figure 2.6 shows the logic tables for the S-R Latch and Flip-Flop to support how the top and bottom paths are adjudicated. If, for the S-R Latch, the top path is faster, the S (set input), arrives first there is a '0' response. If the bottom path is faster, then the R (reset input) is triggered first resulting in a '1' response. This device works because when both of the inputs, S and R, are '1's they hold the current value on the output of the latch. For the Flip-Flop, both paths are resting at '0'. If the top path is faster then a '1' is waiting at the Flip-Flop input when the clock input sees the rising edge of the bottom path. Otherwise if the top path is slower then the input will be a '0' when the rising edge is seen on the bottom path [32].

The response of the arbiter-PUF is determined by the sign of the summation of the delay differences in the top and bottom paths for each challenge. The delays of each of the stage i can be represented by the difference of $\delta_{1,i}$ for when the challenge is '1' and $\delta_{0,i}$ for when the challenge is '0'. The challenge is broken down to its bit components where c_i is the bit for each stage. The total delay, ΔD_i,

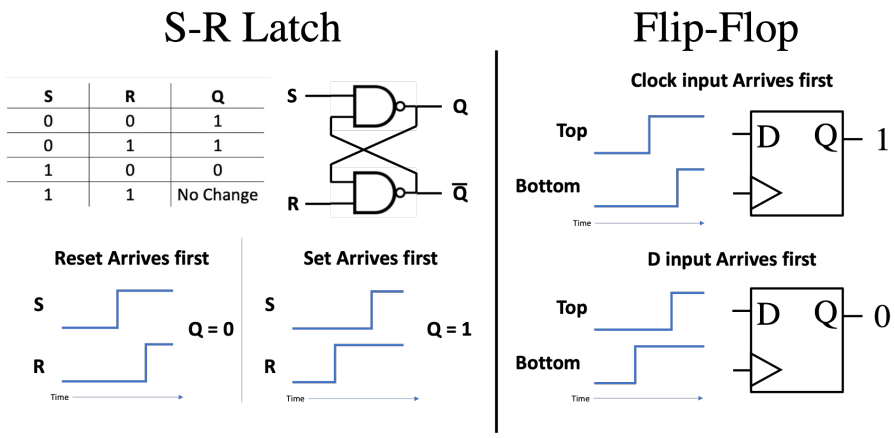


Figure 2.6: Logic tables for the S-R Latch [17] and D Flip-Flop with explanations of arbitration outputs for responses.

can be represented by the cumulative delay in previous stages ΔD_{i-1}, switched or unswitched based on the challenge bit (c_i) for the current stage (i), with the current stage delay added $\delta_{c_i,i}$. The total delay difference is shown in Equation 2.2.

$$\Delta D_i = \Delta D_{i-1} * (1 - 2 * c_i) + \delta_{c_i,i} \tag{2.2}$$

Recall that ultimately the response (r) of a n bit challenge arbiter-PUF is just the sign of the cumulative delay:

$$r = sign(\Delta D_n) \tag{2.3}$$

These equations are integral to the understanding of how the arbiter-PUF is modeled [15].

2.2 Effect of Temperature Change in PUFs

Temperature affects device operation. In fact, manipulation of temperature can be used to perform attacks on integrated circuits. In Kumar et al. [52] they use temperature manipulation to perform fault attacks on cryptographic algorithms. If a significant temperature change is experienced, the circuit can operate outside of its intended bounds. For instance in an arbiter-PUF, the temperature change can affect the race of the top and bottom paths resulting in a change in the response. In other words, if the difference between the top and bottom race is small, then with a slight temperature change, the race will change flipping the response bit. Gassend et al. studied the effects of temperature on PUFs in 2003 showing that with the temperature change as low as $10°C$ the PUF response can be changed [33].

2.3 The Impact of Device Aging on PUFs

Aging has a non-negligible effect on the performance and operation of integrated circuits, including PUFs. As a device is aged, it deteriorates. Specifically for Complementary Metal Oxide Semiconductor (CMOS) technology, the major effects of aging occur based on two primary factors: Hot Carrier Injection (HCI) and Bias Temperature-Instability (BTI) [53, 69].

There are two types of BTI: positive (PBTI) and Negative (NBTI). Although the effects from both types occur in devices, the main contribution to degradation comes from the NBTI's effect on PMOS transistors. NBTI has two separate effect

phases: the stress phase and the recovery phase. The transistor is in the stress phase when the threshold voltage exceeds the gate-source voltage. When this phase occurs the threshold voltage increases due to a rise in the positive interface traps developing in the Silicon-Silicon Dioxide interface. The effect is partially recoverable when the threshold voltage is below the gate-source voltage (i.e., when the transistor turns off) which is when it is in the recovery phase. Continued stressing of the transistor eventually leads to threshold voltage drifting overtime, although the amount depends on the transistor's supply voltage, temperature, and the amount of time the transistor is in the stress phase [46].

Hot Carrier Injection (HCI) primarily affects NMOS transistors. A hot carrier is an electron which gets stuck in the dielectric in the gate of a transistor and occurs due to the repetitive switching of a transistor on and off [37]. The HCI causes changes in the threshold voltage and the operating current. The effects of HCI are permanent and cannot be reversed [92]. This research considers the effect of both BTI and HCI.

Similar to temperature, aging is discussed by Gassend et al. [33], not in detail but rather as a note of future research that *must* be performed. Since the aforementioned suggestion, experiments have been run to assess the effects of aging on PUF circuits. Karimi et al. [44, 45] showed that aging highly affects the reliability of PUF circuits; thus analysis of aging should be done early in the design process so that circuitry can be added to mitigate its effects. Experiments, in both simulation and in real silicon, show that there is an increase in the bit error rate of the PUF circuit over time. Specifically observing the arbiter PUF it is shown that the aging

affects the arbitration component (the S-R latch responsible for adjudicating the response) more than the delay components. It is hypothesized that this is due to the asymmetric operation that occurs on the latch. Recall that the degradation effects of NTBI occur when the PMOS transistor is in stress, when the gate-source voltage is greater than the threshold voltage in the PMOS transistor [44, 45].

2.4 Targeting PUFs with Machine Learning Models

Although a PUF is supposed to be unclonable, as the name states, they can in fact be cloned via Machine Learning (ML) schemes [30]. Supervised ML algorithms require the model to be trained with data to determine the parameters of the model. The model is then used to classify new data. The general process for developing and testing the model is shown in Figure 2.7. First, the data is separated into a set for training and a set for testing. Next, the training data is used to build a model. This step is known as the training phase. Finally, in the evaluation phase the model is used to predict the response from the data in the testing data set. Comparing the number of correctly predicted responses to the total number of test responses gives the accuracy of the model. If the accuracy is in an acceptable range then the model is ready to be deployed [13].

In unsupervised learning, the data is unlabeled and the algorithm attempts to group said data into logical categories. When new data is added to the system, the learning algorithm updates and attempts to place the data into one of the logical categories. There are a number of ways to constrain these type of algorithms,

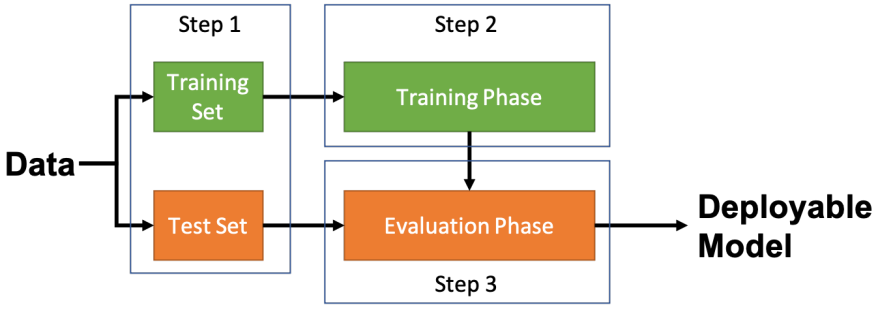


Figure 2.7: Flowchart for the general operation of the supervised learning algorithms.

including but not limited to the following: specifying the number of expected categories; limiting the number of points in a category; and/or specifying the closeness of data points to one another [13].

2.4.1 Support Vector Machine

Support Vector Machine (SVM) is a supervised learning method which is effective in modeling the arbiter-PUF [30]. The SVM scheme classifies the data by describing a hyper-plane. If the data is on one side of the plane then it is classified as one thing and if it's on the other side then it is classified as something else. If the data is not linearly separable then there are two options: allow for an acceptable amount of error or use a kernel function to manipulate the hyperplane to a non-linear space [90].

The standard equation for an SVM hyperplane is shown in Equation 2.4.

$$w^T x - b = 0 \tag{2.4}$$

Classifying items based on this hyper-plane is rather simple. In Equation 2.4,

x represents the data point to be classified and the result is that data point's classification.

$$w^T x_1 - b = 1$$
$$w^T x_0 - b = -1 \tag{2.5}$$

In Equation 2.5 it can be observed that x_1 is classified as a 1 and x_0 is classified as -1. An example hyper-plane from Equation 2.4 is shown in Figure 2.8, with examples of x_1 and x_0.

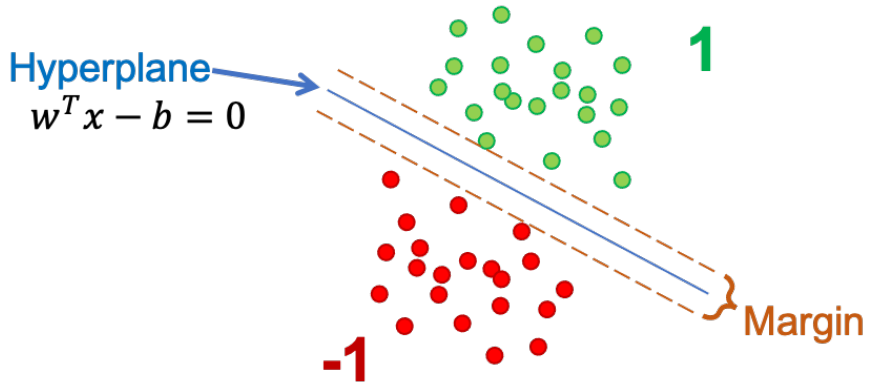


Figure 2.8: An example SVM hyper-plane with classified points [21].

In the training phase, the training set is used to build the model; for PUFs the training data are the challenges (or power traces) and the labels are the response values for those challenges. When training the model, the goal of the algorithm is to maximize the margin (shown in Figure 2.8) between the labeled data while correctly classifying each of the data elements [21].

Chapter 3: Related Work

A principle concept of this research is investigating the resiliency of PUFs, and in particular, the known arbiter-PUFs family against the modeling attacks and improving their resiliency against such attacks. The methods utilized in modeling are Machine Learning (ML) algorithms which have been extensively paired with Hardware Security [9]. Moreover ML applications have been found to be both beneficial and detrimental in terms of PUF security [26, 30, 54]. The two main avenues for utilizing ML schemes to a model PUF are through its CRPs as well as its power consumption.

3.1 Challenge-Response Pair based Modeling

In CRP-based modeling, the primary goal is to learn the mechanism by which the PUF manipulates the challenge to get the response. Recall in Equation 2.1, CRP Modeling attempts to discern the function (f) in this equation [75]. The ability to model a PUF fundamentally compromises its security as it clearly violates a PUF's unclonablity. For weak PUFs the model can be as simple as storage of the PUF response, since weak PUFs have a limited set of CRPs, but for strong PUFs, such as the arbiter-PUF, the modeling is more complicated which is where

ML methodologies come into play [75].

To model a PUF's function, a set of CRPs must be collected. An adversary could collect said CRPs from the target device when it is still in its enrollment phase, before the PUF is locked and access to those lines is restricted through anti-fuses. The enrollment phase is the time when the manufacturer, developer, or distributor creates a list of CRPs to legitimately use the PUF at a later time. This attack requires the attacker to have physical access to the device at a specific stage in its fabrication [83], or it can happen when the adversary eavesdrop on the communication lines between the verifier (say server) and the device including the PUF (i.e., prover) in IoT frameworks or other communication networks. To remove the specificity in the timing of the physical attack, an adversary can develop the skills to invasively extract the CRPs from the target device. In this style of attack, an attacker physically infiltrates the device to gain access to the traces on the IC corresponding to the challenge and response. This process is very precise as the attacker must be careful to not change the functionality of the actual PUF circuit which can inadvertently occur if the physical structure of the PUF is altered [30]. Another method of obtaining a device's CRPs occurs through eavesdropping. In this method an adversary lies in wait collecting CRPs from the device as it sends them out during normal operation. Eventually the adversary will collect enough CRPs to build the model of the device and carry out their intended attack [74]. Other methods for obtaining a set of a device's CRPs can occur from situations besides accessing the device itself. For instance, gaining access to the master list of CRPs from the manufacturer, developer, or distributor through insider threats or

traditional cyber attack methods.

Regardless of the method of collection, a set of CRPs is required to create a model of the device. The actual number of CRPs needed depends on the size of the PUF (how many bits comprise its challenge); the larger the PUF, the more CRPs are required to make a robust model. As stated in Rührmair et al. [74], to achieve an accuracy of 99% in a 64-bit arbiter-PUF, 2555 CRPs are needed and for a 128-bit arbiter-PUF, 5570 CRPs are needed for the same level of accuracy. The tradeoff for the increased security from a larger PUF size is an increase in the overall footprint and the time required to query the PUF [74].

Previously, the reliability of the arbiter-PUF has been attacked through response repeatability exploitation. By such repetition and extracting the challenges that result in unstable responses, timing information of the PUF stages are extracted and the related PUF eventually is modeled [23]. Recently researchers proposed the Interpose PUF, an arbiter-PUF variant which used an intermediate arbitration chain to add a random bit to the challenge of another arbiter-PUF. They aimed at making the targeted PUF resilient to CRP-based modeling attacks [67]. However, this structure has since been thwarted using ML techniques [7, 18, 87]. This displays the evolving landscape of PUF vulnerability research.

CRP-based modeling of a PUF is a well known vulnerability therefore researchers have created various countermeasures to thwart this type of modeling. The countermeasures specifically discussed here are the use of analog PUF variants or deploying Challenge Obfuscation schemes. Note that using obfuscated challenges prevent the adversary from having access to the real CRPs related to the deployed

PUF and modeling it through its CRPs. Similarly, the methods that encrypt responses also prevent the leakage of CRPs to the adversary [82, 94]. This research shows that although such countermeasures make the PUFs secure against CRP-based attacks, they cannot tackle power-based attacks.

3.1.1 CRP-based Modeling of the Arbiter PUF

Before countermeasures are discussed it is important to understand how CRPs are modeled. First, individual PUFs may require a transformation for their CRPs to be easily modeled as is the case for the arbiter-PUF. Recall in the total delay equation (Equation 2.2), the difference of the delay of paths selected based on a given challenge is the sum of all the delays of the switches for each of the switching components based on their individual challenge bits. The total delay difference can also be rewritten as:

$$\Delta D_n = \overrightarrow{w}^T \overrightarrow{\phi} \tag{3.1}$$

In Equation 3.1, the delay components have been assembled into $\overrightarrow{w}$ and the challenge components into $\overrightarrow{\phi}$. The lengths of $\overrightarrow{w}$ and $\overrightarrow{\phi}$ are equal to $n+1$ (n is the length of the challenge as well as the number of switches in the delay chain). Here $\overrightarrow{\phi}$ can be found based on Equation 3.2.

$$\begin{gathered} \phi_i = \prod_{l=i}^{n} (1 - 2c_l) \text{for} 1 \leq i \leq n \\ \phi_{n+1} = 1 \end{gathered} \tag{3.2}$$

The $\overrightarrow{w}$ is composed of the differential delay in each stage as well as components of the delay in the previous stage to track the inversion due to switching.

$$\begin{aligned} w_1 &= \delta_{0,1} - \delta 1, 1 \\ w_i &= \delta_{0,i-1} + \delta_{1,i-1} + \delta_{0,i} - \delta_{1,i} \\ w_{n+1} &= \delta_{0,n} - \delta 1, n \end{aligned} \tag{3.3}$$

When expressing the total delay for the arbiter-PUF in the form of Equation 3.1 it is easy to see that this conforms to the hyperplane definition of a SVM (Equation 2.4). Hence SVM can be a good candidate for modeling arbiter-PUFs using its CRPs [15, 75]. Please note that this does not exclude the efficiency of some other ML schemes for modeling the PUF as well.

3.1.2 Analog PUF Variants

One technique to thwart the CRP-based modeling of a PUF is to use analog PUF variants where the response is generated from a space that is continuous (e.g., $0\,V$ to $1\,V$) rather than the discrete points (e.g., '0' or '1') in the digital realm. Here, the randomness is introduced by having a continuous operating space. There are several PUFs developed to operate in the analog domain [77, 78, 84].

One analog variant is the Voltage Transfer Characteristic (VTC) PUF which was proposed to address the arbiter-PUF's vulnerability against CRP-based modeling attacks [84]. As shown in Figure 3.1, its similarity to arbiter-PUF is apparent.

The VTC PUF is composed of a series of switching components equal to the

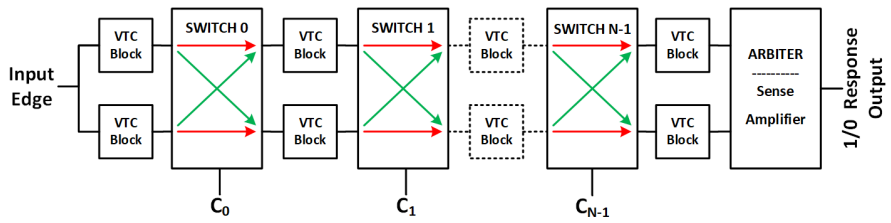


Figure 3.1: Voltage Transfer Characteristic PUF switching chain; note its semblance to an arbiter-PUF [84].

number of challenge bits. Before each of these switching components is a VTC block which non-linearly translates the voltage. The circuitry and function of the VTC block can be seen in Figure 3.2.

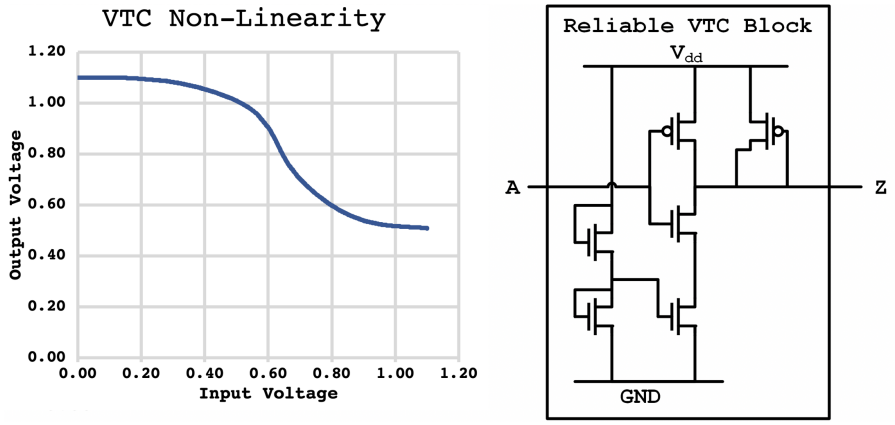


Figure 3.2: VTC block circuit diagram and non-linearity translation graph [84].

The VTC PUF is fed with an input edge of $Vdd/2$, which puts the voltage translation from the VTC block in the area of largest variance and non-linearity. The switches are Transmission Gate based analog switches transferring the voltage level rather than the edge propagation The circuitry for this is shown in Figure 3.3.

Finally the analog voltage levels, of the top and bottom traces, are compared to one another in a sense amplifier to determine which is greater. The main difference between the VTC PUF and the arbiter-PUF is its operation in the analog

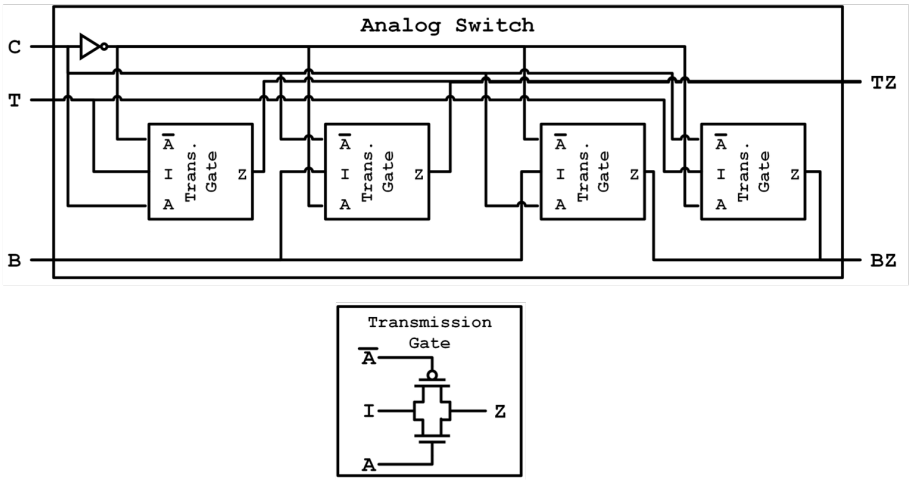


Figure 3.3: Circuit diagram for the analog switch used in the VTC PUF [84].

domain, performing voltage translations and eventual voltage comparison rather than comparing the sign of the path delay [84].

3.1.3 Challenge-Obfuscated PUFs

Another method for countering CRP-based modeling attacks is through challenge obfuscation. Challenge obfuscation is the art of methodologically manipulating the input challenge such that it becomes something different. In this case, the challenge that is passed to the device is not necessarily the challenge that is being provided to the PUF. In general, the challenge obfuscation technique is an addition to a standard PUF implementation. As an example, challenge obfuscation circuitry can be added to the incoming challenge bits of the arbiter-PUF as shown in Figure 3.4. The desired goal of challenge obfuscation is to provide an unknown challenge to the PUF, the result is that the CRP seen by the attacker will be incorrect and will confuse or poison the attacker's CRP set [31, 60, 82, 85, 91, 94].

Zalivaka et al. [91] present a comprehensive PUF system whereby one compo-

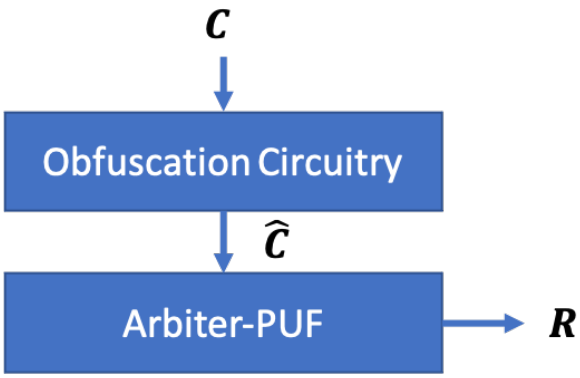


Figure 3.4: Generic challenge obfuscation technique implemented for an arbiter-PUF.

nent is a circuit for challenge obfuscation that modifies the challenge before applying it to the arbiter-PUF. Here the input challenge (C) is adjusted with a programmable nonce (α) to produce a new input challenge ($\hat{C}$) which can be expressed as shown in Equation 3.4 and the corresponding circuit implementation is shown in Figure 3.5. Note that the effects of challenge obfuscation are cascading, meaning that the resulting obfuscated challenge ($\hat{C}(t)$) is dependent on the previously generated obfuscated challenge ($\hat{C}(t-1)$).

$$\begin{bmatrix} \hat{C}_0(t) \\ \hat{C}_1(t) \\ \vdots \\ \hat{C}_{N-1}(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \cdots & \alpha_0 \\ 0 & 1 & 0 & \cdots & \alpha_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & \alpha_{N-1} \end{bmatrix} \times \begin{bmatrix} \hat{C}_0(t-1) \\ \hat{C}_1(t-1) \\ \vdots \\ \hat{C}_{N-1}(t-1) \end{bmatrix} \oplus \begin{bmatrix} C_0(t) \\ C_1(t) \\ \vdots \\ C_{N-1}(t) \end{bmatrix} \tag{3.4}$$

When employed in the protection of a circuit, the relation between challenge and responses becomes more complex than they were in their original form as they

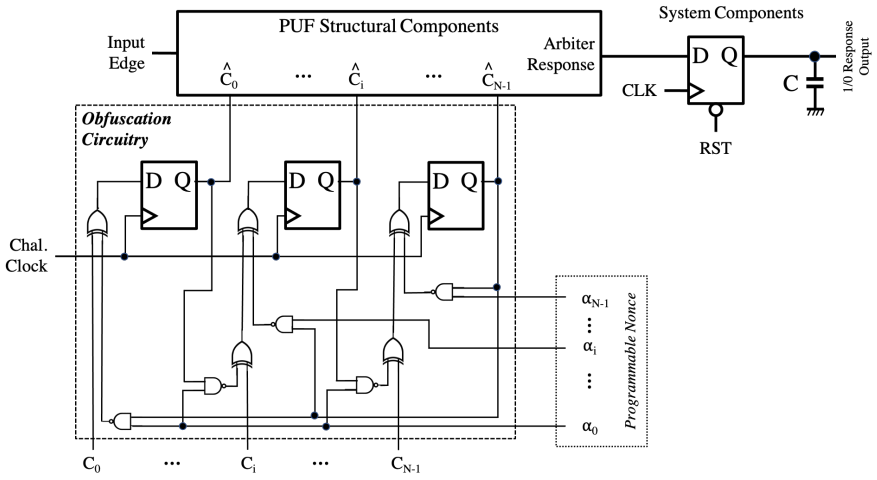


Figure 3.5: Block diagram for the implemented Challenge Obfuscation technique presented in [91] and replicated here.

are now a product of the input challenge (C), a nonce (α), and the previously generated obfuscated challenge ($(\hat{C}(t-1))$). This CRP generation technique increases the modeling complexity for would-be CRP-based attackers.

3.1.4 Other Countermeasures against CRP-Based Modeling Attacks

Analog PUF variants and challenge obfuscation are not the only protections from CRP-based modeling. Other methods attempt to confuse or poison the modeling algorithm. To do this they intentionally attempt to poison the responses once a while, so the adversary model is fed with incorrect responses; thus results in poor modeling of the PUF and induces mispredictions in the operation of the model.

One of the methods that is used to poison ML algorithms is the use of a misleading PUF. In this methodology a 'Fake PUF' is implemented in the device and challenges are queried on the decoy device. The queries of the Fake PUF are performed between those of the genuine PUF such that an eavesdropping attacker will

collect illegitimate CRPs. If the attacker attempts to model a PUF with these CRPs their model will fail at modeling the genuine PUF due to the poisoned dataset [35].

Poisoning takes a lead in another creative methodology for preventing CRP-based modeling attacks utilizing adversarial ML techniques. In this methodology the attacker's ML modeling efforts are craftily poisoned with inputs that intentionally create mistakes in the attacker's model. This poisoning is performed in the Adversarial PUF using a modified CRP technique which systematically flips the output response for carefully chosen challenges to make the model intentionally make the wrong prediction [86]. Ebrahimabadi et al. [26] also used machine learning to decide when to poison the PUF response such that it cannot be modeled. This protocol level approach is used in authentication of IoT devices. Another adversarial-learning based protection scheme against CRP-based attacks was proposed in [54]. This scheme estimates the modeling accuracy that an adversary may achieve and toggles the transmitted responses if the estimated accuracy is beyond a pre-defined threshold value. Note that most of these methods protect the PUFs against CRP-based modeling attacks, and don't consider implications on the power consumption of the device.

3.2 Side-Channel based Modeling

Various forms of side-channels such as acoustic [34], Radio Frequency/ electromagnetic [68, 73], and power [15, 57], may be exploited to compromise the security. Side-channel based attacks can be exceptionally effective in compromising ICs and

leaking sensitive data; particularly for PUFs the power side-channel is of interest. In these attacks, rather than a set of CRPs being collected, underlying side-channel characteristics are gathered.

For power-based modeling attacks, the power consumption (current draw) of the circuit is collected along with the PUF response. More specifically, the side-channel characteristic of interest is the current draw during the time that the PUF is being queried until its response is stable. This information is collected for a number of different challenge queries. This combination of power trace and response, creates the trainable feature set for the ML algorithm. In this case, the actual challenge does not matter, only the current draw and the eventual response [15,57] are needed for modeling the PUF. This modeling is possible due to the current leakage of the components that make up the PUF circuitry. Note that here the current draw of the entire power trace is not required; only the power trace for those components that clearly show the difference between a response being a '0' and a '1' are needed. It is important to note that the loading on the components affects their current draw so that with increased loading, the power consumption of the components is higher [15, 57].

Modeling the PUF via its power traces is more efficient than through its CRPs. In the power-based modeling, a model can be directly trained on portions of the power trace in which the components discern the response. If the noise of the circuit is minimal, then the current draw of the critical components for discerning the PUF's response makes its response readily identifiable without modeling [15]. In reality there will be systematic noise when collecting the power traces from the IC

containing the PUF. This noise includes the measurement noise as well as the current draw of the other components and circuitry within the IC which exist outside of the PUF circuit. In power attacks, the Signal-to-Noise Ratio (SNR) serves to represent the feasibility of being able to attack a circuit. The generally accepted calculation of SNR is presented in Equation 3.5 and is the ratio of the variance of the signal (inter-variance) and the variance of the noise (intra-variance) [62, § 4.3.2].

$$\text{SNR} = \frac{\text{Var}(Signal)}{\text{Var}(Noise)} \tag{3.5}$$

Research on the arbiter-PUF shows that a plausible SNR for a real silicon implementation is around 1.81 [29].

Chapter 4: Self-PUF Attacks

This chapter pertains to the results and discussions on Self-PUF attacks. Self-PUF attack data is collected from the same PUF that is intended to be attacked. For instance CRPs or power traces collected from a PUF are used to attack the PUF from which they were collected. These attacks are the traditional way in which devices are compromised.

4.1 CRP-based Modeling Attacks

The conventional way to exploit PUFs via modeling is through the use of their CRPs. While this type of attack is not novel, it is important to conduct as a baseline. CRP-based modeling is an attractive method for exploiting PUFs because it does not involve invasive manipulation of the device or in some cases the device does not need to be physically accessible at all as the CRPs used for device authentication can be eavesdropped and used for modeling the deployed PUF. In other words, they are a straightforward type of attack to execute. This straightforwardness makes CRP-based modeling attacks a focus for investigation and for developing countermeasures. The CRP-based modeling attack results are shown for two selected countermeasures: an analog PUF variant (the VTC PUF,

see Section 3.1.2), and a Challenge Obfuscated PUF (hereafter referred to as a CO-PUF, see Section 3.1.3) alongside the results for a regular unprotected arbiter-PUF. Before discussing the results of the attack, it is important to understand the threat model.

4.1.1 Threat Model

In this attack an adversary is attempting to make a model of the PUF based on its CRPs. The acquisition of the CRPs is usually performed when the chip is still "open" (in its enrollment phase) to interrogate for any challenge. The attack occurs afterward, when the IC has been deployed in a critical system, to retrieve valuable information (in the post-customization phase). This process is shown in Figure 4.1. Note that this attack can be also launched when a device is authenticated using a server or a controller node through its CRPs, for example in IoT frameworks where the CRPs are transferred via wireless communications and thus can be eavesdropped by the adversary.

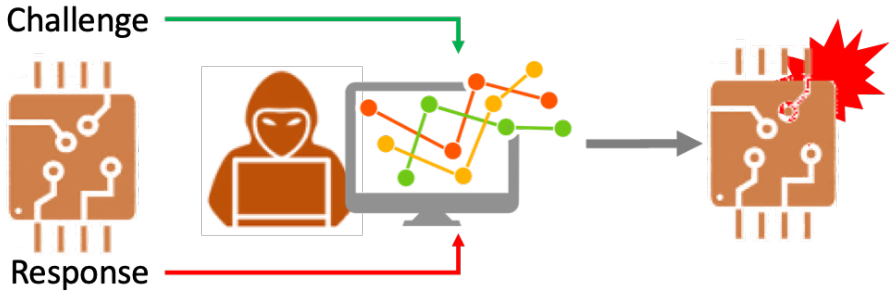


Figure 4.1: Threat model for CRP-based modeling attacks in which an attacker collects CRPs, creates a model, and then deploys said model to exploit the IC.

4.1.2 CRP-based Modeling Results

What follows are the CRP-based modeling attacks for an unprotected arbiter-PUF, the VTC PUF, and the CO-PUF. The details of the simulation environment, logistics, and the data extraction are shared in Appendix A. The results are presented in terms of the accuracy of the attack. A simple method of assessing accuracy is the ratio of the correctly predicted responses and the total number of responses tested (as can be seen in Equation 4.1[1]). Each set of results shares the number of CRPs that are used for creating the model and the number used for testing.

$$\text{Accuracy}(\%) = \frac{\text{Predicted Correctly}}{\text{Total Tested}} * 100\% \tag{4.1}$$

Arbiter-PUF

The results for attacking the arbiter-PUF through its CRPs are shown in Figure 4.2 (note that the axis starts at 95%). The results are for attacking a 64-bit arbiter-PUF and are evaluated using 5000 randomly chosen CRPs. As shown, with just 1000 CRPs the PUF's response can be predicted correctly approximately 97% of the time.

VTC PUF

Figure 4.3 shows the results of attacking a 64-bit instance of the VTC PUF. Although the training accuracy is high the evaluation accuracy hovers a little over 50% (evaluated using 6000 CRPs). This accuracy is as expected for the protection

[1]The accuracy equation is used for assessing the accuracy of all the attack results presented in this research.

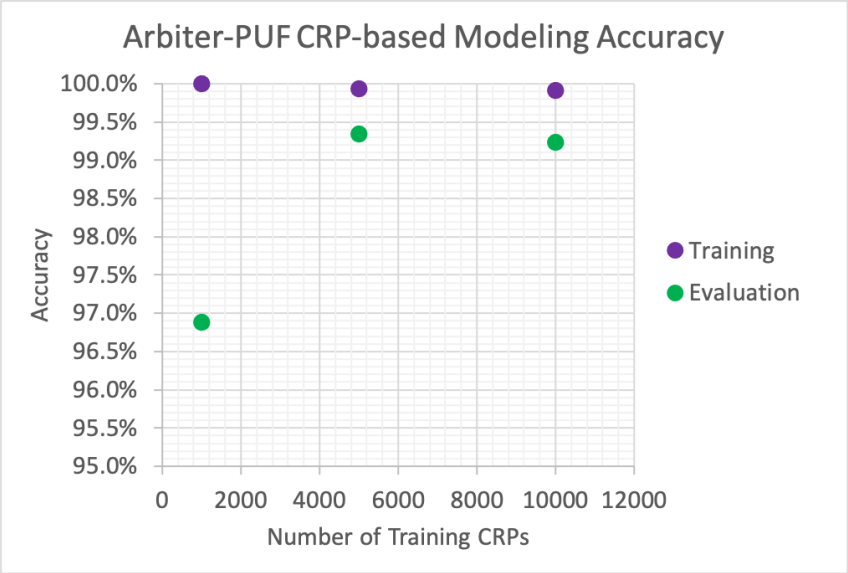


Figure 4.2: The accuracy of the CRP-based modeling attacks on a 64-bit arbiter-PUF. Evaluated on 5000 CRPs.

provided by the VTC PUF and shows that it meets its goal of mitigating the CRP-based model attack. As a point of interest the difference in accuracy between the training accuracy and the evaluation accuracy is likely due to overfitting of the model. In order for the model to reach convergence in training it must essentially learn the responses for the challenges seen in the training dataset. So the model memorizes the result rather than finding the pattern required to use the model on unseen challenges.

CO-PUF

The results of CRP-based modeling attacks on the CO-PUF are shown in Figure 4.4. Here the model was evaluated on 5000 CRPs. Both the training accuracy and the evaluation accuracy are very low. The training accuracy trends towards 50% and the evaluation accuracy is just over 50%. The evaluation accuracy is as expected for the implementation of a PUF that employs challenge obfuscation, and

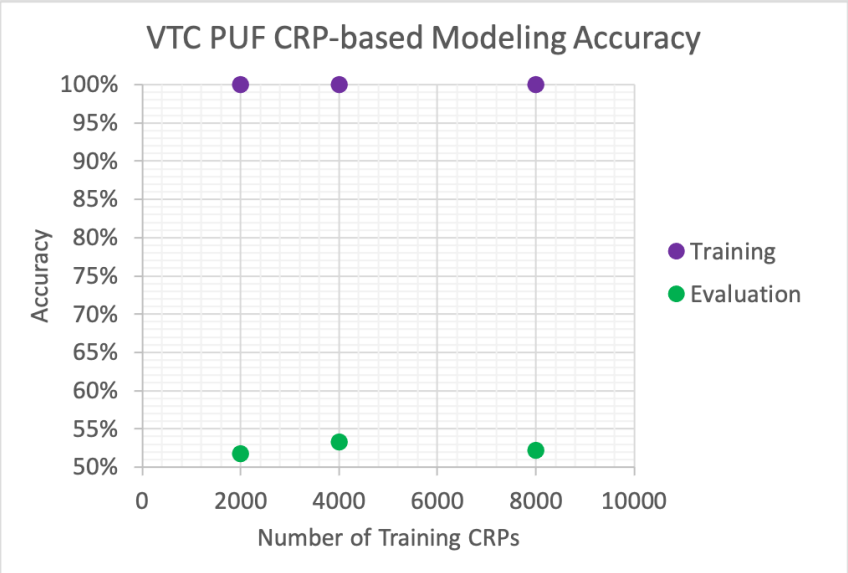


Figure 4.3: The accuracy of the CRP-based modeling attacks on a 64-bit VTC PUF. Evaluated on 6000 CRPs.

the results confirm its mitigation of CRP-based modeling attacks. The low accuracy in the training data shows that the obfuscation technique is effective at manipulating challenges in an unpredictable way. The training accuracy's convergence to 50% makes sense with the thought that there are more random samples being added to the training data and therefore the model gets worse as it is unable to fit to a pattern.

4.1.3 Discussions

There is value, as shown here and in literature, of mitigating CRP-based modeling attacks however, there are issues with the solely focusing on such attacks. The first issue in CRP-based modeling is the large number of CRPs required to create a model of the PUF. An adversary would need considerable access to either the device itself or its communications to gather the required amount of traces for an accurate

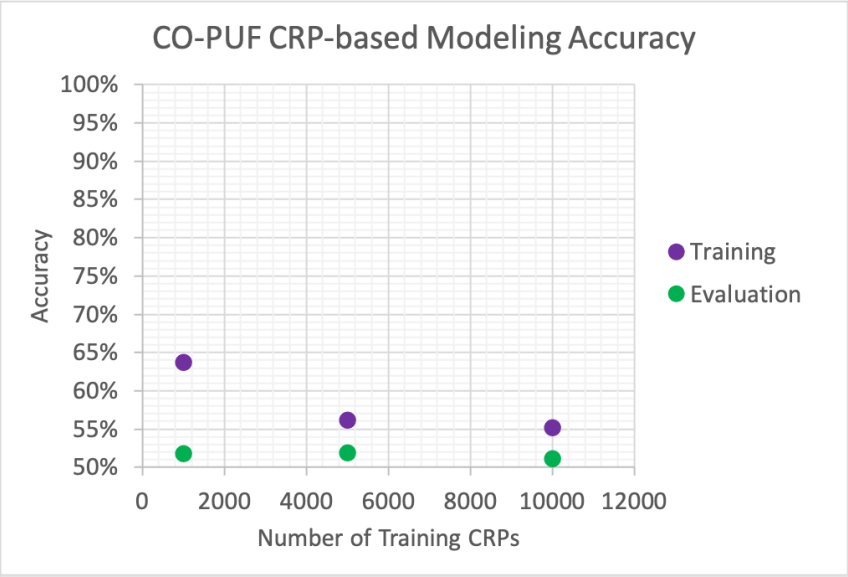


Figure 4.4: The accuracy of the CRP-based modeling attacks on a 64-bit CO-PUF. Evaluated on 5000 CRPs.

model. The second issue is with the accessibility to the portions of the circuit that would reveal the CRPs. After the device's enrollment phase those traces are rendered inaccessible through anti-fuses. Gaining access to the traces is still possible through physical probing but only to the most sophisticated attacker as physical manipulation may damage or change the PUF. Finally CRPs are rarely exchanged in the clear; there is likely some form of encryption that prevents mere logging of the CRPs to prevent eavesdropping as a feasible collection tactic. These issues make practical CRP attacks improbable.

4.1.4 Contributions

From the stated CRP-based modeling results it is shown that:

1. The arbiter-PUF is attackable through its CRPs (as mentioned in previous research studies [15, 74, 75]);

2. Both the VTC PUF and CO-PUF are effective countermeasures against CRP-based modeling attacks.

The CRP-based modeling attacks are of interest as this type of modeling is the conventional modus operandi of investigations on the security of PUFs. It is therefore the most common vulnerability for the development of countermeasures.

4.2 Power-based Modeling Attacks

In power-based modeling attacks, information from the power side-channel of the PUF is collected in tandem with the PUF's response to create the model of the device. The side-channel information gathered is the current draw of the device referred to as a power trace. Instead of modeling the function using the challenge response pairs, the physical characteristics of the power consumption or current leakage of the device are used in modeling.

Before discussing the power attacks, it is important to point out that following every instance of a PUF there will be system components that surround the PUF. The particular system components of interest are those involved with using and storing the response of the PUF. This interest in turn means that the arbiter-PUF in Figure 2.4 is more accurately represented in a system as shown in Figure 4.5 with a register (Flip-Flop) and a load (simulated by an added capacitance C). The rationale for the addition of these components is that whenever a PUF is being used it will need to store the response in some way before it is either used in internal logic or sent for authentication.

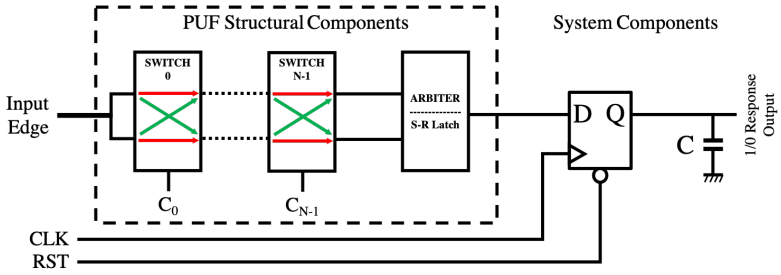


Figure 4.5: Arbiter-PUF with system components.

The process for querying a PUF is to provide the challenge to the switches then the rising edge to the input and waiting for the response to become stable before registering it in the output Flip-Flop. The period of time from the rising input edge till the registration of the response is the period in time that the current is sampled to create the power traces. Examples of power traces during the sampling period are shown in Figure 4.6.

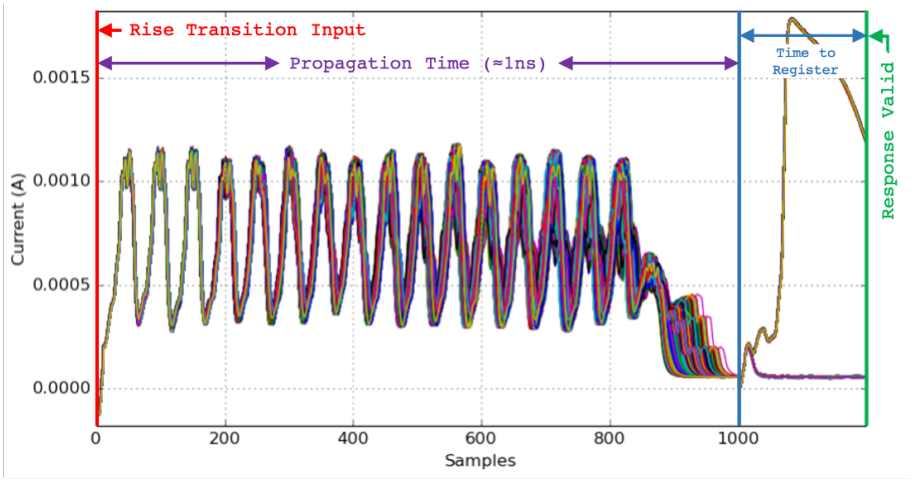


Figure 4.6: A collection of power traces for a 16-bit arbiter-PUF.

Observing the power traces with the structure of the PUF in mind there are components that are integral to distinguishing the response of the PUF. These components of course are the arbiter and the response Flip-Flop. Figure 4.7 shows

these components and highlights their leakage contributions to the power trace of the device. It is these portions of the power trace which will reveal the most information about the PUF's response.

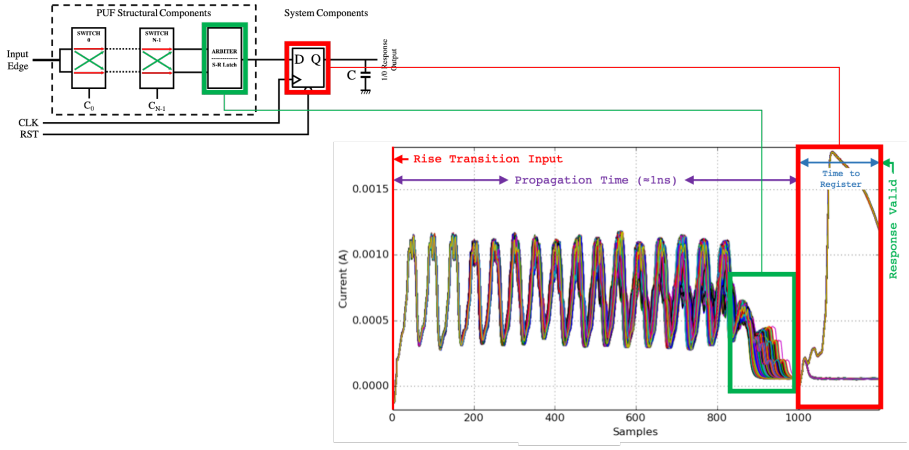


Figure 4.7: Components attributing to the current leakage of the PUF which reveal the response.

4.2.1 Threat Model for Power-based Modeling

The threat model for power-based modeling attacks is shown in Figure 4.8. An attacker monitors the power consumption of the device while it is in operation, extracting the power for the periods of time that the PUF is being queried. These traces, as previously mentioned, correspond to the physical operation of the device and are used to train a model from which the response can be inferred. This model can then be used to attack the device at a later time once it has been deployed in a system of interest to the adversary.

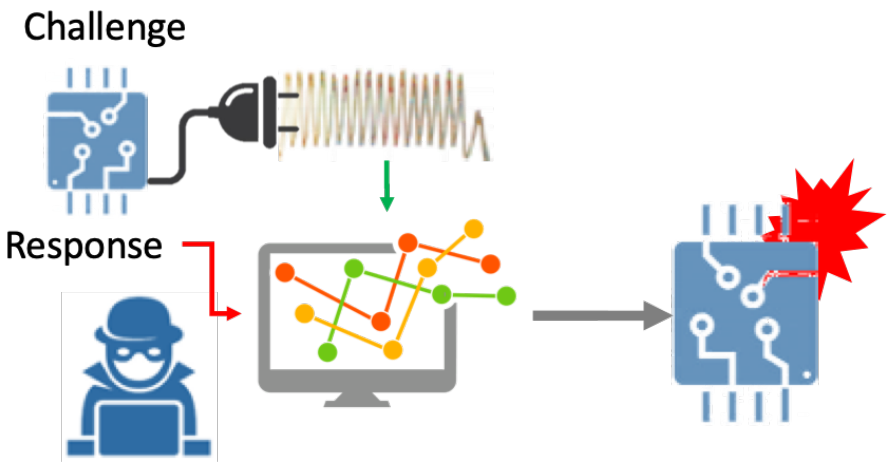


Figure 4.8: Threat model for power-based modeling attacks.

4.2.2 Power Trace Analysis

The information discernible in the power traces of the device can be assessed by the SNR of said traces. To calculate the SNR the originally presented equation (Equation 3.5) is expanded in context to these power traces.

To represent the SNR in this way, suppose that there is a collection of i number of power traces (x) with j number of samples for a response r. Then a per sample mean and variance can be obtained for a response r through the following equations:

$$\mu_r(j) = \frac{1}{N}\sum_{i=0}^{N-1} x_i(j) \ , \ \sigma_r^2(j) = \frac{1}{N}\sum_{i=0}^{N-1}(x_i(j) - \mu_r(j))^2 \tag{4.2}$$

where r can be either 0 or 1 for a single bit PUF. The equations in 4.2 expand the SNR equation to the following:

$$\mathrm{SNR}(j) = \frac{1}{2}\frac{[(\mu_0(j) - \frac{\mu_0(j)+\mu_1(j)}{2})^2 + (\mu_1(j) - \frac{\mu_0(j)+\mu_1(j)}{2})^2]}{[\sigma_0^2(j) + \sigma_1^2(j)]} \tag{4.3}$$

For the investigations within this research the observation of the maximum point of equation 4.3 is the principal concern.

From a vulnerability standpoint, traces with a high SNR are more likely to be successfully attacked than those with low SNR. This rate of success is because the signal, the differentiability of a 0 and a 1, is high when the SNR is high as such the SNR is a measure of the feasibility of an adversary's attack.

Note that the SNR calculation only makes sense when there is noise introduced to the power traces. For these results noise introduced by adding gaussian noise to the extracted power traces. The process for adding noise is given in Appendix A.

4.2.3 Power-based Modeling Results

The results of Self-PUF power-based modeling attacks are shared below for the arbiter-PUF, VTC PUF, and the CO-PUF. The details of the simulation environment, logistics, and the data extraction are shared in Appendix A. The accuracy is still the ratio of correctly predicted results over the total number tested as shown in the accuracy Equation 4.1. The gaussian noise added to these results was σ ={2.5e-4, 16e-4, 32e-4, 64e-4}. The noise levels were chosen to determine the level at which the attack accuracy would start to degrade. The attack accuracy for targeting both latch and Flip-Flop are discussed (with the exception of the VTC PUF). Each set of results shares the number of traces that are used for creating the model and the number used for testing.

Arbiter-PUF

The power-based modeling attack results for attacks on the arbiter-PUF are shown in Figure 4.9. The presented results are for an instance of a 64-bit arbiter-PUF trained on 500 power traces and evaluated on 5000. The SNR results are presented in Table 4.1 for both the arbiter and Flip-Flop at each of the investigated noise levels.

Without noise, attacking the arbiter has a very high level of success ($> 99\%$). When noise with σ =2.5e-4 is added, the accuracy drops to just 90.4%. At greater levels of noise the attack starts to fail because the signal is hidden by the noise completely, as seen in Table 4.1. For the larger noise values the SNR becomes very low.

The attack, when the Flip-Flop is targeted, is much more successful. The SNR, from Table 4.1 for σ =32e-4 and σ =64e-4 is 0.08 and 0.023 respectively, are very low yet there is still a high level of accuracy in the attack. The accuracy of the power-based modeling attack dips to below 100% once the noise level reaches σ =32e-4. At σ =64e-4 the accuracy falls to just below 90%.

The result of the attack on the latch and Flip-Flop show that arbiter-PUFs are not only vulnerable to CRP-based modeling attacks but also those based on modeling the power. These attacks have been investigated in literature previously [15, 57] but it is presented here to be used as a point of comparison for eventual investigations in to countermeasures to the power-based modeling attack. The fact that the attack can occur at SNR levels far below that seen in real devices is concerning as it points to just how powerful these attacks are.

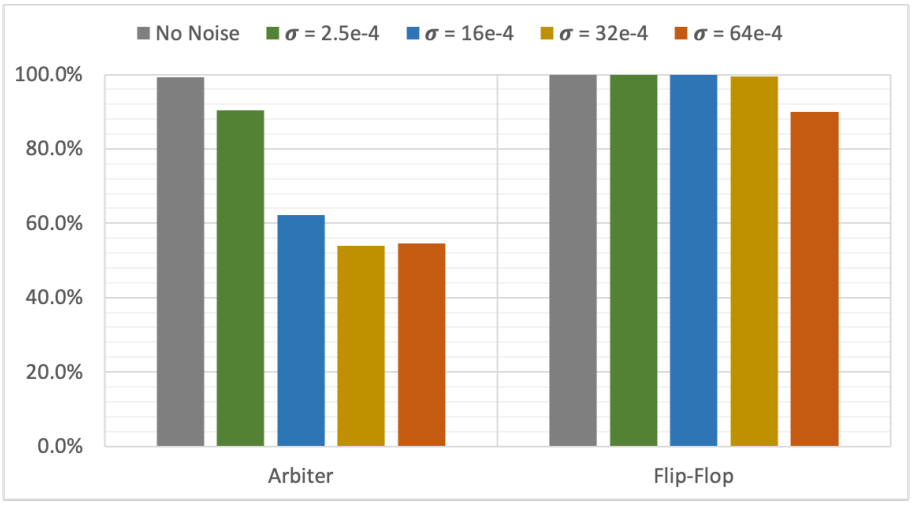


Figure 4.9: The accuracy of the power-based modeling attacks on a 64-bit Arbiter-PUF. Trained on 500 traces and evaluated on 5000 traces.

Table 4.1: The maximum SNR for the traces related to the Arbiter-PUF's arbiter latch and Flip-Flop.

	σ = 2.5e-4	σ = 16e-4	σ = 32e-4	σ = 64e-4
Latch	0.183488	0.006920	0.001671	0.001244
Flip-Flop	12.320019	0.308742	0.079990	0.022701

VTC PUF

The VTC PUF was proposed as a modeling-resilient counterpart of the arbiter-PUF recently in Vijayakumar et al. [84]. As was shown in Section 4.1.2, these PUFs are effectively resistant to CRP-based modeling attacks. However, this research attacked VTC PUFs for the first time and showed that these PUFs can be modeled by monitoring their power consumption [50].

Before delving into the power-based modeling attacks, a discussion on the power traces of the VTC PUF is necessary. The power traces for the VTC PUF are different from those for the arbiter-PUF shown in Figure 4.6 due to it operating based on analog voltages. The traces for the VTC PUF are shown in Figure 4.10.

Seen in this figure is a power spike when the response query starts followed

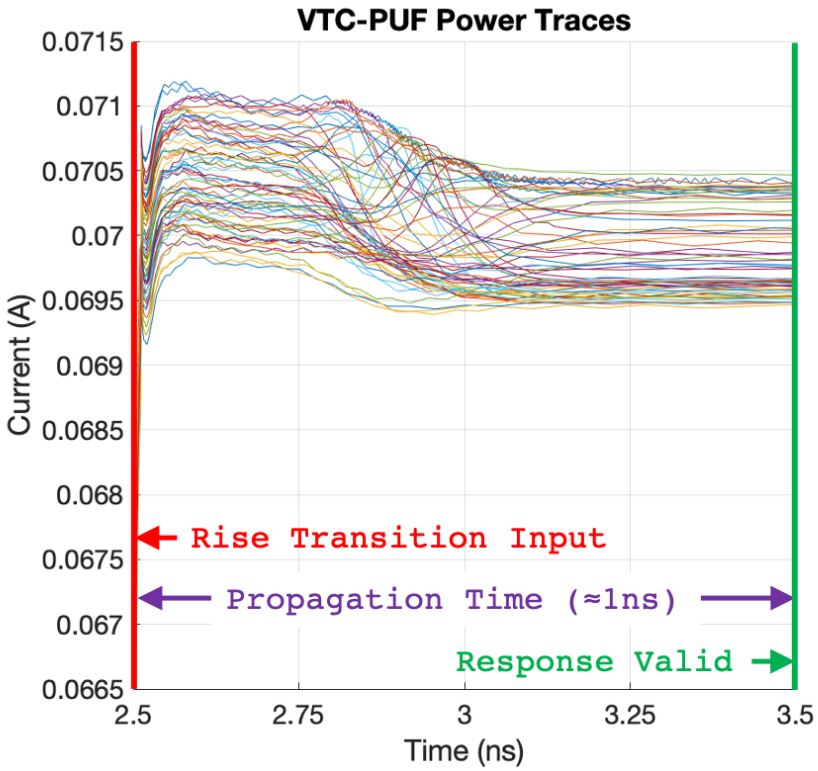


Figure 4.10: The power traces extracted from the VTC PUF.

by slight oscillations due to the transition through the VTC Blocks and analog switches. The leakage from the arbiter operation can be seen midway through the power trace characterized by the apparent switchbacks in the trace. The arbiter operation is highlighted in Figure 4.11.

The results of attacking the aforementioned VTC PUF traces are shown in Figure 4.12. These results are for the no noise situation of attacking the arbiter of the VTC PUF. The accuracy of attacking the arbiter is approximately 83% when 2000 traces are used to model the PUF. There is only moderate increase in success of the attack when increasing the number of attack traces to 8000 as the accuracy only increases to approximately 85% in this case. The results of attacking the PUF using its arbiter (latch) show that the PUF is vulnerable to power-based modeling

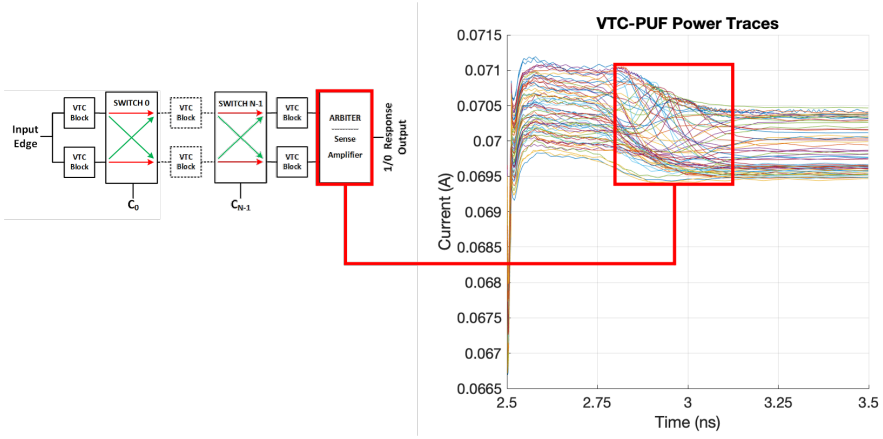


Figure 4.11: VTC PUF where arbiter operation highlighted.

attacks. The leveling off of the result may show that there may be a limit to the accuracy that can be achieved by attacking the latch. The misclassified responses are very likely close to the boundary between being classified as a '0' or a '1' and therefore are more difficult responses to distinguish.

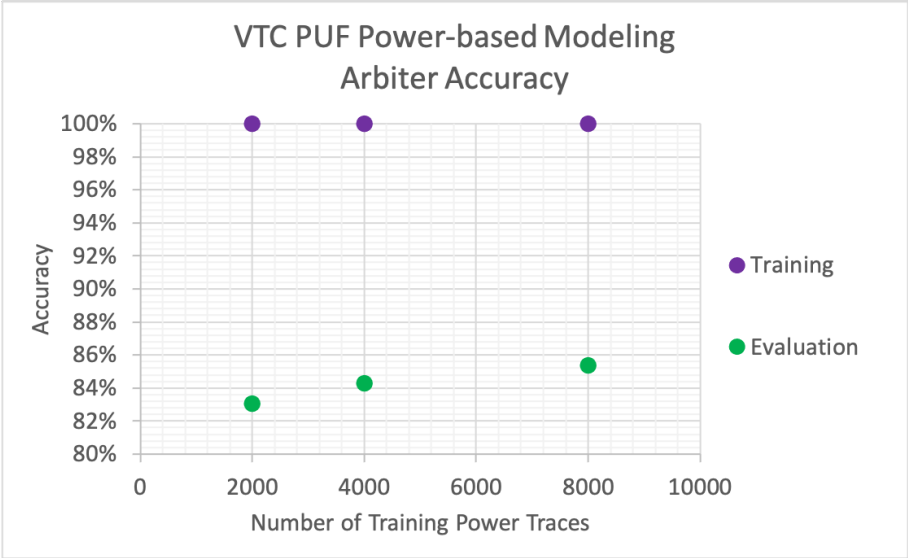


Figure 4.12: The accuracy of the power-based modeling attacks on a 64-bit VTC PUF. Evaluated on 6000 traces.

Returning the attention to the traces, it can be seen via Figure 4.13 that the outputs do create groupings in which the power gives away the response of the PUF.

Some mixing of the power traces for each response value occurs which is where the model has difficulty distinguishing their value.

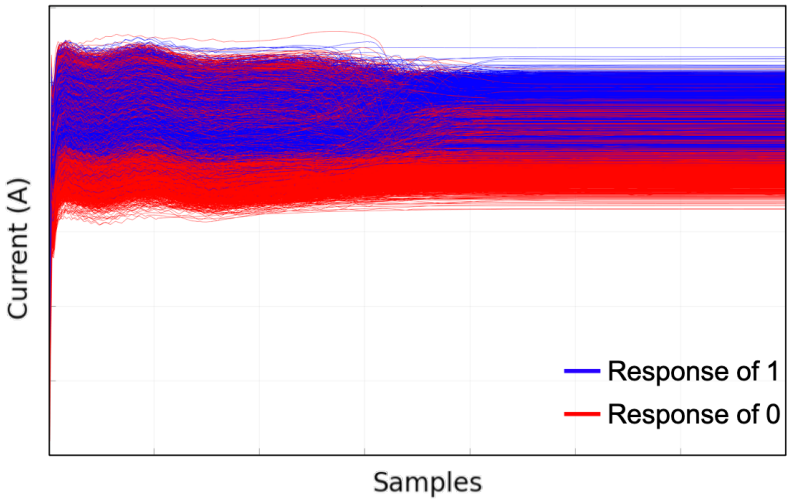


Figure 4.13: VTC PUF power traces with their response value highlighted.

CO-PUF

The CO-PUF, like the VTC PUF, was implemented to protect against CRP-based modeling attacks [91]. This protection is provided by the fact that the attacker does not have access to the actual challenge being given to the PUF rather the CRP set they are working with contains the challenge that is observed before it is obfuscated. This CRP set confuses the modeling algorithm because the challenges don't reflect the actual value of the switches used for the response query.

This addition to the PUF was very effective in thwarting CRP-based modeling attacks, however the technique of challenge manipulation serves little to no success in doing the same for thwarting power-based modeling attacks. There is no existing research on the power-based modeling of challenge obfuscation techniques outside of that presented in [51]. The power-based modeling results suggest that these

types of modeling attacks circumvent such protections by bypassing the use of the challenge entirely. If the challenge is not needed by the model to determine the response of the PUF then such protections will not be effective.

Figure 4.14 shows the accuracy of the power-based modeling attacks on the CO-PUF. For attacks on the latch the accuracy is very high when there is no noise present (again $>$ 99%). When there is added noise of σ =2.5e-4 the accuracy drops to 88.8% and decreases ever closer to 50% for the higher noise levels. At first glance, targeting the latch appears to be somewhat difficult to attack in the presence of additional noise showing again that it is not the best avenue for the power-based modeling attack to be performed. However, if the attention is turned to the system components in which the response Flip-Flop lies, the effectiveness of the attack becomes more evident. When attacking the Flip-Flop the results are very promising; achieving 100% accuracy for no noise and the first two levels of noise (σ ={2.5e-4, 16e-4}. When σ =32e-4 the accuracy only drops to 99.4% and when σ =64e-4 the accuracy drops to 87.4%. These accuracies are no different than those seen by the unprotected arbiter-PUF showing that the challenge obfuscation is not an effective countermeasure for these attacks. As it is shown in these results, the location or focus of the attack is highly important and can drastically change the effectiveness of the attack therefore it is important to know exactly when to launch the attack, in this case it is the time that the Flip-Flop is queried.

The SNR of the power traces for the CO-PUF are shown in Table 4.2. At σ =2.4e-4, the noise level for which the modeling attack on the arbiter latch was successful, the SNR is 0.157530. This result is far below the level seen in real silicon.

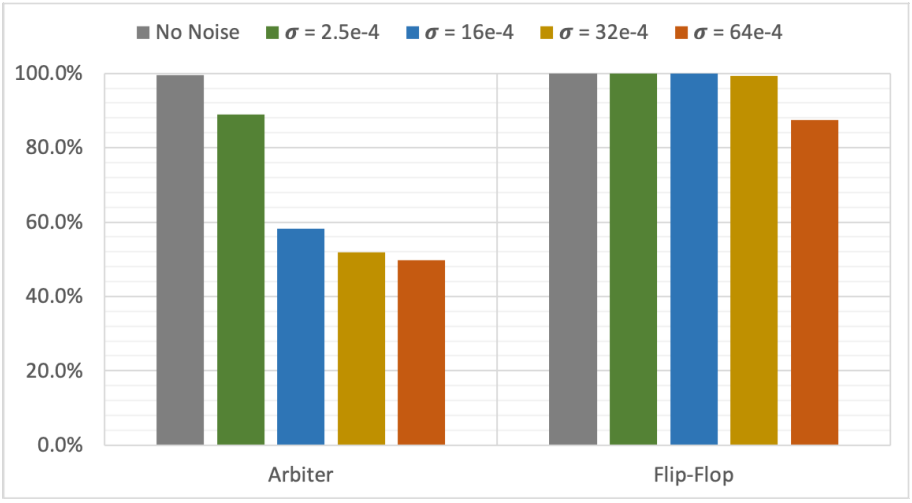


Figure 4.14: The accuracy of the power-based modeling attacks on a 64-bit CO-PUF. Trained on 500 traces and evaluated on 5000 traces.

At the greater noise levels it can be seen that the SNR is significantly smaller showing that the noise is masking the effects seen in the latch. For the Flip-Flop the SNR at σ =32e-4 is 0.07999 which is also below the noise level seen in real silicon. The low levels of SNR for which the attack is successful show, just like the arbiter-PUF, how powerful these attacks are at compromising PUFs.

Table 4.2: The maximum SNR for the traces related to the Arbiter-PUF's arbiter latch and Flip-Flop.

	σ = 2.5e-4	σ = 16e-4	σ = 32e-4	σ = 64e-4
Latch	0.157530	0.005831	0.001893	0.001463
Flip-Flop	12.320019	0.308742	0.079990	0.022701

4.2.4 Discussions on Launching Power-based Modeling Attacks on Protected PUFs

The results showed that the arbiter PUF and its (supposed) resilient counterparts were all vulnerable against power side channel attacks. Also the attack

required less information than CRPs to achieve a high accuracy of attack. Recall that the CRP based modeling attacks required 1000 traces to achieve an accuracy of 97% (for this research's implementation as it may vary slightly from one technology to another) and from literature approximately 2500 to achieve 99% accuracy [74], whereas the power-based modeling presented here achieved $> 99\%$ with just 500 traces. It is also observed that, even though the latch is attackable, the Flip-Flop is a much better attack point on the PUFs. The Flip-Flop is such a great target that when viewed without noise the response can be easily determined as seen in Figure 4.15. However, noise is unavoidable in real circuits therefore additional noise is added to the power traces here before modeling the PUFs. Even with the additional noise the leakage from the Flip-Flop is highly considerable. That being said the latch is still important in investigations as it behaves as the arbiter determining the PUF response based on the underlying physical manufacturing variations being exploited (time delays for the arbiter-PUF, voltage levels for the VTC PUF). The Flip-Flop is a system component and will always be present to register the response before being used. However, to make the arbiter-PUF family resilient against power-based attacks that target such the Flip-Flop, in this research a few circuit level modifications are proposed to the circuitry storing the PUF response [51] (discussed later in this chapter).

Another interesting observation is that the size of the PUF does not matter for targeting the leakage of the output Flip-Flop. Figure 4.16 shows power traces for a 16-bit, 24-bit, and 64-bit arbiter PUF. In each set of traces the leakage of the Flip-Flop is highlighted and shows that the leakage is independent of PUF size. This

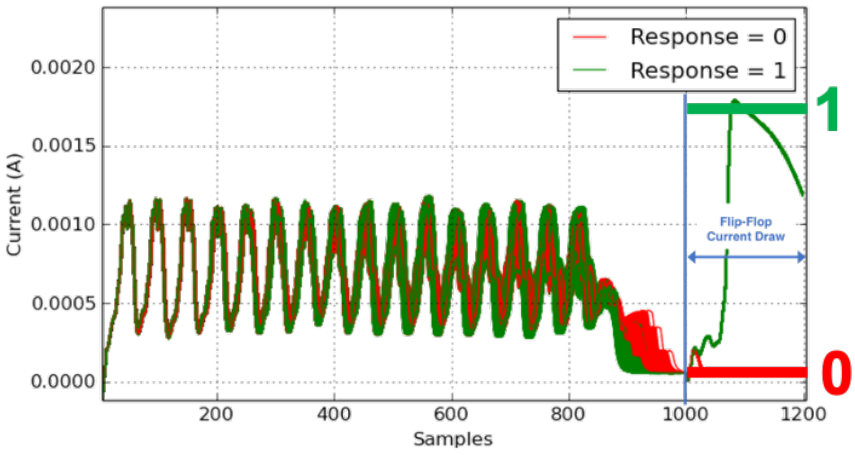


Figure 4.15: A collection of arbiter-PUF power traces highlighting the distinguishability of the response based on the behavior of the Flip-Flop.

indifference means that increasing the PUF size does not increase the resiliency of PUF against power-based modeling attacks.

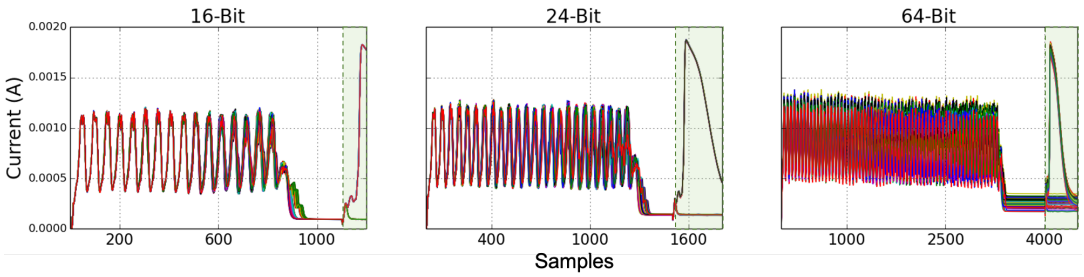


Figure 4.16: Power traces for different sized arbiter-PUF implementations highlighting the presence of output Flip-Flop leakage.

The final observation that stands out from the results is that the arbiter-PUF and CO-PUF have nearly identical results. This similarity should be expected due to the fact that the CO-PUF is a modified version of the arbiter-PUF. The modifications are also only placed on the challenge input and since the challenge is settled in the switches before the PUF is queried for a response, the additional circuitry does not change the power trace. This lack of change can be verified by

observing the power traces of the arbiter-PUF and CO-PUF as seen in Figure 4.17. Comparing the SNR for the arbiter-PUF in Table 4.1 and that of the CO-PUF in Table 4.2, indeed even the SNR observed for each of the noise levels is nearly the same. From these observations it is assumed that the CO-PUF and the arbiter-PUF display the same vulnerability to power-based modeling attacks. The same observation is expected for other derivatives of arbiter-PUF such as XOR-PUF [93] and FEED-Forward PUF [11].

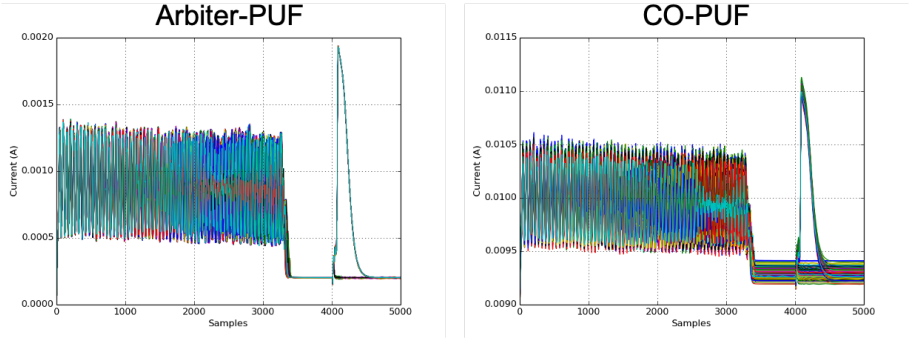


Figure 4.17: The accuracy of the power-based modeling attacks on a 64-bit CO-PUF. Trained on 500 traces and evaluated on 5000 traces.

The results of the VTC PUF showed that it was indeed vulnerable to the power-based modeling attacks on the latch. Due to the leveling off of the results shown by increasing the number of training traces there may be a maximum achievable accuracy of approximately 85%.

4.2.5 Aging Effects on Power-based PUF Modeling

The following section expands on the understanding of the PUF's vulnerability to power-based modeling attacks by observing how device aging affects the accuracy of the attack. The motivation for investigating the effects of aging on the modeling

attacks is that an adversary very likely will not create and deploy the model at the same time. This observation means that an age differential, between the target device and the model, will be unavoidable. As previously mentioned, transistors degrade over time due to the stresses placed on them. The primary culprits of this degradation are HCI and BTI which are simulated for the results presented here. For the logistics of performing aging simulations see Appendix A. The variability in the power traces due to these effects is shown in Figure 4.18 from no aging (i.e., a fresh IC) to 20 weeks. Over the 20 week time period there exists a significant variation in the power traces. A close up view of the traces show that as expected there is larger variability early in the life of the IC.

In these results a model is created at a specific age of the device using data collected from the PUF at that age. The model is then deployed on data collected from the PUF at different ages. As an example a model of the PUF is created after one week using data from that week. This model is then subsequently deployed to predict the responses from the power traces of the PUF at those other ages.

Arbiter-PUF

Recall that when the attacker develops the model of the PUF there will be a period of time before they are able to deploy said model to attack the device. The effects of this age difference on the power-based modeling of the PUF were previously uninvestigated. The results of attacking the arbitration latch of an arbiter-PUF over time are shown in Figure 4.19. In these results a 16-bit arbiter-PUF was simulated for twenty weeks and attacked with models developed at zero weeks, one week, ten

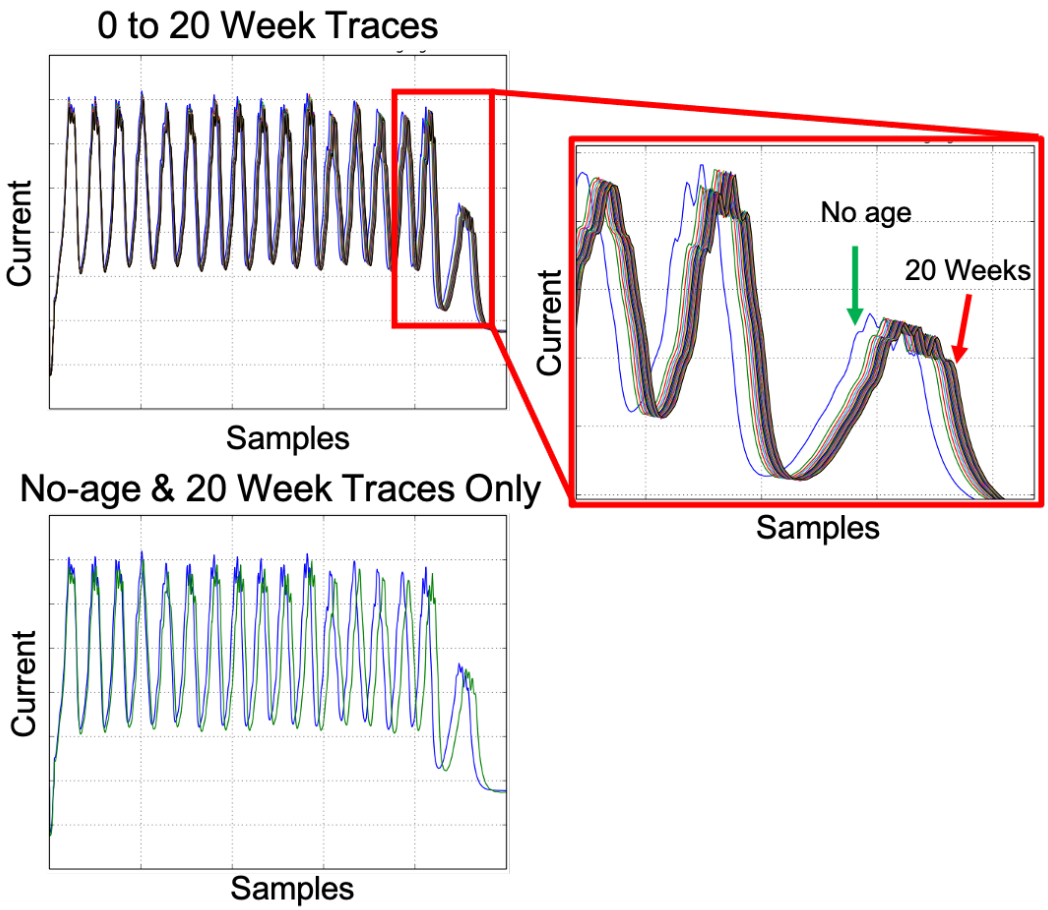


Figure 4.18: Observed power trace variability for the arbiter-PUF from no aging to 20 weeks.

weeks, and twenty weeks. These models were deployed against the other weeks to get a measure as to how effective they were at modeling the PUF over time. The model created at week zero (the model of a fresh device) has limited usefulness due to the fact that it is ineffective at modeling the PUF, since after just two weeks the accuracy drops to approximately 65%. The model created at one week of aging fairs a little better however the accuracy drops blow 90% after five weeks of additional aging. The inaccuracy of these early developed models' ability to attack the differently aged device puts a lifetime on them and their usefulness. At week

ten, the developed model shows much more stability the attack accuracy falls off fairly gently and at week twenty the model is still able to predict the PUF's behavior with over 90% accuracy. The latter aged models perform better over larger periods of times which expands their usefulness lifetime.

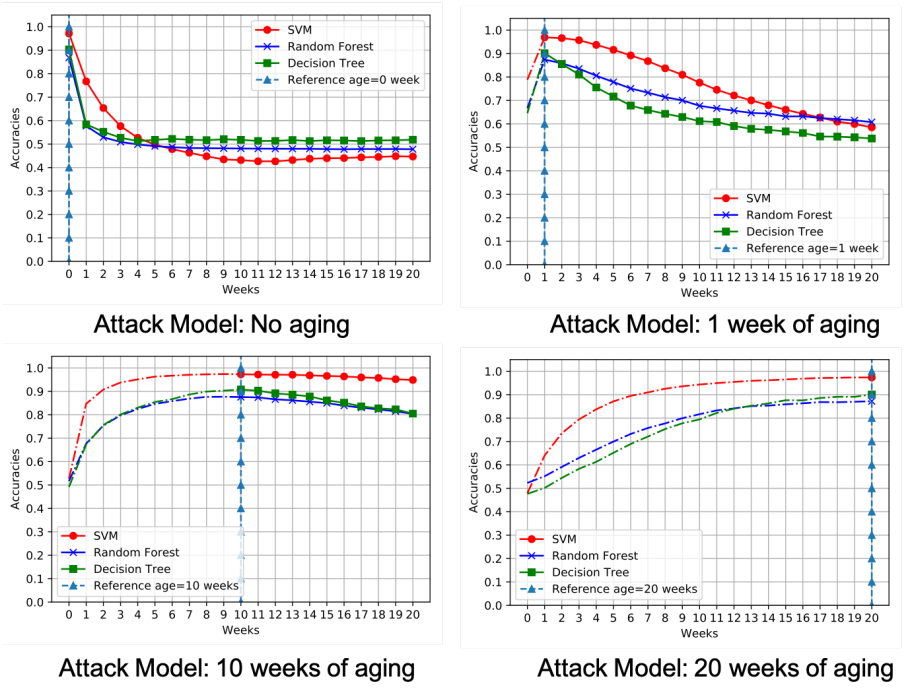


Figure 4.19: Accuracy for modeling attacks on 16-bit arbiter-PUFs (aged between 0 and 20 weeks) when the training model was extracted from an X-week old device, where X$\in \{0, 1, 10, 20\}$. The portions shown with dashed lines represent the cases in which the attacker uses the model of the PUF to extract the previous PUF responses.

VTC PUF

Investigating the effects of aging on the analog counterpart of the arbiter-PUF (i.e., VTC PUF) is performed in this thesis for the first time. Figure 4.20 shows the results of attacking the arbiter of a 16-bit VTC PUF over twenty weeks. A model of the VTC PUF was created for zero weeks, one week, ten weeks, and twenty weeks and then deployed against the rest of the weeks. The results of modeling are

quite similar to those of the arbiter-PUF with early models performing poorly as compared to models created using information from the latter weeks. As observed in the figure, the fresh PUF model accuracy drops below 80% when deployed on the PUF after one week. The model developed from the power traces at one week of aging drops below 90% after the PUF has aged to five weeks. The power-based model developed at ten weeks of aging is able to achieve over 90% accuracy when deployed on power traces from week three to week twenty. Again this is similar to the results shown by the arbiter-PUF.

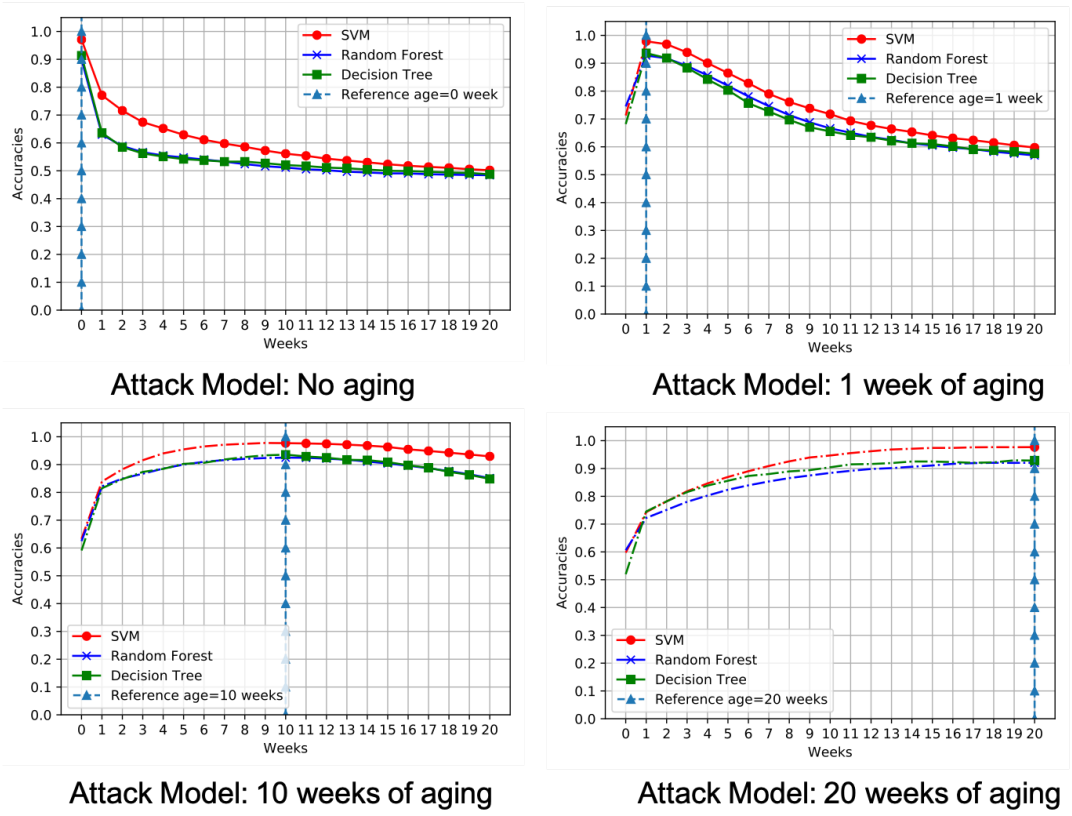


Figure 4.20: Accuracy for modeling attacks on 16-bit VTC PUFs (aged between 0 and 20 weeks), when the training model was extracted from an X-week old device, where X$\in \{0, 1, 10, 20\}$. The portions shown with dashed lines represent the cases in which the attacker uses the model of the PUF to extract the previous PUF responses.

CO-PUF

Recall that the CO-PUF is an extension of the arbiter-PUF therefore its power traces will follow the behaviors of the standard implementation of the arbiter-PUF. This similarity is shown in Section 4.2.4. Thereby, similar to the arbiter-PUF, the misalignment between the aging of the CO-PUF used to create the model and the targeted CO-PUF results in a lower modeling accuracy.

4.2.6 Discussions on the Aging Impacts on Power-based Modeling Attacks

The experimental results show that aging has a large effect on the ability to model a PUF via its power traces when the PUF's arbiter (in this case an S-R latch) is targeted. Models developed from power traces early in the PUF's lifecycle have difficulty achieving high accuracy in latter ages of the device as seen in the results of both the arbiter-PUF and the VTC PUF. This phenomenon is not startling due to the variations in the PUF created by aging; recall that the aging effects are more significant in the early stages of the device lifetime. Also as seen in the results, the models created after the initial large variations due to aging appear to have a window of usefulness. With the decreasing accuracy (however gradual it is) it seems that after a period of time the model will fail to achieve the performance needed to accurately model the PUF.

These results show that an adversary will have limited success if using a model created from power traces from the early life of the device. Since the early period

of time is when the PUF is with the manufacturer and distributor this is a likely time for an attacker to develop a model of the PUF to be later deployed. As it has been shown this model may have limited usefulness. Furthermore an attacker likely cannot wait too long before deploying the model as its accuracy degrades with the age of the PUF.

4.2.7 Contributions in Power-based Modeling Attack Investigations

The above results confirmed that PUFs are vulnerable to power-based modeling attacks. Furthermore it was shown that countermeasures for CRP-based modeling do not protect against power-based modeling attacks which provides motivation for extending countermeasure investigations to power-based modeling attacks. The results of the aging simulations when targeting the arbitration latch showed that there are effects on the power-based models due to the age at which they are developed and deployed. Earlier age models perform poorly when deployed on the older PUF, and there appears to be an inherent lifetime for the model's usefulness when targeting the arbitration latch. In summary this section shows:

1. the vulnerability of PUFs to power side-channel based modeling;
2. the ineffectiveness of CRP modeling countermeasures in preventing against power-based modeling;
3. the effects of aging misalignments targeting the arbitration latch between the template and targeted PUFs on power-based modeling attacks.

Chapter 5: Modeling Multi-Bit Parallel PUFs

The operation of a PUF can be expanded to produce more bits to be used for authentication and other applications, e.g., secret key generation. Certainly this expansion must be done to authenticate devices since a simple '1' or '0' will not differentiate more than two situations. To generate multiple bit of response, One can query the PUF multiple times providing different challenges each time, then utilize a linear shift register on the output to capture the desired number of response bits for the response bit vector [91]. This method utilizes the same PUF multiple times which does not increase the difficulty of a power-based modeling attack. These challenges can be provided externally or can be generated internally from the first given challenge bit-stream. Another method for producing more response bits is through multi-bit parallel PUFs where multiple instances of the PUF are resided in the chip. Considering the area and power overhead of including several PUFs in the chip, usually a combination of both methods are used, i.e., inclusion of a few PUFs but repeatedly querying them. One may think of including of multiple PUFs in the design with smaller size to impose less area and power overhead. Such an example is shown in Figure 5.1. However, this makes the PUF structure more vulnerable to CRP-based attacks.

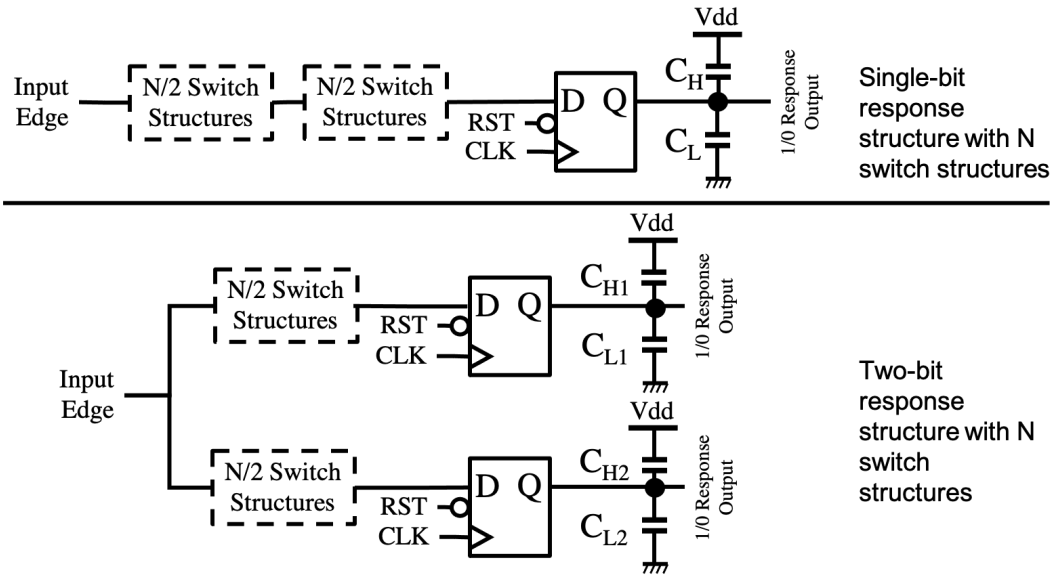


Figure 5.1: The tradeoff between a single-bit PUF implementation and a multi-bit PUF implementation whilst conserving the implementation footprint.

Accordingly, to conserve space on devices, as well as power consumption, it is likely that the instances of PUF are limited to only a few in number. In multi-bit parallel PUF implementations all instances of the PUF operate at the same time which of course means that they will be drawing power at the same time and due to efficiency likely register their outputs at the same time. This timing means that the leakages from the different components will all be present in the sampled power trace. Comparison of the traces for a single-bit PUF and for a 2-bit response parallel PUF are shown in Figure 5.2. Here it can be seen that the leakages in the power traces overlap which motives us to investigate the power-based modeling attacks in these parallel PUF implementations.

When performing power-based modeling attacks on parallel PUFs. If all responses are equally likely then the chances of randomly guessing the correct output is $\frac{1}{2^{\#\text{ of Response Bits}}}$. For example for a 2-bit parallel PUF the accuracy of a chance guess

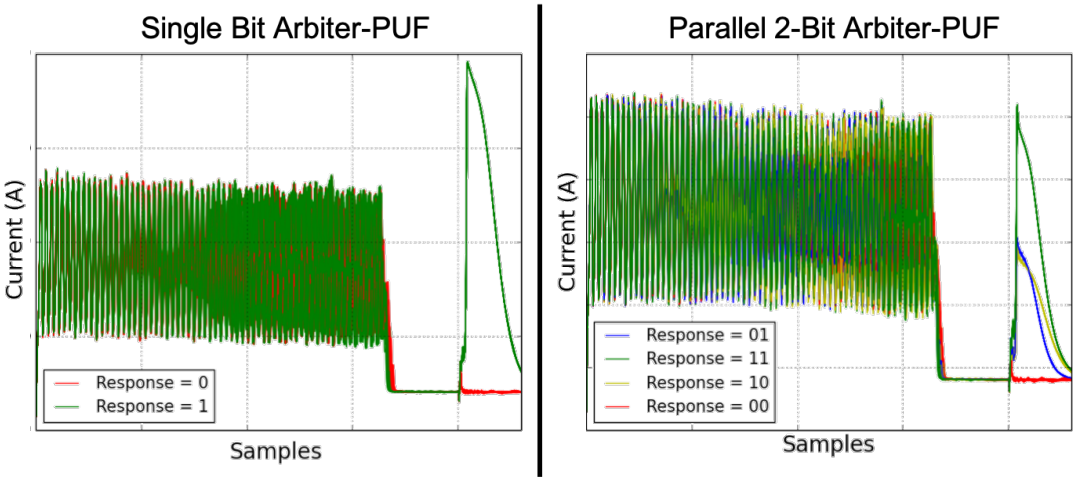


Figure 5.2: Power Traces for a single-bit arbiter-PUF and a parallel 2-bit arbiter-PUF.

is 25%. This lower expected chance creates a lower bound for the attack accuracy for multi-bit parallel PUFs.

It is noteworthy to mention that there are negligible effects on CRP-based attacks. Due to the fact that multi-bit parallel PUFs are multiple instances of the PUF, each bit in the response can be modeled individually. The ability to model each bit individually makes the CRP-based modeling attack similar to deploying on multiple instances of the single-bit PUF.

5.1 Threat Model

The Threat model for this attack is similar to that shown in Section 4.2.1. The adversary collects power traces and creates a model. The portion of the attack that changes is that instead of attempting to infer a single response bit the adversary must infer two response bits.

There are methodologies that an attacker can employ to increase the SNR of

their attacks. One such method is to repetitively play the challenges and average their traces. In this process the same challenge is given to the PUF multiple times and all of the power traces are averaged to reduce the overall SNR in the attack. These repetitive queries will assist the attacker in increasing their attack accuracy.

5.2 Expansion to SNR Evaluations

To characterize the SNR of a multi-bit parallel PUF, the SNR Equation 4.3 needs to be expanded to include the additional bits, recall that this is the expansion of the SNR from Equation 3.5. The SNR is still the ratio of the variance of the signal over the variance of the noise however the individual points now include samples from the expanded response space. If $\mathcal{L}$ is an individual point in the sample trace then $\mathcal{L}_0$, $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ are samples for responses of 00(0), 01(1), 10(2), and 11(3) respectfully. This expanded SNR equation characterizes the distinguishability of the responses and is used for analysis of multi-bit parallel PUFs with two response bits.

$$\mathrm{SNR} = \frac{\mathrm{Var}([\mathrm{Mean}(\mathcal{L}_0), \mathrm{Mean}(\mathcal{L}_1), \mathrm{Mean}(\mathcal{L}_2), \mathrm{Mean}(\mathcal{L}_3)])}{\mathrm{Mean}([\mathrm{Var}(\mathcal{L}_0), \mathrm{Var}(\mathcal{L}_1), \mathrm{Var}(\mathcal{L}_2), \mathrm{Var}(\mathcal{L}_3)])}. \tag{5.1}$$

A generalized version of the SNR equation for any number of bits can extrapolated from Equation 3.5. Similar to Section 4.2.2 suppose that there is a collection of i number of power traces (x) with j number of samples for a response r. Then a per sample mean and variance can be obtained for a response r through the following Equations 4.2. In this the response is any available response from a PUF therefore

the number of responses R is:

$$R = 2^n \tag{5.2}$$

where n is the number of parallel response bits (for a 2-bit parallel PUF $n = 2$ and $r \in \{0, 1\}$, for a 4-bit parallel PUF $R = 4$ and $r \in \{0, 1, 2, 3\}$). If r is the individual response then the average of all the response means is:

$$\mu_q(j) = \frac{1}{R} \sum_{r=0}^{R-1} \mu_r(j) \tag{5.3}$$

Finally, the SNR equation for any size (n) parallel PUF is as follows:

$$\text{SNR}(j) = \frac{1}{R} * \frac{\sum_{r=0}^{R} (\mu_r(j) - \mu_q(j))^2}{\sum_{r=0}^{R} \sigma_r^2(j))} \tag{5.4}$$

5.3 Power-based Modeling Attack Results

The results of the power-based modeling attack on a single-bit PUF and a multi-bit PUF (with two response bits) are shown in Figure 5.3. In these results a 64-Bit two-bit response parallel PUF was implemented in addition to the previous results from a single-bit PUF. For these results gaussian noise was added for σ ={2.5e-4, 9.5e-4, 16e-4, 32e-4, 64e-4}. Here only the Flip-Flop was attacked as it was shown to be a better attack point than the latch. The attack on the single bit PUF (as shown before) was highly successful only decreasing in accuracy (to 90%) when the highest added noise level was attacked. Comparing this to the results of the two-bit parallel PUF there is a large difference in the achieved attack accuracies

in the noise levels where σ >2.5e-4. The accuracy decrease to at σ >9.5e-4 to 95% and continues decreasing to 55% when σ >64e-4.

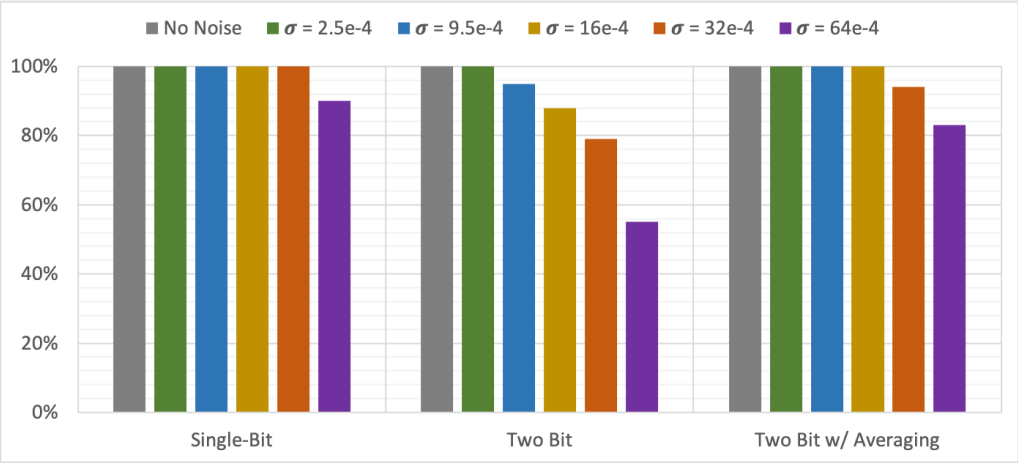


Figure 5.3: Power-based modeling accuracy for the single bit PUF and the Multi-bit Parallel PUF (with 2-bits).

This decrease in accuracy shows that increasing the number of response bits does increase the confusion of the power-based modeling algorithm. However 50% is greater than the accuracy of randomly guessing the response bits. Figure 5.4 shows the traces for the two-bit parallel PUF implementation when there is no added noise with the responses colorized based on their value. As shown in the traces for the Flip-Flop leakage there are four distinct trace lines, one for each response: 00 (red), 01 (blue), 10 (yellow), 11 (green). They appear at three apparent levels: one for 00 (hamming weight=0), one for 01 and 10 (hamming weight=1), and one for 11 (hamming weight=2). Since the levels for 00 and 11 are unique it can be assumed that most of these are being classified correctly and most of the inaccuracy is generated by differentiating 01 and 10.

Figure 5.3 also shows the accuracy of the more sophisticated averaging attack

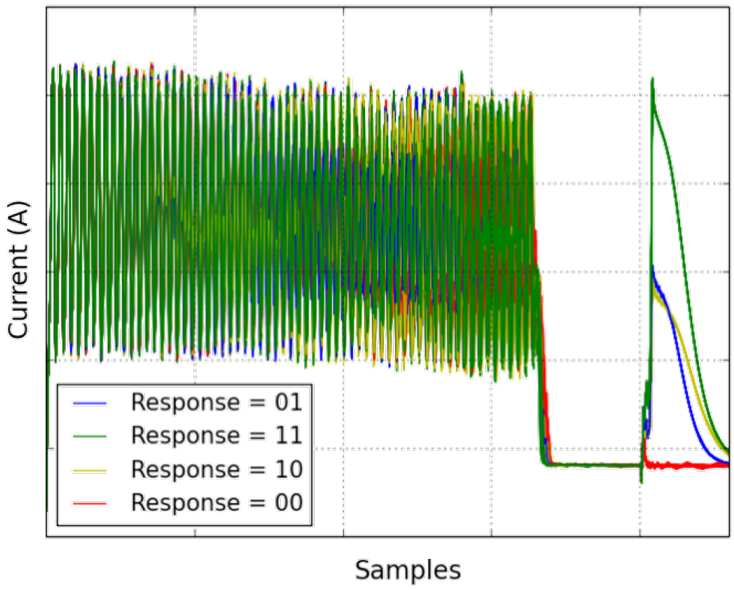


Figure 5.4: Two bit Parallel PUF power traces (no added noise) with the response highlighted.

where the adversary uses multiple queries of the same challenge and averages their power traces to increase the SNR. In this investigation the number of averaged traces is 10. Here it can be seen that the averaging technique does in fact increase the attack accuracy as the accuracy does not drop until σ =32e-4 and only to 95%. The accuracy when σ =64e-4 is higher at 83%. The SNR calculated from Equation 5.1 can be seen in Table 5.1. It can be observed that the averaging technique increase the SNR to levels greater than that of a single bit PUF implementation, and the corresponding increases in accuracy show that it is equivalently attackable. With averaging, the SNR seems to increase by a factor equal to the number of traces. In this case the number of averaged traces is 10 and accordingly the SNR increases by a factor of 10 as expected when dealing with averages of mean and variances from independent and identically distributed random variables [40].

Table 5.1: The maximum SNR when the Flip-Flops are queried in the unprotected parallel PUF with & without averaging.

	σ = 2.5e-4	σ = 9.5e-4	σ = 16e-4	σ = 32e-4	σ = 64e-4
Non-averaging	9.713419	0.658853	0.232567	0.062253	0.016153
Averaging	94.948925	6.666043	2.344949	0.589082	0.146669

5.4 Discussions

The results of the multi-bit parallel PUF show that increasing the parallelism of the PUF implementations decreases the accuracy in straightforward power-based modeling attacks. The accuracy can be boosted significantly through averaging the power traces for repeated challenges. It is apparent that this averaging technique also serves to increase the SNR by a factor equivalent to the number of traces averaged.

The power traces from multi-bit parallel PUFs have apparent power levels in the response traces which coincidentally correspond to the hamming weight of the response vector. The hamming weighs are shown in Figure 5.5 overlaid on the power traces from a single-bit PUF and a two-bit parallel PUF. The simultaneous switching of the response Flip-Flop from a '0' to a '1' creates the apparent power levels which in turn corresponds to the hamming weight of the response vector ranging between 0 and n where n is the number of bits in the response vector.

Extrapolating the levels for larger PUFs can be seen in Figure 5.6 which contains the hamming weight for multi-bit parallel PUFs from one to four response bits.

The knowledge of the hamming weight of the response vector is helpful in

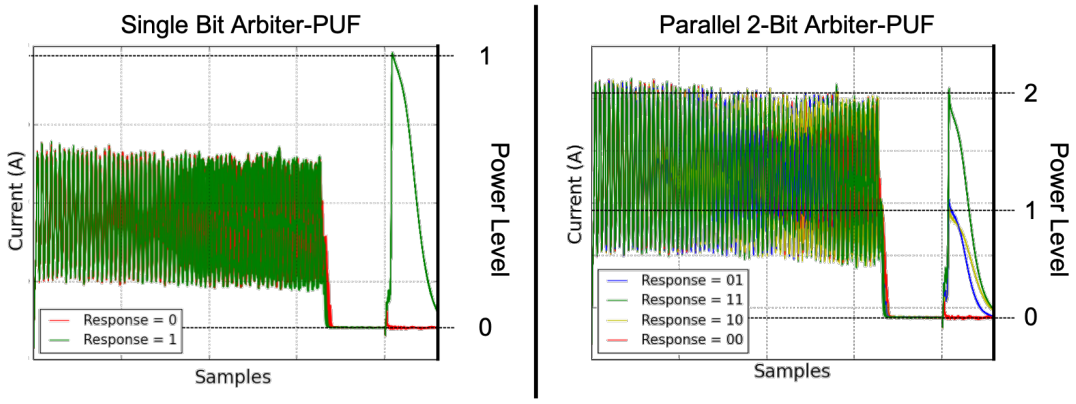


Figure 5.5: Hamming weights overlaid on the power traces of a single-bit PUF and two-bit parallel PUF.

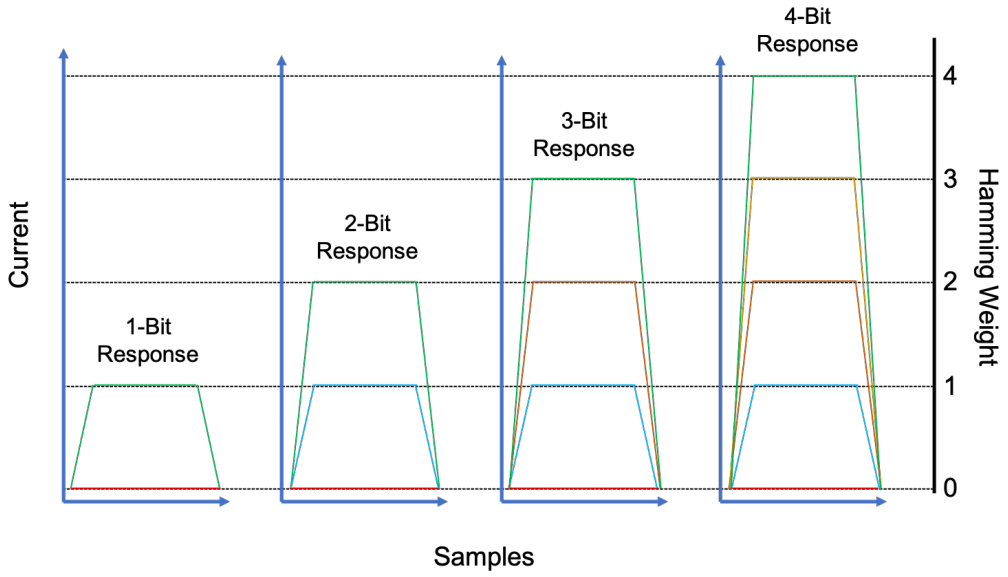


Figure 5.6: Predicted hamming weights for the power traces at each Flip-Flop leakage level for increasing the number of multi-bit responses.

revealing which responses may be misclassified by the model. The model should be able to find what the hamming weight of the response vector is and further modeling can occur to successfully predict the response.

Indeed the hamming weight prediction does occur as can be seen in the traces for a four-bit parallel PUF as seen in Figure 5.7. Correspondingly the results of attacking said PUF are shown in Figure 5.8. These results show that increasing the

size does in fact create the predicted hamming weight bins and also increasing the size of the PUF hampers the attack in its own right, however it is still vulnerable to a sophisticated attacker.

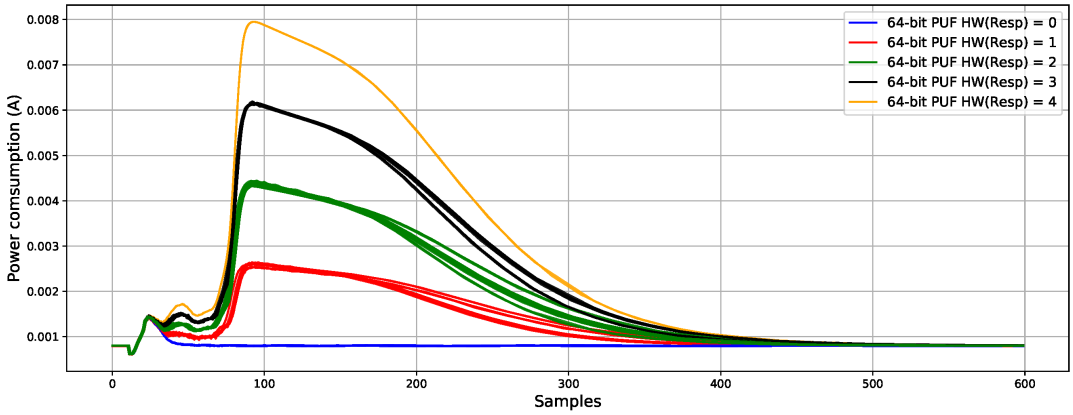


Figure 5.7: Superimposing 50 traces of the 64-bit PUF in 4-bit parallel settings. Note that *HW(Resp)* denotes the hamming weight of a 4-bit response *Resp*.

Although parallel PUFs are more difficult to attack however due to area and power overhead constraints large scale parallel PUFs cannot be readily implemented, i.e., the number of the 1-bit PUFs are limited. The attacks can also be enhanced through leveraging averaging techniques to increase the SNR and the overall attackablity of the PUF, such as the averaging of the power traces. This averaging technique creates a further basis for investigating specific countermeasures for power-based modeling attacks (as is done in Chapter 7).

5.5 Contributions

The results of the power-based modeling of multi-bit parallel PUFs showed that they could be successfully modeled through their power traces. It was also shown that the accuracy of the power-based models were increased through averag-

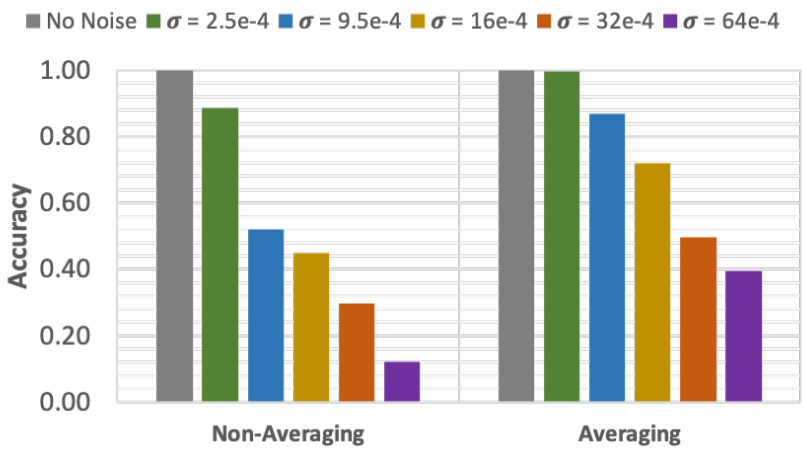


Figure 5.8: The attack accuracy targeting a 4-bit response arbiter-PUF for various noise levels. Note the accuracy for randomly guessing the output of a 4-bit PUF is 6.25%.

ing multiple power traces from the same challenge to increase the SNR of the attack. Furthermore it was observed that the power traces realized in the Flip-Flop can be characterized by the hamming weight of the response vectors which serve to reveal characteristics of the response even when a misprediction of the model occurs.

Chapter 6: Cross-PUF Attacks

Normally when a PUF is attacked through its power traces the developed model is used to attack the same PUF from which they were collected. In this thesis we moved one step further and showed the feasibility of the Cross-PUF attacks in which the power traces used to create the model are collected from one PUF and the model is then used to attack another PUF. We first proposed these Cross-PUF attacks in [49] as novel investigations into the PUF's vulnerability to said attacks. This attack can be successful as the power traces that are extracted are very similar for PUFs realized from same GDSII file.[1] The ability to attack any PUF with a similar GDSII file gives the attacker the opportunity to build a model from a PUF in their possession and attack all others. The ability to model and attack a PUF in this way presents a significant vulnerability and urges the community to develop strong countermeasures against such attacks.

Cross-PUF attacks are successful due to the similarities in the leakages observed within the power traces from multiple PUFs (those that would be realized from the same GDSII file) seen in Figure 6.1. In this figure the traces for two PUFs are shown. At the end of the trace the leakages introduced by the Flip-Flop are

[1]The GDSII file contains all of the information including geometries used to create an IC.

very similar. Furthermore the leakages introduced by the arbitration latch are also similar, which provides motivation for attempting to attack one PUF with the model created from another.

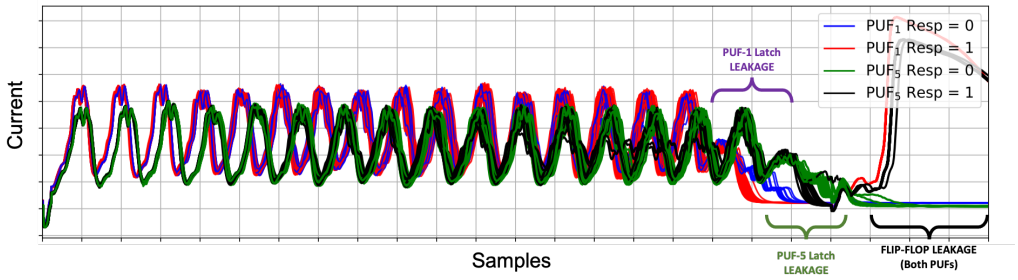


Figure 6.1: Power traces of two different PUFs superimposed to highlight their similarities.

Note that cross attacks cannot be performed with a PUF's CRPs considering the unique fingerprints (CRPs) of PUFs. Indeed, if this type of attack were possible then basis for using a PUF wouldn't exist as they would not produce unique values.

6.1 Threat Model

The threat model for Cross-PUF attacks is quite interesting since the attacker does not need access to the device prior to the attack. When the manufacturer sells the IC, the adversary can acquire one of the devices and create a power-based model. Other devices, made from the same GDSII file, will get sold and integrated into critical systems. When the adversary comes across another IC with the PUF realized from the same GDSII file, the previously created model can be deployed and attack said device. This threat model is shown in Figure 6.2. This ability removes a barrier for the attacker as they only need momentary access to the device to perform their attack.

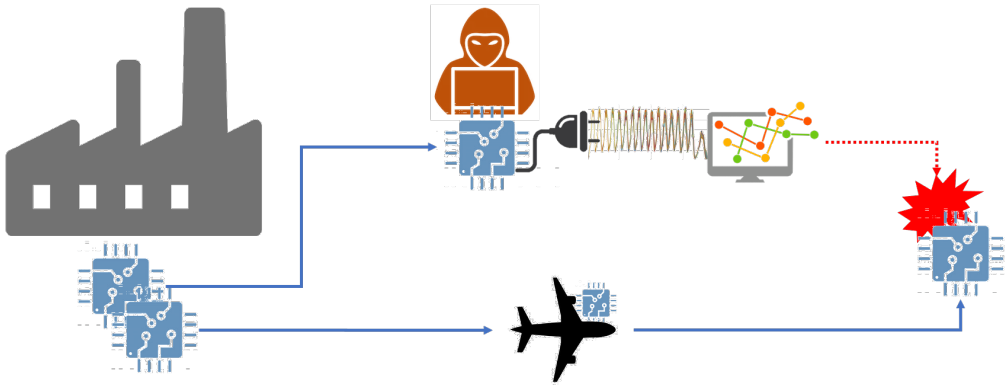


Figure 6.2: Threat model for the Cross-PUF attack.

6.2 Cross-PUF Attack Results

For the cross-PUF attacks five 16-bit arbiter-PUFs that all were realized from the same design were simulated (see Figure 4.5 for the diagram of the arbiter-PUF with system components). In these results both the arbitration latch and the response Flip-Flop were targeted, their contributions to the power traces are shown in Figure 4.7.

Attacks in the Absence of Noise

The results of Cross-PUF attacks using the leakage from the arbitration latch are shared in Table 6.1. The accuracies highlighted in the diagonal of the table are the results of Self-PUF attacks on the PUF (attacking the PUF with a model from its own power traces). The other accuracies are for Cross-PUF attacks on other PUFs. All of the Self-PUF attacks achieved accuracies above 99% reaffirming the latch's vulnerability to power-based modeling attacks. For the Cross-PUF attacks the accuracy ranges from 94% to greater than 99%, showing that the PUFs' arbiter latch are vulnerable to the novel Cross-PUF attacks. When performing the Cross-

PUF attacks on the latch the traces had to be aligned; that is they had to be shifted so the power trace samples corresponding to the latch leakage were presented to the deployed model in similar fashions. As seen in Figure 6.1 the leakage from the latches of different PUFs is not aligned. This misalignment is due to the variations in delay before the arbitration latch. For Cross-PUF attacks to be successful the leakage from the latch had to be shifted so that they were appropriately aligned with the leakage of the modeled device. Without the realignment of the latch's power traces (as seen in Figure 6.3) the attacks failed.

Table 6.1: Accuracy of Cross-PUF attacks on the arbitration latch for each PUF pair with no added noise.

Modeled PUF	Traces used for Training	Attacked PUF				
		PUF_1	PUF_2	PUF_3	PUF_4	PUF_5
PUF_1	200	**0.9998**	0.9605	0.9965	0.9997	0.9483
PUF_2	200	0.9454	**0.9987**	0.9776	0.9517	0.9545
PUF_3	200	0.9735	0.9997	**0.9983**	0.9775	0.9494
PUF_4	200	0.9936	0.9700	0.9815	**0.9975**	0.9470
PUF_5	200	0.9880	0.9951	0.9640	0.9855	**0.9895**

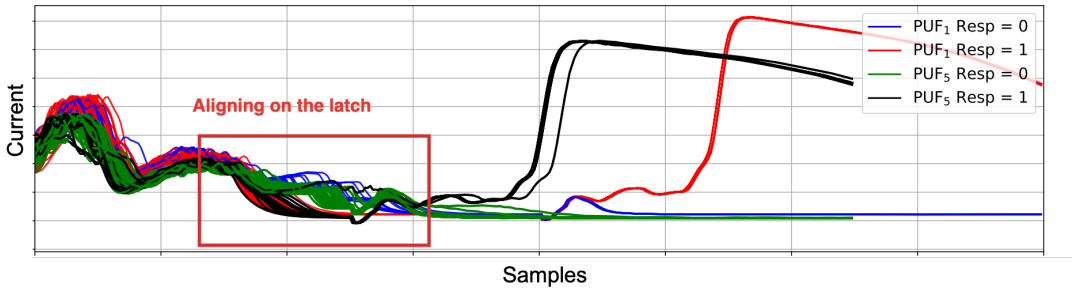


Figure 6.3: Zoomed traces from Figure 6.1, displaying the alignment (by shifting) on the latch.

Turning the attention to the Flip-Flop component, the power traces for a stated arbiter-PUF are shown in Figure 4.15 with the trace's response highlighted. It is apparent from this figure that the response outputs are clearly visible, as

previously shown, therefore this attack is trivial. This result reinforces that PUFs are highly vulnerable to power-based modeling attacks perpetrated on the leakage on Flip-Flop.

Attacks with Added Noise

To simulate the effects of real circuits the same five PUFs were attacked with added noise. To add a level of realism, gaussian noise with σ ={2.5e-4, 9.5e-4, 16e-4, 32e-4, 64e-4} was added to the power traces post simulation. In these results 2,000 traces were used for training and tested was performed against 11,000 traces.

The results of Cross-PUF attack on the latch are shown in Figure 6.4 for the various levels of added noise. In these attacks PUF-1 was used to develop the model which was deployed on the remaining PUFs. With added noise with σ =2.5e-4 the attack was successful with Cross-PUF attack accuracies of 92% or more. The accuracy significantly drops at the higher noise levels, likely due to the noise completely masking the signal in the arbitration latch.

When focusing the attack on the leakage of the Flip-Flop the accuracy of attack is 100% for σ ={2.5e-4, 9.5e-4, 16e-4}. When σ =32e-4 the accuracy is still 99% as shown in the graph of Figure 6.5. When the noise reaches σ =64e-4 the accuracy drops to between 82% and 89%. These results show that the Cross-PUF attack targeting the response Flip-Flop is possible even in the presence of high levels of noise. This observation reinforces previous notions that the Flip-Flop leakage is an effective attack point on PUFs for power-based modeling attacks, showing that the leakage from the Flip-Flop is vulnerable to Cross-PUF attacks in the presence of noise.

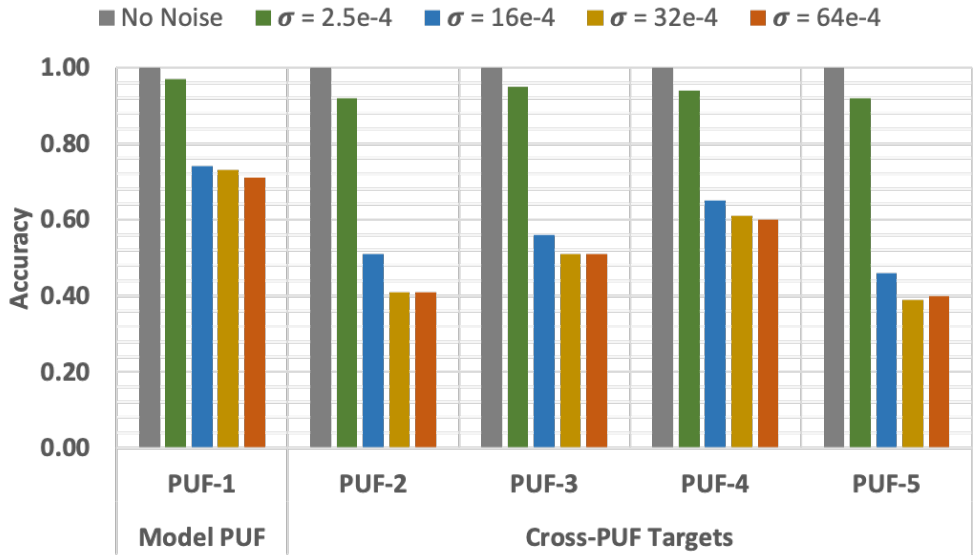


Figure 6.4: Accuracy of Cross-PUF attacks on the arbitration latch for various levels of noise. PUF-1 was used to create the model and deployed against the other PUF instances.

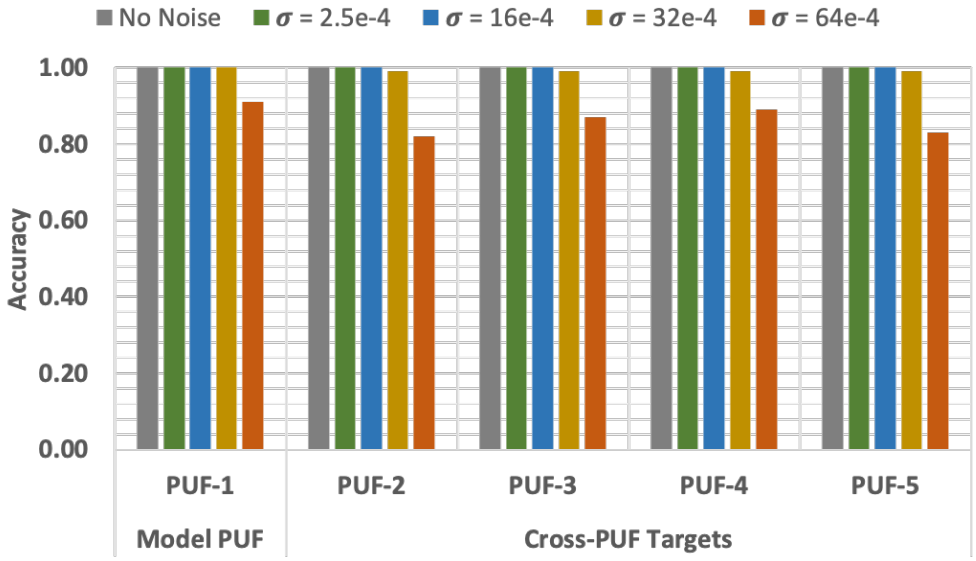


Figure 6.5: Accuracy of Cross-PUF attacks on the Flip-Flop for various levels of noise. PUF-1 was used to create the model and deployed against the other PUF instances.

To characterize the level of noise present in these attacks the SNR was calculated for each of the different levels of noise (see Table 6.2). Recall that Fukushima et al. [29] showed that the SNR of a silicon implementation of an arbiter-PUF is 1.81.

The SNR for the latch when there were successful Cross-PUF attacks at σ =2.5e-4 was 0.235. For the successful attack on the Flip-Flop at σ =32e-4 the SNR was 0.078. Both of these levels of SNR are much lower than those seen in real silicon implementations.

Table 6.2: The maximum SNR for the traces related to the PUF-1's latch and Flip-Flop.

	σ = 2.5e-4	σ = 16e-4	σ = 32e-4	σ = 64e-4
Latch	0.235314	0.009083	0.002350	0.001593
Flip-Flop	12.320019	0.308742	0.079990	0.022701

6.3 Temperature effects on Cross-PUF Attacks

The objective of this section is to see if the Cross-PUF attack is feasible when there is a temperature misalignment between the modeled PUF and the target one. We investigate the effects of this temperature mismatch on the power-based modeling of PUFs as well as the Cross-PUF attack [49]. This investigation is critical due to the fact that an adversary attempting to model the PUF will likely not perform the Cross-PUF attack when the devices are at the same temperature (indeed this is true for Self-PUF attacks as well). This temperature misalignment may cause failures in the modeling attack due to the changing behavior of the PUF at different temperatures; thus need to be investigated. The temperature effects on the power traces for a single PUF are shown in Figure 6.6.

In Figure 6.6 it can be observed that the power traces for the PUF are contracted in lower temperatures, i.e., the PUF operates faster. Despite the fact that these traces are for the same PUF, the leakage induced by the latch occurs at a different period in time. To achieve success for the Self-PUF attack the traces

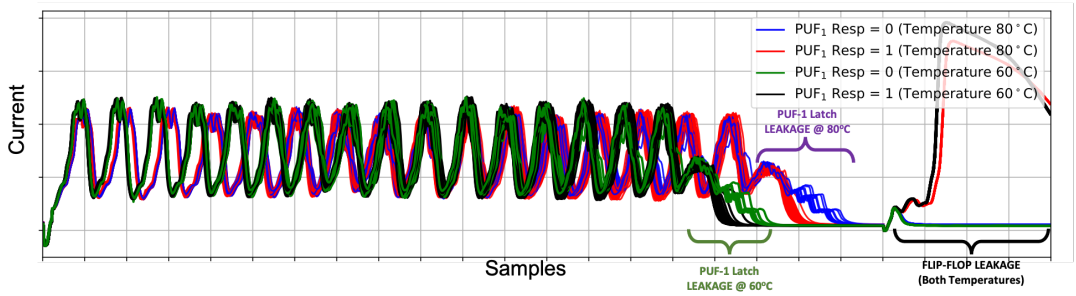


Figure 6.6: Superimposition of 50 traces of PUF-1 under different temperatures to observe the similarities in the collected traces.

have to be shifted to align the latch leakages similar to the process used to make similar temperature Cross-PUF attacks successful. Since the effect of temperature misalignment on the latch leakage is similar to the innate variation between PUFs, the same shifting process can be used for Cross-PUF attacks on devices of different temperatures. The leakage created by the Flip-Flop still occur with the rising clock edge therefore temperature has little effect on the attack.

The results of attacking the latch and Flip-Flop are shown in Figure 6.7 and Figure 6.8 respectively. These results show that the Self-PUF attacks, on both the latch and the Flip-Flop, at different temperatures achieve similar accuracies for all levels of noise. Comparing the results of the Cross-PUF attack on the latch from Figure 6.4 to those presented in Figure 6.7 indicate there is little difference in the resulting accuracies due to the temperature misalignment. This observation is similar when comparing the results in Figure 6.8 to those from Figure 6.5. These results serve to show that temperature misalignment has little effect on the ability to perform power-based modeling attacks as well as Cross-PUF attacks.

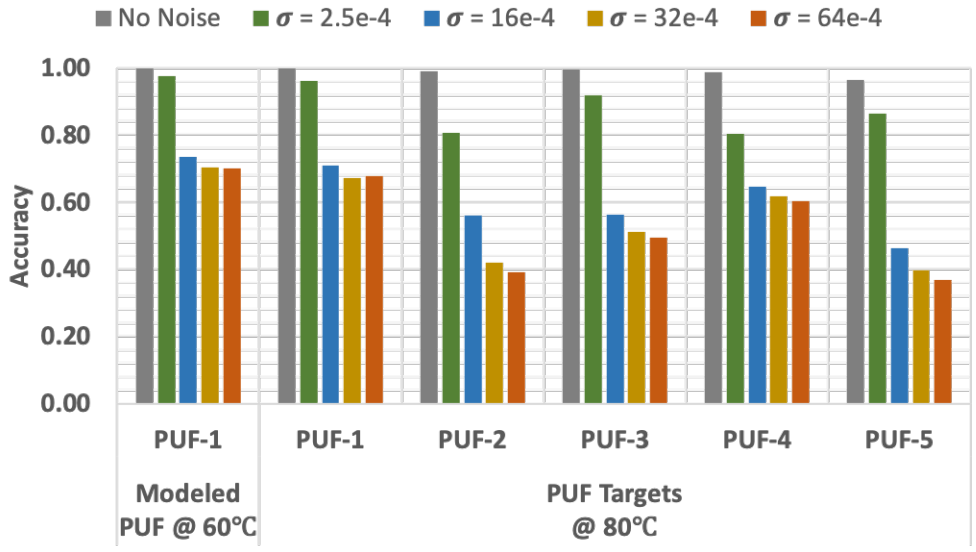


Figure 6.7: The temperature misalignment modeling results for the Self-PUF and Cross-PUF attacks targeting the leakage from the arbitration latch.

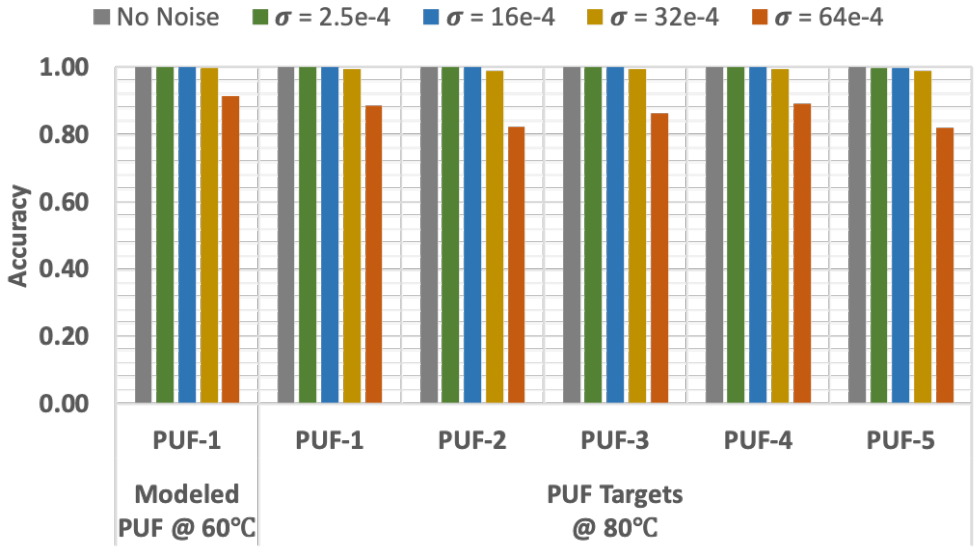


Figure 6.8: The temperature misalignment modeling results for the Self-PUF and Cross-PUF attacks targeting the Flip-Flop leakage.

6.4 Aging Effects on Cross-PUF Attacks

Similar to the temperature misalignment, the effects of aging must be investigated. Aging misalignments are less controllable than temperature as the attacker

may not be aware of the exact operation of the device or its overall usage, therefore aging misalignments between the target device and the device used as a reference are inevitable when performing the Cross-PUF attacks. The results of the attack on the latch are shown in Figure 6.9 and the results of attacking the Flip-Flop are shown in Figure 6.10. For these experiments the reference PUF was trained using 1,000 power-traces and tested against 11,000 power-traces. The reference PUF (PUF-1) was un-aged, whereas PUF-2 was aged from 0 to 24 months in two month increments. Comparing the Latch results of Figure 6.9 each successive two month result shows little variability in the success of the attack. There is a slight decrease in the accuracy of the attack for traces with noise of σ =2.5e-4 going from 93% un-aged to 88% at an age of 24 months. Furthermore, comparing these results with those presented in Figure 6.4 shows that the accuracy of attack is very similar to those of the unaged attack. The Flip-Flop results, shown in Figure 6.10, indicate that the accuracy of the attack is much greater than those seen by attacking the latch. It is similar in the fact that the results do not change from those seen from the attacks absent of aging as compared to Figure 6.5. In fact the attack is so resilient that there are negligible effects from aging on the accuracy when attacking the Flip-Flop.

These results show that aging does not have a large effect on the Cross-PUF attacks of either the latch or Flip-Flop. They also confirm that the Flip-Flop remains the better target for attack. Here it is important to recall that in Figure 4.18 the traces of highest variability were from the young or new PUFs, therefore utilizing the un-aged traces of the reference PUF, as was done here, should result in a worst case scenario for these attacks.

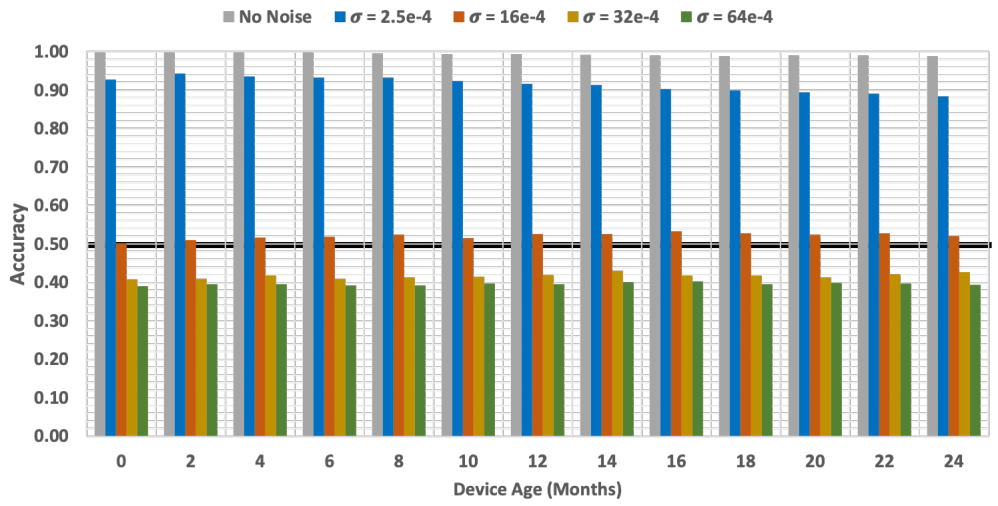


Figure 6.9: The modeling accuracy for the *Cross-PUF* attacks on the original PUFs targeting the latch at 80°C in presence of aging misalignments. The model was built based on the power traces of the un-aged PUF-1 and used to attack PUF-2 operating at the same temperature.

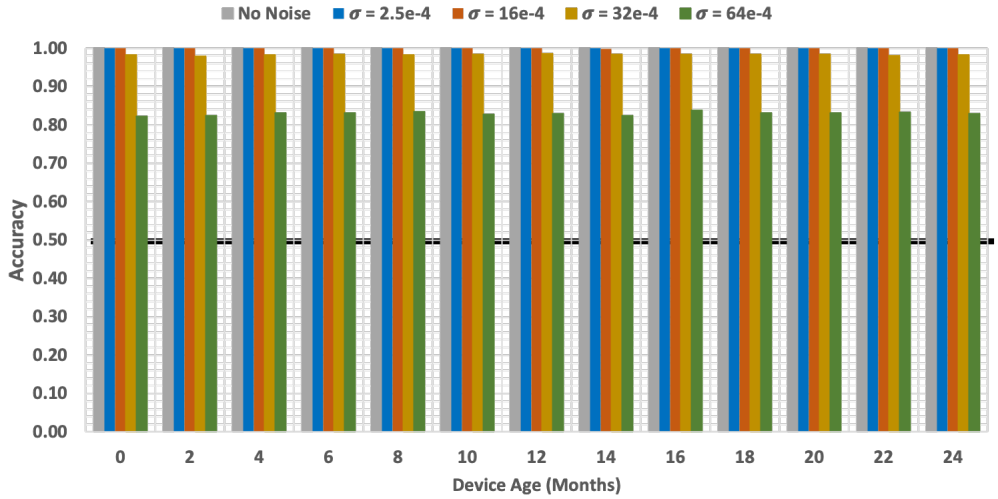


Figure 6.10: The modeling accuracy for the *Cross-PUF* attacks on the original PUFs targeting the Flip-Flop at 80°C in presence of aging misalignments. The model was built based on the power traces of the un-aged PUF-1 and used to attack PUF-2 operating at the same temperature.

6.5 Temperature and Aging Effects on Cross-PUF Attacks

In the previous sections the effects of aging and temperature were explored in relation to the Cross-PUF attack. Here the effect of both of these environmental

conditions are taken into account simultaneously. It is plausible that the attacker may in fact not be able to align both the temperature and the age of the reference PUF and the target PUF, therefore investigating the impact of attacking a PUF in presence of both of these misalignments is required.

The results of attacking the latch with the combined misalignment are shown in Figure 6.11. Interestingly there seem to be some effect on the attack in this case which was not the case separately (see Figure 6.7 for temperature and Figure 6.9 for aging). As seen in Figure 6.11 the accuracy of the high levels of noise remain the same. However, the accuracy of the attack on traces with noise at *sigma* =2.5e-4 decrease from 90% down to 75% displaying a rather significant decrease.

For the Flip-Flop the results (see Figure 6.12) are similar to just temperature misalignments (Figure 6.8) and aging effects (Figure 6.10). Here the result of attacking the Flip-Flop is unaffected by the combined misalignments.

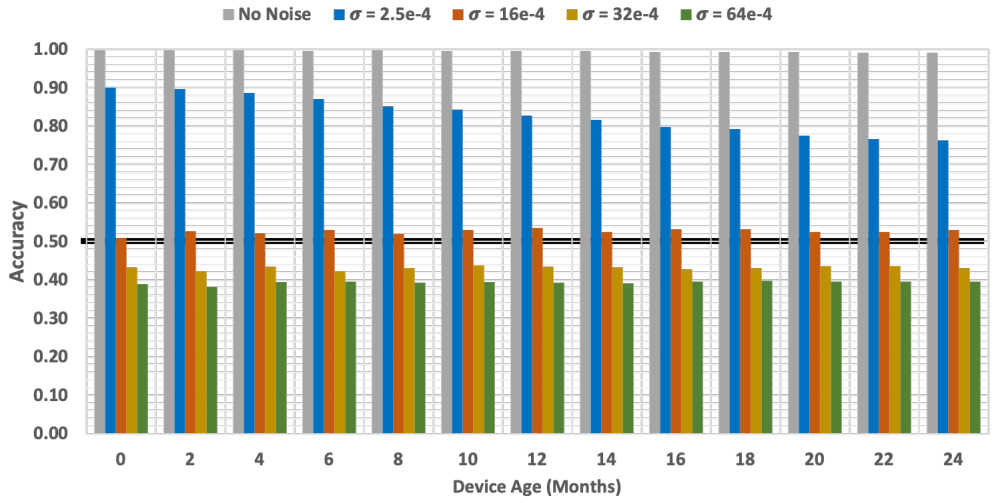


Figure 6.11: The modeling accuracy for the *Cross-PUF* attacks on the original PUFs targeting the latch in presence of both temperature and aging misalignments. The model was built based on the power traces of the un-aged PUF-1 in 60°C, and tested based on the traces extracted from the aged PUF-2 operating at 80°C.

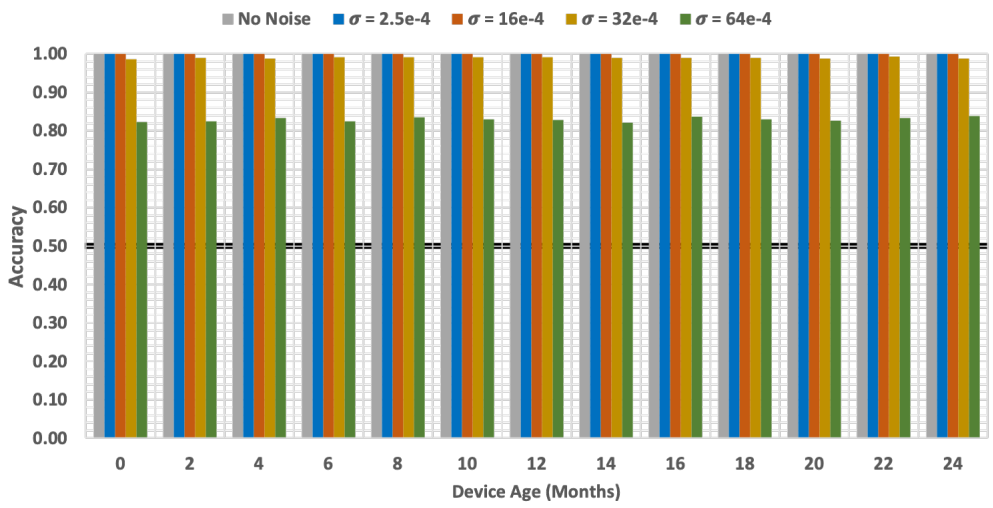


Figure 6.12: The modeling accuracy for the *Cross-PUF* attacks on the original PUFs targeting the Flip-Flop in presence of both temperature and aging misalignments. The model was built based on the power traces of the un-aged PUF-1 in 60°C, and tested based on the traces extracted from the aged PUF-2 operating at 80°C.

The takeaway from these results is that the accuracy of attacking the latch is slightly effected by the combined effects of temperature and aging misalignments whereas the Flip-Flop attack results are mostly unaffected. These results make the proposed attacks highly applicable according to the considered threat model, where the adversary does not have to take any control of the temperature or age of the target PUF, especially when targeting the Flip-Flop.

6.6 Discussions

Cross-PUF attacks on the arbiter-PUF proved to be successful even in the presence of considerable noise showing that these attacks can be performed on PUFs that are created with the same GDSII file. They also confirm that both the latch and the Flip-Flop can be attacked, and certify that the Flip-Flop is the superior

attack point as displayed in the high levels of accuracy present in both the Self-PUF and Cross-PUF results. Since these components (arbitration latch, output Flip-Flop) are present in the derivatives of the arbiter-PUF, the Cross-PUF attack is a vulnerability present in all of them.

Validity is added to the attacks on simulated traces by observing that the attacks are performed at noise levels much higher than what would be seen in real devices. This success provides confidence that the attacks would be viable if performed on silicon implementations of the PUF.

Furthermore, Cross-PUF attacks are highly successful despite having a misalignment in temperature and age between the modeled and attacked PUFs. This observation makes the attack more realistic as the adversary may not be able to control the temperature of the target PUF nor the age difference between the devices.

6.7 Contributions

This section showed the effectiveness of power side-channel modeling based attacks where the the target PUF and the PUF used for modeling are different yet realized from the same GDSII file. This attack proved to be successful even when the noise level is high and when there are temperature and age misalignments between the devices. Finally the attack can be expanded to other derivatives of the arbiter-PUF such as XOR-PUF and FEED-FORWARD PUF by targeting the flip-flop in both structures would result in a similar attack accuracy.

Chapter 7: Countermeasures to Power-based Modeling Attacks

It has been shown that power-based modeling attacks are extremely powerful in attacking PUFs. These attacks exploit the leakage of the arbitration latch and that of the Flip-Flop component storing the PUF response. This Flip-Flop, as previously shown, is the component of greatest vulnerability to power-based modeling attacks when compared to the leakage from the latch. Due to the fact that an attacker will focus on the most exposed attack vector, the countermeasures concentrate on mitigating the vulnerabilities created by the Flip-Flop storage component.

In addition to launching successful Self-PUF and Cross-PUF attacks, this research develops efficient countermeasures to tackle such attacks. There are two avenues to investigate mitigation of power-based modeling attacks. The first avenue is to reduce the SNR of the leakage to levels that are indistinguishable by the modeling algorithm. Reducing the SNR seen in the power traces sets out to ensure that the modeling algorithm is unable to characterize the responses since the traces can reveal less about the eventual response. The second avenue, to thwart power-based modeling attacks, is to increase the unpredictability of the responses through techniques that serve to confuse and poison the modeling program. In this technique the model becomes corrupted by inputs that do not logically make sense with data

presented to and/or learned by the model.

The proposed countermeasures either attempt to reduce the SNR or increase the randomness of the response output. The following are descriptions of the countermeasure and the results of their implementation on PUF circuitry. Due to the fact that all silicon implementations of PUFs will have systematic noise the following levels of gaussian noise were added to the power traces post simulation: σ ={2.5e-4, 16e-4, 32e-4, 64e-4}. For a single bit response PUF the ideal mitigation accuracy is 50%. At this accuracy the model is effectively guessing the response therefore it is the ideal accuracy for protected single bit PUF.

7.1 Self-PUF Countermeasures

7.1.1 Dual Rail Logic

Our first countermeasure is based on using Dual Rail Logic (DRL). The block diagram for the methodology is shown in Figure 7.1. The S-R latch that makes up the arbiter in the arbiter-PUF typically has a primary and complementary output (Q and $\bar{Q}$). In this mitigation method, both of these outputs feed separate Flip-Flops: one is used for the output of the Flip-Flop whereas the other is used as a decoy. The decoy Flip-Flop, by storing the inverted response, will present a leakage that is complementary to that of the genuine response output. The complementary leakage makes the power traces seen for a '0' and a '1' similar, and by doing so reduces their SNR.

In the implementation shown in Figure 7.1 there are design characteristics that

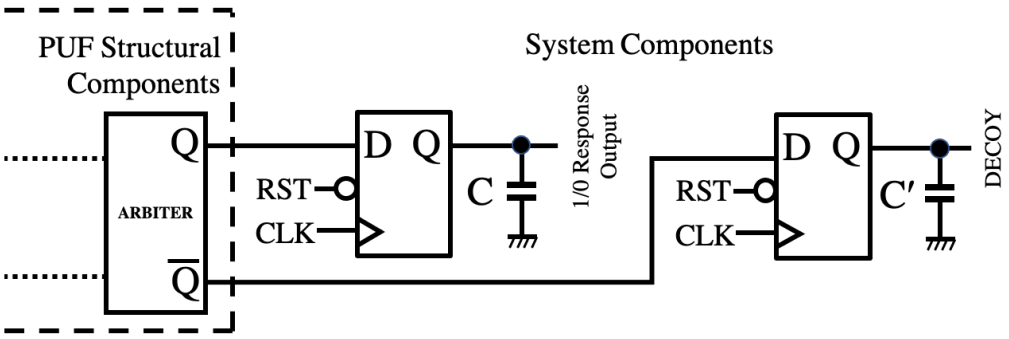


Figure 7.1: Block diagram for the DRL Countermeasure.

must be accounted for: the loading on the primary response and the loading of the decoy. To determine the best methodology for loading three cases were investigated: a low balanced load, a high balanced load, and an unbalanced load. The values of the capacitors simulating these loads are shown in Table 7.1. These loading characteristics will determine the importance of loading on the implementation of the DRL countermeasure. This countermeasure was inspired by methods in [62, § 7.3].

Table 7.1: Loading on the Capacitors for the DRL technique.

Loading	C **(fF)**	C'**(fF)**
Low	0	0
High	250	250
Unbalanced	250	0

7.1.2 DRL Results

The DRL countermeasure was implemented on a 64-bit arbiter-PUF and was assessed for levels of gaussian noise at σ ={2.5e-4, 16e-4, 32e-4, 64e-4}. In these results there were 500 power traces used for testing and developing the model and 5000 power traces were used to evaluate the model.

The results of the DRL countermeasure are shown in Figure 7.2. First, it can

be seen that there is considerably high accuracy of attack on the the unbalanced load, with near 100% accuracy for $\sigma \leq$32e-6. At the highest σ of 64e-6 the accuracy is still quite high at 88%. These results nearly identical to that of an unprotected arbiter-PUF implementation when compared with the results from Section 4.2.3. This result means that the unbalanced load provides no protection to the circuit. Looking towards the balanced loading scenarios it can be seen that both the high and low balanced loads are nearly identical. This observation means that the amount of loading does not matter. For the noiseless traces, the model still achieves 100% accuracy however in the presence of noise at σ =2.5e-4 the accuracy starts to drop to 93% for the low balanced situation and to 96% for the high balance situation. At higher levels of noise the accuracy drops closer and closer to 50%. This drop in accuracy shows that the DRL countermeasure can mitigate the power-based modeling attack as the accuracies are less than an unprotected arbiter-PUF. However, its mitigation is dependent having sufficient levels of noise in the circuit. Note that in real-silicon there is noise. Therefore, the cases of no-noise or very little noise are not realistic, and were used as basis for our experiments.

7.1.3 Randomized Initialization Logic

The next countermeasure is Randomized Initialization Logic (RIL). In this countermeasure a Flip-Flop that contains a set input is used to randomly set the Flip-Flop between challenge queries. This countermeasure involves connecting the set input to a pseudorandom number generator which is seeded by the challenge

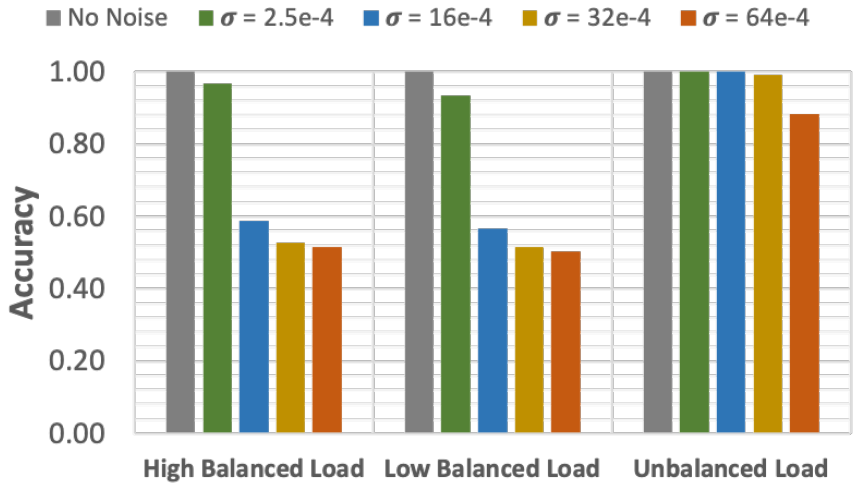


Figure 7.2: Results for implementing the DRL Countermeasure.

input. Note that the random bit can also be provided externally. The block diagram of this countermeasure is shown in Figure 7.3.

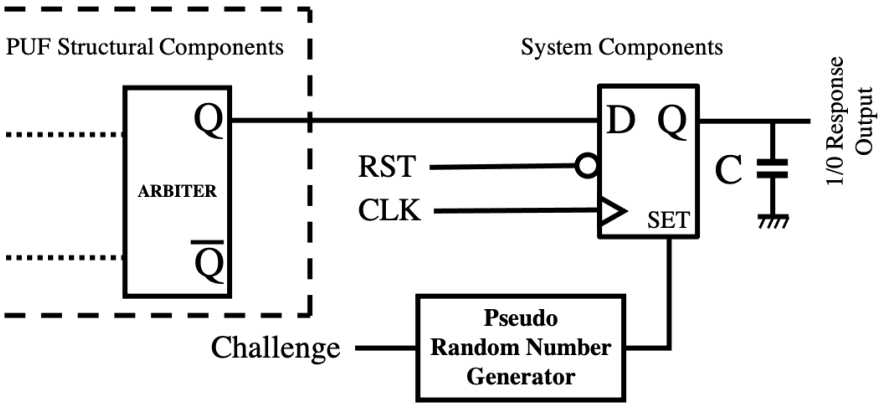


Figure 7.3: Block diagram for the RIL Countermeasure.

By randomly setting the Flip-Flop between challenges the characteristics of the leakage from the Flip-Flop change. Now instead of only transitioning from '0' to '0' or '0' to '1' it transitions from '1' to '0' and '1' to '1' as well. The high spikes in leakage come from transitions of the Flip-Flop, originally that is only when the response is a '1' (the transition from '0' to '1'). When the response is a '0' (the

Flip-Flop stays at '0') there is little leakage from the device. With random setting transitions can occur with the Flip-Flop being a '0' (going to a '1') or a '1' (going to a '0'). Also low levels of leakages will be seen for '0' to '0' and '1' to '1' situations. The additional transitions seen in the output serve to lower the SNR due to the assumption that the model will have difficulty determining if the PUF response is staying at '0' or '1', or if it is transitioning to or from a response of '1'.

7.1.4 RIL Results

The results of the power-based modeling attacks on a PUF with the RIL countermeasure are shown in Figure 7.4. These results are quite promising. The first observation to be made is that RIL protects the PUF even when there is no noise present; the accuracy is 79%. As the noise gets higher the attack accuracy decreases by about 5%. When at σ =64e-4 the accuracy is only 60%. This result shows that the countermeasure was successful at driving down the SNR to levels low enough to create difficulties for the model to distinguish the response clearly.

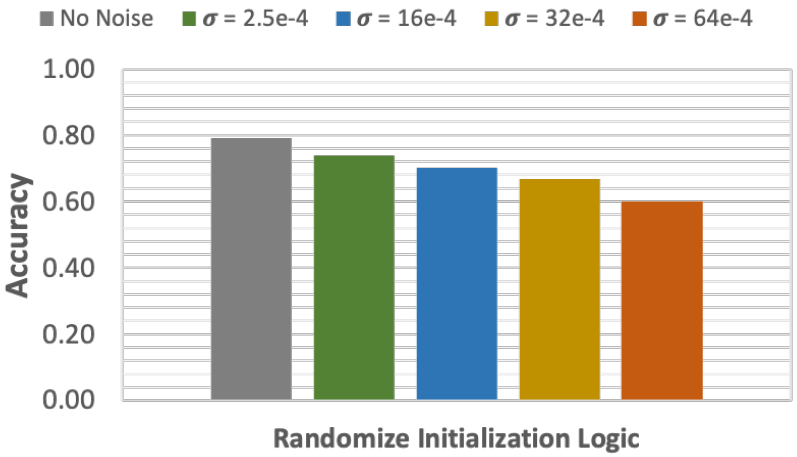


Figure 7.4: Results for implementing the RIL Countermeasure.

7.1.5 DRILL - Combined DRL and RIL Countermeasures

The DRILL countermeasure investigates the results of combining the DRL and RIL countermeasures to determine their combined effectiveness at mitigating power-based modeling attacks. The countermeasures were implemented together on a 64-bit arbiter-PUF. For the DRL portion of the circuit a balanced capacitance approach was chosen with both capacitors being set to $0fF$. The implementation of DRILL on the arbiter PUF is shown in Figure 7.5, with the portions of the circuit pertaining to DRL highlighted in blue and those for RIL highlighted in yellow. For the RIL connections it is important to note that the set input on both of the Flip-Flops is directly connected to the pseudorandom number generator. These connections mean that the Flip-Flops will be unset or set to the same value allowing them to retain their complementary values and complementary leakages for which they are valued in mitigated the attack.

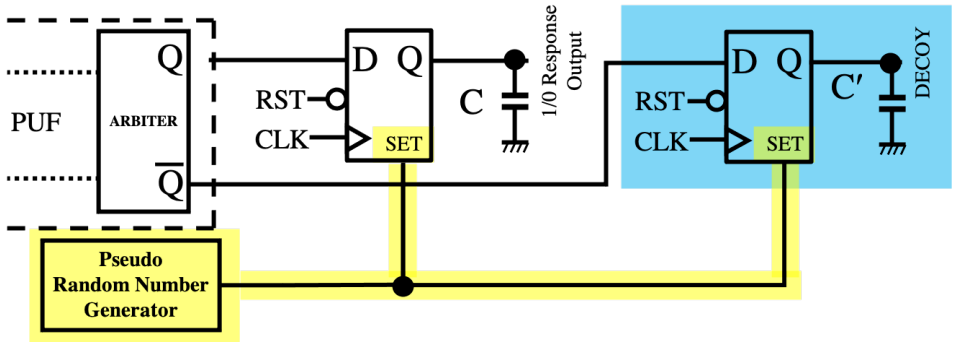


Figure 7.5: Block diagram for the DRILL countermeasure combining the techniques of the DRL (highlighted in blue) and RIL (highlighted in yellow) countermeasures.

7.1.6 DRILL Results

The results of the DRILL countermeasure are shown in Figure 7.6. The result of the attack when no noise is present is actually worse (i.e., higher accuracy of modeling) than when using the RIL countermeasure by itself. This result means that there are underlying features of the DRL technique that reveal the challenge. The difference is likely the difference in the process variations seen by the two Flip-Flops which can be clearly observed when there is no noise present in the circuit As mentioned earlier, in real circuits noise will always be present; thus the no noise case is unrealistic. When noise is introduced the countermeasure actually performs as expected. When σ =2.5e-4 the accuracy of the attack on the DRILL protected PUF is only 64%; a reduction of 10% lower than the RIL countermeasure showing that it is more effective at countering the attack with noise. For higher noise levels the accuracy of the attack is approximately 50% the level at which the model is effectively guessing the response, indicating the countermeasure operates as expected in high noise situations.

It is important to state that noise is not a countermeasure, it is a naturally occurring phenomenon which will always occur in real systems.

7.1.7 Discussions on Self-PUF Countermeasures

The DRL, RIL, and DRILL countermeasures were effective at mitigating the power-based modeling attack on the PUFs for which they were implemented. For DRL it is important to point out that the unbalanced load provided no protection to

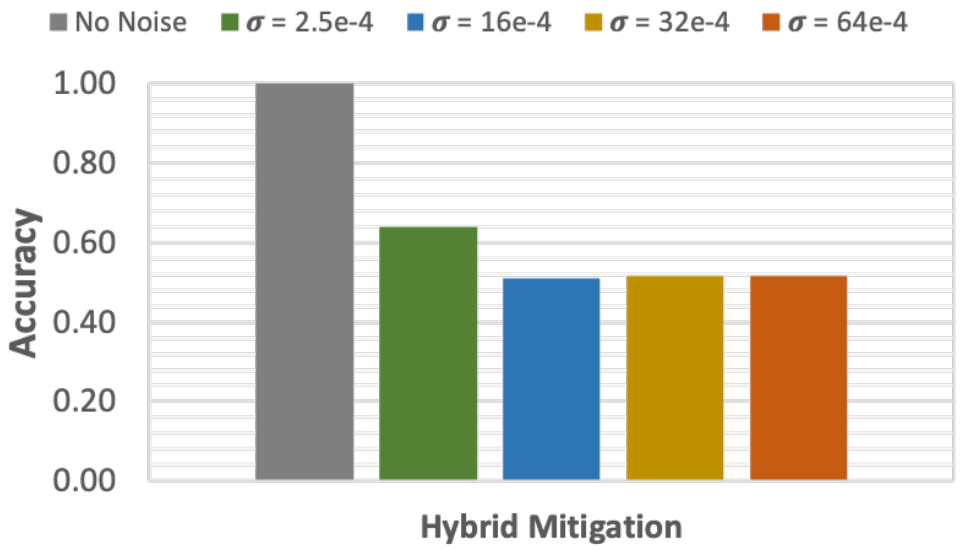


Figure 7.6: Results for implementing the DRILL Countermeasure.

the circuit. It can be observed in Table 7.2 that the SNR of the DRL countermeasure for the unbalanced loading situation and the unprotected arbiter-PUF are virtually the same corroborating its ineffectiveness. Both the high and low balanced load situations had nearly identical results. Together these facts show that the amount of loading does not matter so long as the loading is balanced. This observation is important since a portion of DRL is a decoy Flip-Flop. When implementing the PUF for use, this decoy may be forgotten and therefore may unintentionally be left unloaded creating an imbalance.

RIL provided protection even to the noiseless traces. The protection provided by RIL was successful at minimizing the SNR. As shown within Table 7.2, the levels of are significantly less than the SNR in the unprotected PUF. The difficulty in the RIL countermeasure is ensuring that the pseudorandom number generator's operation does not inadvertently reveal more information of the circuit[1].

[1]This is out of scope for these investigations.

Note that in real silicon the existence of noise is unavoidable. Therefore, our DRILL countermeasure is highly powerful in real silicon. Once noise was introduced to the circuit the countermeasure performed well providing even better protection than the RIL countermeasure.

Table 7.2: The maximum SNR for the traces when the Flip-Flop is queried in presence of the proposed countermeasures.

	σ = 2.5e-4	σ = 16e-4	σ = 32e-4	σ = 64e-4
Unprotected Arbiter-PUF	12.224361	0.299846	0.079410	0.021712
DRL H. Bal. Load	0.766351	0.020974	0.005704	0.001650
DRL L. Bal. Load	0.431002	0.010695	0.004800	0.001697
DRL Unbal Load	10.760024	0.285023	0.079133	0.022343
RIL	0.420705	0.068503	0.022047	0.007547
DRILL	0.034492	0.001739	0.000761	0.000970

7.2 Countermeasures for Multi-bit Parallel PUF Implementations

When dealing with implementations of multi-bit parallel PUFs the previously mentioned countermeasures need to be expanded. Also the circuitry within the parallel PUF presents an opportunity to develop additional countermeasures. The two new countermeasures are Randomized Arbiter Swapping (RAS) and Random Response Masking (RRM). These two countermeasure serve to attempt to confuse and poison the power-based modeling of the PUF by introducing more randomness in the response.

7.2.1 DRILL

Recall that the DRILL countermeasure combines the DRL and RIL countermeasures. This combination creates complementary switching Flip-Flops (from

DRL) and randomly sets the Flip-Flops to increase the switching characteristics that are observed by the registering the response (RIL). This technique again serves to reduce the SNR to make the response indistinguishable through the PUFs power traces. The implementation for the DRILL technique on a two-bit parallel response PUF is shown in Figure 7.7. Extension of the countermeasure is fairly straightforward as the countermeasure is just implemented on the outputs of both of the PUFs.

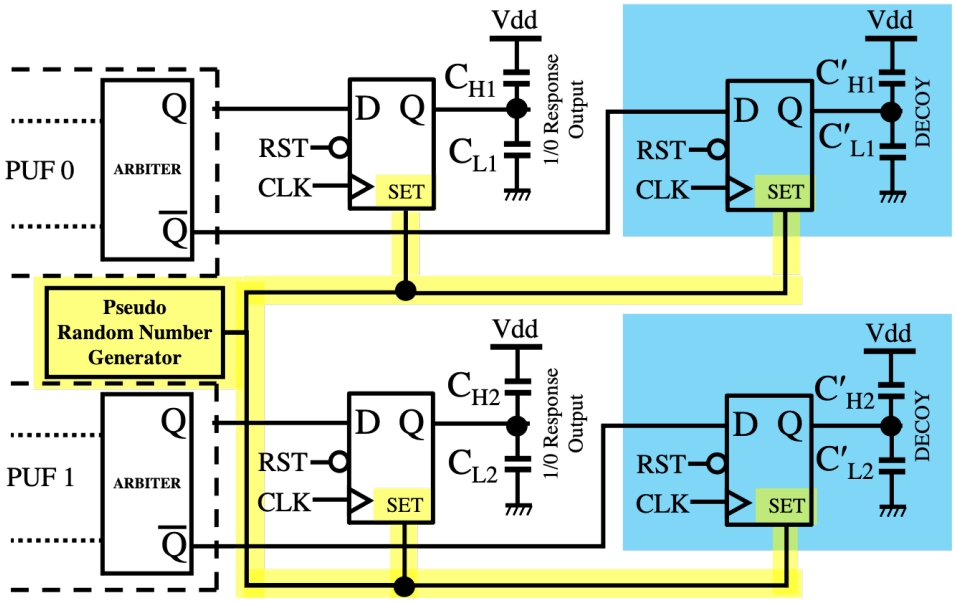


Figure 7.7: Block diagram for the DRILL countermeasure combining the techniques of the DRL (highlighted in blue) and RIL (highlighted in yellow) countermeasures. Shown for a multi-bit parallel PUF.

7.2.2 Randomized Arbiter Swapping

The RAS countermeasure takes advantage of the multiple PUF outputs in the parallel PUF. Each output of the PUF is fed into two multiplexers that randomly swap the outputs of the PUF. The select lines of the multiplexers are connected to

a pseudo random number generator to provide the random value for switching. If swapped then the response of PUF-0 becomes the response of PUF-1. This technique serves to poison the model of the PUF such that the response vector bits, '01' and '10' become indistinguishable by being randomly swapped. The swapping must be reversed during the processing of the response when it is being used.

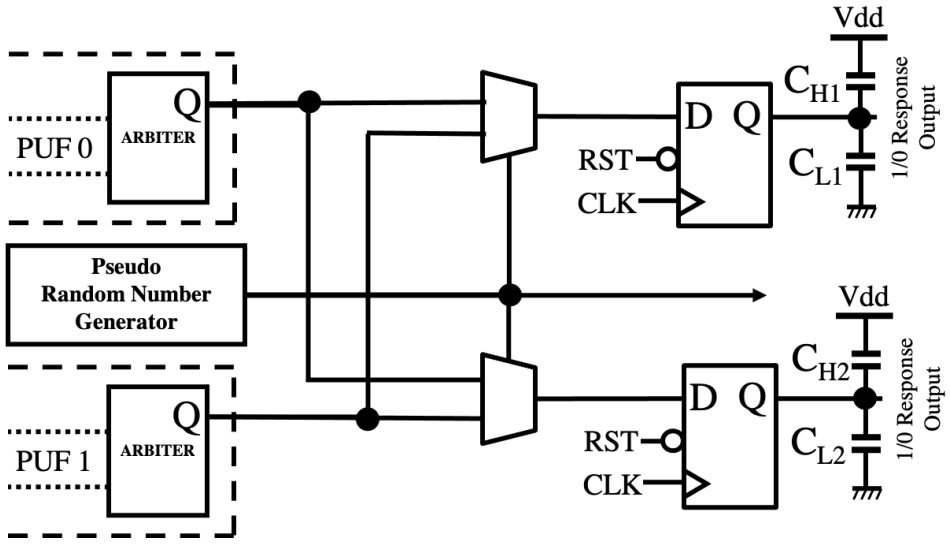


Figure 7.8: Block diagram for the RAS Countermeasure.

7.2.3 Random Response Masking

Figure 7.9 shows the block diagram for the RRM countermeasure. It takes the output of each of the implemented PUFs and XORs them with a bit randomly produced by a pseudo random number generator. This algorithm straightforwardly produces an output that is meant to confuse the modeling algorithm by randomly inverting the response value. The model should not be able to determine the response at a level that is beyond a chance guess. Like RAS the effects of the countermeasure on the response must be reversed in post processing of the response.

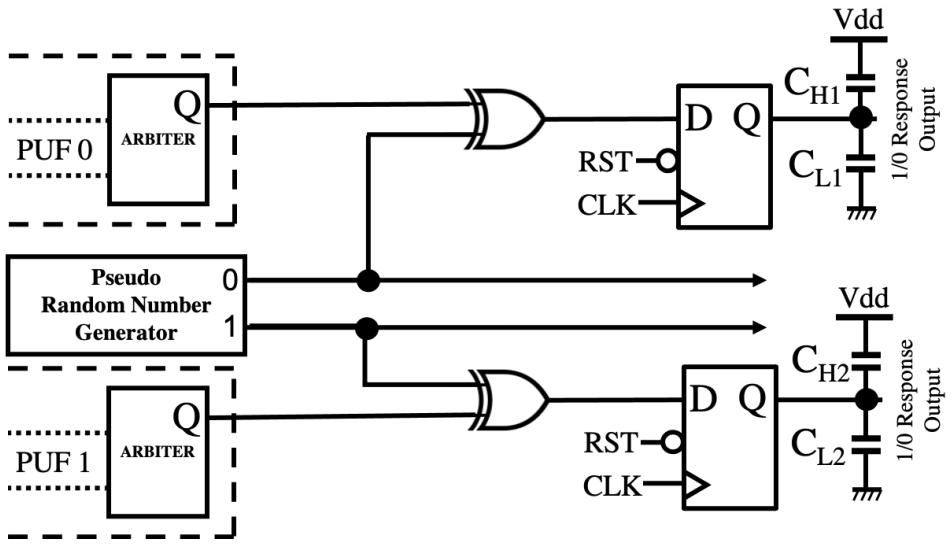


Figure 7.9: Block diagram for the RRM Countermeasure.

7.2.4 Results of Countermeasures for Multi-bit Parallel PUFs

This section discusses the results of the various countermeasures for a two bit response parallel PUF. Recall that the probability of guessing the correct response of a two-bit parallel PUF is 25%. At this accuracy the model is effectively guessing the response therefore it is the ideal accuracy for a protected PUF.

As previously discussed, an adversary can average the power traces for multiple trace collections of a single challenge to boost the SNR of the attack. This methodology is utilized here to assess the effectiveness of the countermeasures under this more sophisticated attack. The power traces within these results had added gaussian noise of σ ={2.5e-4, 9.5e-4, 16e-4, 32e-4, 64e-4} and all results were extracted from a 64-bit challenge two-bit parallel response arbiter-PUFs with the stated countermeasure.

DRILL Results

The results of the DRILL countermeasure for a two-bit parallel PUF are shown in Figure 7.10. The results of the non-averaging power trace modeling attack show that there is 100% accuracy when attacking traces with noise equal to and less than σ =2.5e-4. The higher level of noise experience modeling accuracies closer to the ideal protected circuit accuracy of 25%. This result shows that in sufficient noise, if the adversary cannot gather traces for averaging, DRILL works as a countermeasure.

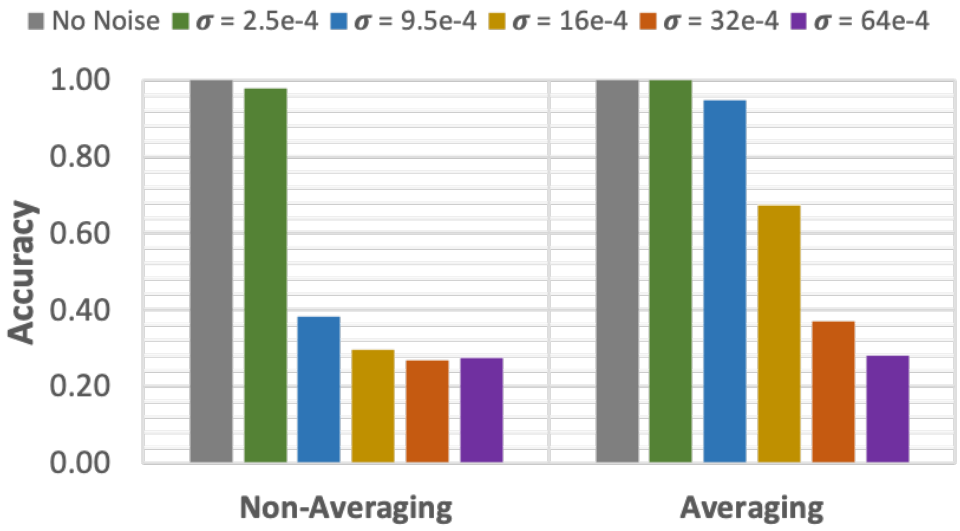


Figure 7.10: Results for implementing the DRILL countermeasure for the Multi-bit PUF.

When the adversary uses traces averaging to increase the SNR the results get worse as expected. For no noise and σ =2.5e-4 there is 100% accuracy in the attack. At σ =9.5e-4 the accuracy decreases slightly to 95% and then drops further to 67% at σ =32e-4 on its way to the ideal chance accuracy. Clearly DRILL is less effective when averaging is employed but the accuracy eventually heads towards 25% when there is sufficient noise.

Note that the DRILL countermeasure does mitigate the power-based modeling attack in real silicon measurement (in presence of realistic noise). As it will be seen (in Chapter 8, the operating noise of a real implementation corresponds to noise levels of σ >16e-4 and for these levels of noise DRILL preforms well.

RAS Results

The RAS countermeasure attempts to confuse/poison the model developed for predicting the response. The results of the model on a RAS protected two-bit parallel arbiter-PUF are shown in Figure 7.11. The non-averaging results show that there is an accuracy of about 75% across all cases except where the power trace is completely hidden amongst the noise at σ =64e-4 where the accuracy drops to 50%. Even when averaging is employed the accuracy is stable at 75%. This result makes sense given the understanding of the implementation of the countermeasure. The response values are being randomly swapped between the two PUFs therefore the response vector of '00' and '11' will be unchanged. If the modeling algorithm is effective it will correctly predict these two sets of outputs giving it an accuracy of 50%. The remaining response outputs are '01' and '10' which can be randomly swapped since there are just two outputs here the accuracy of guessing is these responses is 25%. It can be deduced that total model accuracy of 75%. In this case it is the combination of the accuracy of predicting '00' and '11' as well as guessing the '01' and '10' responses.

From these results it can be said that the RAS technique confuses the modeling algorithm but only for responses of dissimilar value.

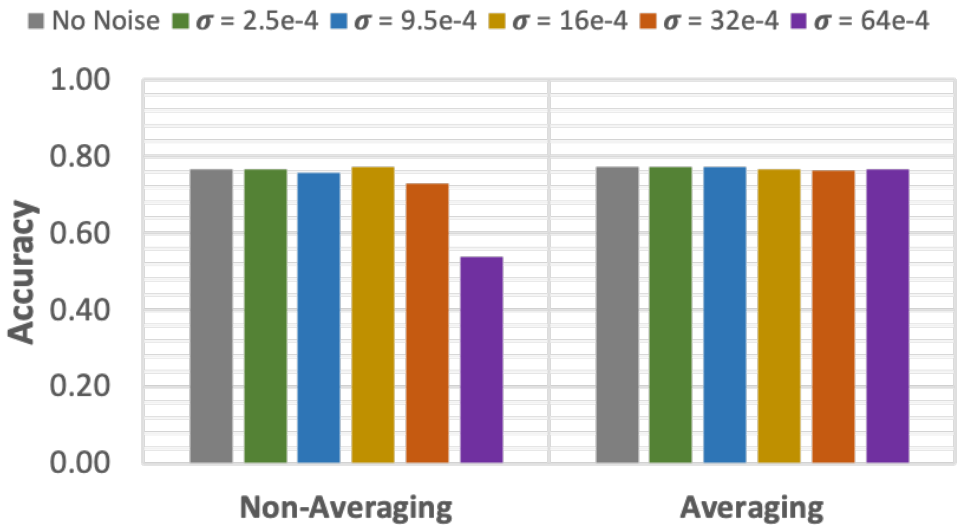


Figure 7.11: Results for implementing the RAS Countermeasure.

DRILL+RAS Results

To expand on the results protections offered through DRILL and RAS these results look at how they can be perform when used in tandem. By using both the goal is to utilize the benefit of the reduction in SNR from DRILL and the confusion techniques provided by RAS to reduce the modeling accuracy.

The results of these investigations are shown in Figure 7.12. When there is no averaging involved, the accuracy of the attack steadily decreases from an already low accuracy of 63% without noise towards the ideal protection accuracy of 25% with increasing noise. This result is very similar for the accuracies of the attack with averaging starting with the no noise situation with only 64% accuracy. The decline with increasing noise is more gradual however there is certainly diminished accuracy in the attack.

The similarity between the results with and without the averaging technique are exciting due to the fact that it means that there is a base level of protection

provided by the combined countermeasures. Furthermore, previous results showed that by using RAS only certain responses were capable of being protected. However, by combining RAS and DRILL, there is indeed difficulty in all of the responses being predicted.

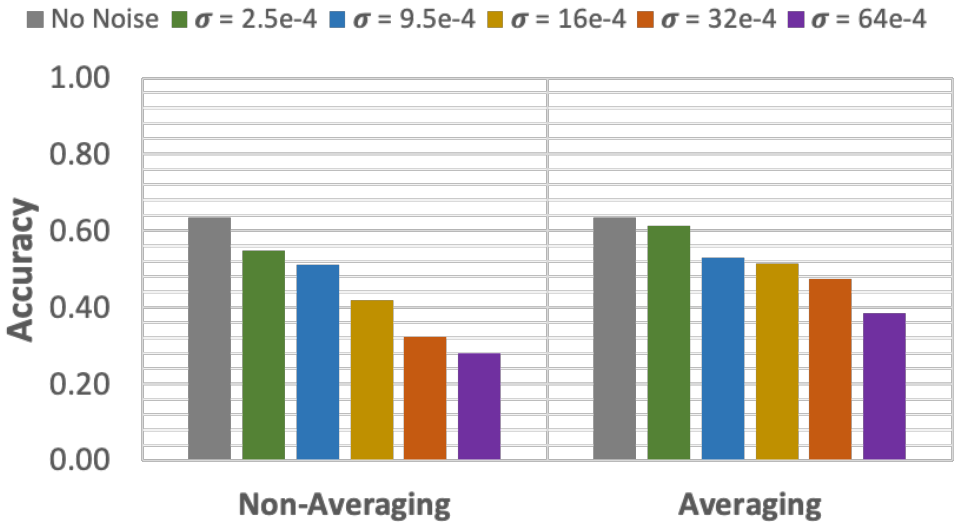


Figure 7.12: Results for implementing the combined DRILL and RAS countermeasures.

RRM Results

The RRM technique is setup to provide completely random masking of the response bits, which acts to poison the modeling algorithm with responses that will upset its predictions. Since this protection is an ideal situation for countermeasures, ideal chance prediction results are expected. As shown in Figure 7.13 the results of RRM are all the ideal chance guess accuracy of 25%. This result confirms that the best the model can do is guess the response. This methodology is highly effective at mitigating the attack.

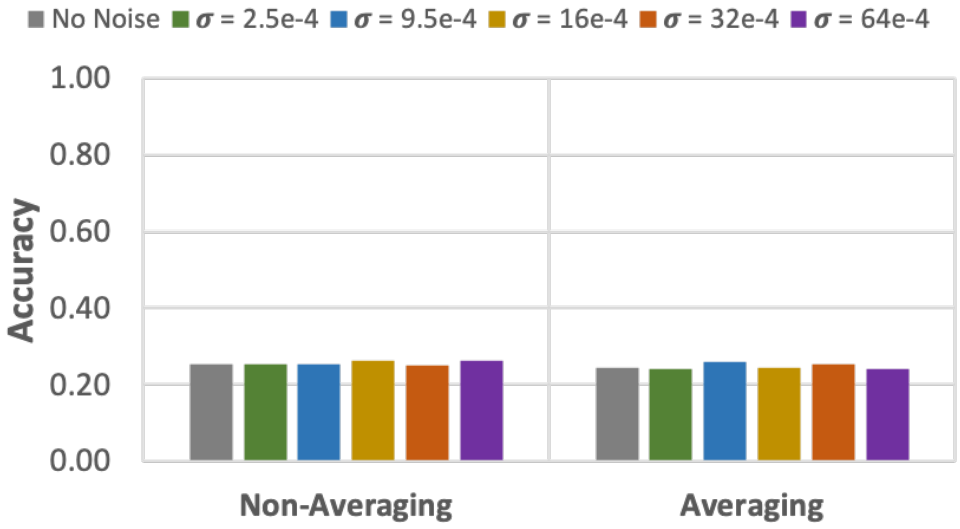


Figure 7.13: Results for implementing the RRM Countermeasure.

7.2.5 Discussions on Countermeasures for Multi-bit Parallel PUFs

The DRILL countermeasure proved to be successful at mitigating power-based modeling attacks when the noise level was σ >16e-4 when implemented on a two-bit parallel PUF. The countermeasure also successfully achieved its purpose of reducing the SNR of the Flip-Flop (as seen in Table 7.3) and was effective for high noise levels.

Table 7.3: The maximum SNR for the proposed DRILL Protected PUF with and without averaging.

	σ = 2.5e-4	σ = 9.5e-4	σ = 16e-4	σ = 32e-4	σ = 64e-4
Unprotected Non-averaging	9.713419	0.658853	0.232567	0.062253	0.016153
Unprotected Averaging	94.948925	6.666043	2.344949	0.589082	0.146669
DRILL Non-averaging	0.034776	0.00476	0.002327	0.001941	0.001105
DRILL Averaging	0.185653	0.025787	0.011142	0.004322	0.001901

For RAS and RRM it does not make sense to observe the SNR due to the fact that their basis as a countermeasure does not attempt manipulation of the PUF's SNR. In other words the SNR will be the same for these techniques. The

two methods of confusing the modeling algorithm are effective within the bounds of their technique. RAS will only effectively protect the prediction of response vectors of '01' and '10', and '00' and '11' will be revealed since swapping them provides no protection as they are the same. For RAS, the maximum level of protection of 75% was achieved. For RRM, all of the responses are protected and the results achieve the ideal accuracy of a countermeasure for a two-bit parallel response PUF of 25%. These accuracies assume that the bit or bits from the pseudo random number generator cannot be leaked.

Added to the options was the combined DRILL countermeasure with RAS. This option served to both reduce the SNR as well as poison the modeling algorithm. The combination of these two provided reasonable protection against power-based modeling attacks. Moreover, since the accuracies were stable between the normal modeling attack and the averaging modeling attack, the countermeasure provides some resiliency to this more sophisticated attack unlike either of the countermeasures alone.

The proposed countermeasures that reduce the SNR of the traces (DRL, RIL, and DRILL) are self contained to the PUF itself and can be used as is. These countermeasures are ideal for the instances where self containment is a requirement for how the PUF will be used. The countermeasures that are meant to poison the operation of modeling algorithm (RAS, RRM) all require some post processing off the PUF to reverse the effects of the countermeasure. These countermeasures are ideal for PUFs that need to remain compact and where there is extra processing power on the system using the PUF result. Moreover, all of our countermeasures

have a very low cost for implementation: the DRILL countermeasure adds only two Flip-Flops, RAS adds just two multiplexers, and RRM's overhead is an XOR gate per instantiated PUF. The low amount of additional logical components are added to the PUF thereby conserving the area and power overhead as much as possible.

7.3 Cross-PUF Attack Countermeasures

In this section the DRILL countermeasure is thoroughly investigated as a protection mechanism against Cross-PUF attacks. Before discussing the results recall that the DRILL countermeasure provides protection through the combined DRL and RIL methodologies which seek to reduce the SNR of the leakage within the power consumption of the device. These results are focused on attacks to the response registration Flip-Flop, as the Flip-Flop is an easier target than the arbitration latch.

7.3.1 DRILL Results

The results of protecting against Cross-PUF attacks using DRILL are shown in Figure 7.14. These results show that DRILL is successful at mitigating the attack when the noise is greater than $sigma$ =2.5e-4 after which the resulting accuracy drops from around 90% to between 35% and 68%. Notably the accuracy does not diminish at noise levels σ >16e-4 meaning that ML algorithm is randomly selecting the response of the PUF in these higher noise levels. Comparing these results to those of the unprotected results in Figure 6.5 it can be seen that the DRILL countermeasure certainly mitigates the Cross-PUF attack.

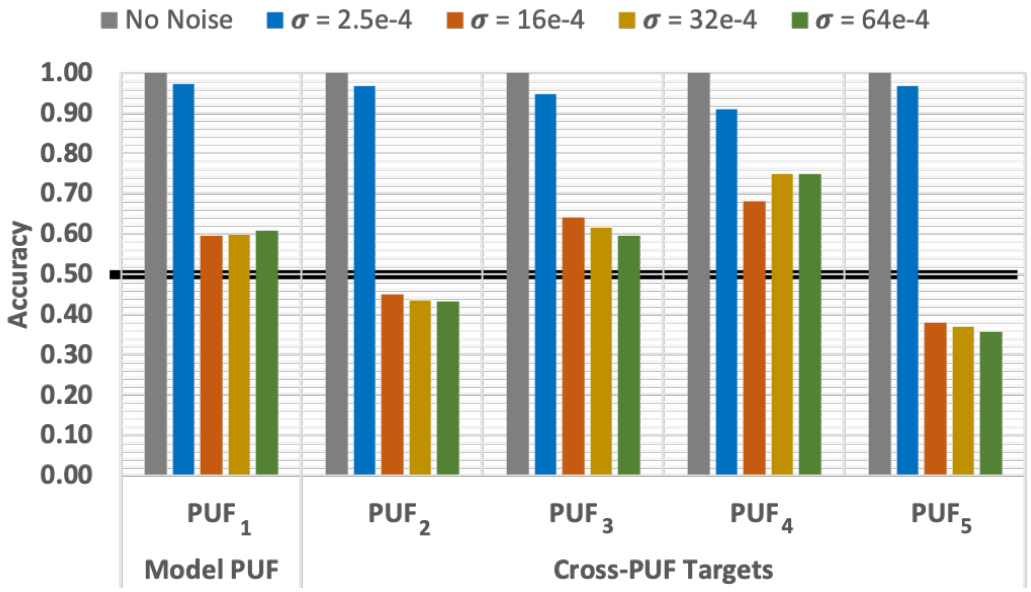


Figure 7.14: Cross-PUF attacks targeting the Flip-Flop in five PUFs equipped with the DRILL countermeasure in the presence of different noise levels. 1,000 power-traces of PUF-1 were used for training. 11,000 power traces were deployed for model validation in each case. All PUFs operate at 80°C.

7.3.1.1 Effects of Temperature and Aging Misalignment on Cross-PUF Attacks

Much like the previous results presented in Chapter 6, it is important to explore the uncontrollable environmental effects such as temperature and aging since the adversary may not be able to control these variables. First the effects of temperature variability are shown in Figure 7.15. In these results PUF-1 operating at 60°C is used to create the model and this model is used to attack all of the PUFs (including the Self-PUF attack) operating at 80°C. These results are very similar to those presented in Figure 7.14 showing that temperature has a trivial effect on the accuracy of the attack. Not only do these results show the lack of effect for each of the Cross-PUF

attack results but also they confirm this for the Self-PUF attack since PUF-1 at 80°C is also attacked.

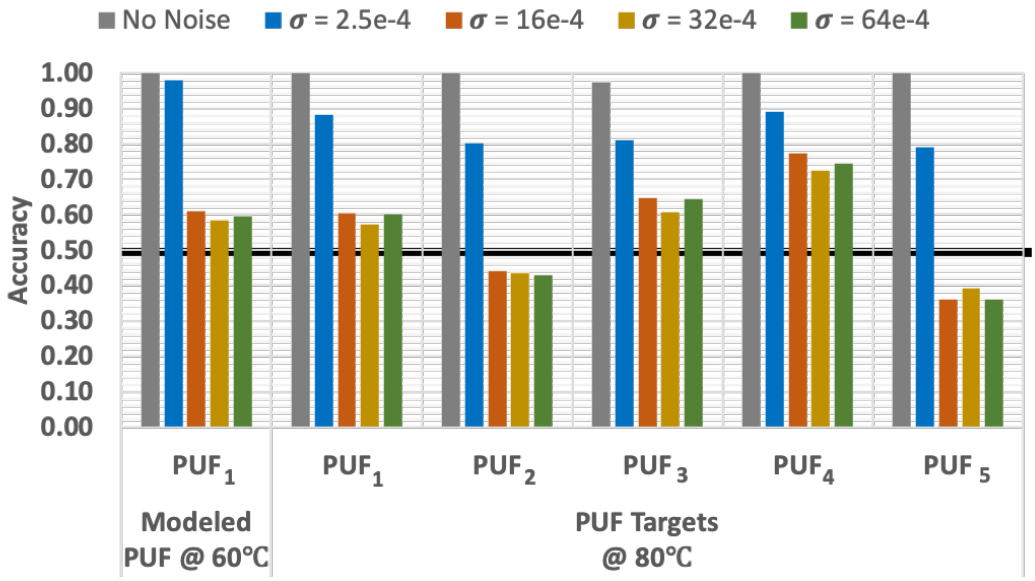


Figure 7.15: The modeling results for the Self-PUF and Cross-PUF attacks targeting the Flip-Flop for the DRILL protected PUFs operating at 80°C. The model was built based on PUF-1 operating at 60°C.

Second, Figure 7.16 shows the results of the Cross-PUF attack with aging misalignment between the reference PUF and the target PUF. In these results a model was made using the power consumption of an unaged PUF-1 and used to attack PUF-2 over 2 years of age every two months. Throughout the results it can be seen that the accuracy of attack at each age has little variation from one another. The absence of variation shows that similar to the previous result of aging misalignments aging has little effect on the outcome of the attack. Moreover there is little effect on the accuracy of the attack when compared to Figure 7.14.

The final situation that could be seen by an attacker is a combined misalignment in the temperature and age of the reference and target PUFs. It is very likely

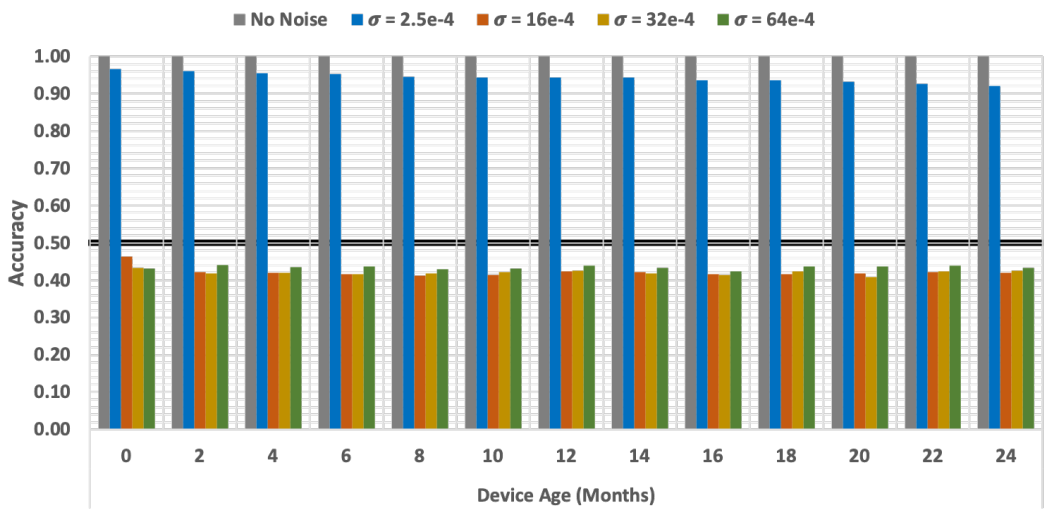


Figure 7.16: The modeling accuracy for the Cross-PUF attacks targeting the Flip-Flop of a DRILL protected PUF at 80°C in presence of aging misalignments. The model was built based on the power traces of the un-aged PUF-1 and used to attack PUF-2 operating at the same temperature.

that the attacker would see both of these when performing the attack. The results of the combined misalignment temperature and aging are shown in Figure 7.17. The accuracy of the attack is shown for an unaged PUF-1 reference PUF operating at 60°C and PUF-2 aged for two years (at two months increments) operating at 80°C. These results show that the accuracy drops for added noise of *sigma* =2.5e-4 from approximately 90%, for the results when the temperature is the same, to around 80% when there are differences in age and temperature. The accuracy of the attack at greater levels of noise is relatively unchanged, again this lack of change shows that the algorithm is guessing the response bit in these attacks.

The contribution of these investigations show that, much like the ability to perform the Cross-PUF attack (as seen in results seen in Chapter 6), a designer can perform protection from the attack using DRILL with the knowledge that said protections will be minimally effected by environmental effects. Moreover, if there is

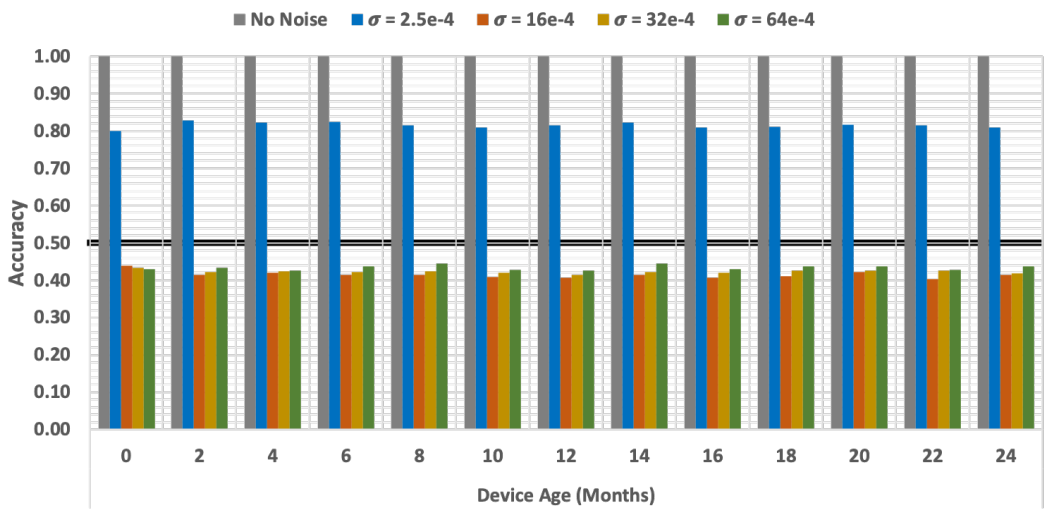


Figure 7.17: The modeling accuracy for the Cross-PUF attacks targeting the Flip-Flop of a DRILL protected PUF in presence of both temperature and aging misalignments. The model was built based on the power traces of the un-aged PUF-1 in 60°C, and tested based on the traces extracted from the aged PUF-2 operating at 80°C.

no control over the environmental conditions, the adversaries ability to attack may actually be hampered further.

7.3.2 Cross-PUF Countermeasure Discussions

The results in this section show that DRILL is an effective countermeasure to Cross-PUF attacks. There are some interesting characterizations of the presented results that should be discussed. First, note that in Figure 7.14 there is a large variability in the accuracy of the attack across the multiple PUFs at noise levels where $\sigma \geq$16e-4. This variability is due to the bias on the response output (known as uniformity) created by the technological dispersion of the PUF. Technological dispersion is not something that, in this instance, should be compensated for since it is due to this variability that the PUF generates its random values. The uniformity

of the PUFs are shown in Table 7.4.

Table 7.4: Uniformity of the PUF instances used in the assessment of the DRILL countermeasure.

	Percent of Response = 0	Percent of Response = 1
PUF-1	36.18%	63.82%
PUF-2	60.99%	39.01%
PUF-3	31.25%	68.75%
PUF-4	14.44%	85.56%
PUF-5	70.10%	29.90%

As the table shows, the uniformity of the PUFs are varied about the desired 50% mark. This fact is important because it shows that the design of the PUF is not inherently flawed rather the instances of the PUF have biases, towards a 0 or 1, related to their process variations. These biases are reflected in the results of the Cross-PUF attack of Figure 7.14 in the variability of the accuracy. In the results it can be seen that where the bias is similar the accuracy is higher i.e., since PUF-1 is bias towards response of 1 the attack accuracy of PUF-3 and PUF-4 are also higher due to their bias towards a response of 1. There are three important facts that are highlighted from this:

1. the bias is varied about 50% for all PUFs showing there is not a problem with the PUF design/architecture but that this is due to process variations of the PUF;

2. the accuracy of the attack is varied about 50%, since the biases of an unseen PUF are unknown to an attacker therefore the bias provide little information on whether the attack will be more or less successful for a new PUF instance;

3. since these results indicate that the attack is mitigated by the use of the

DRILL countermeasure, and there exist uniformity bias in the individual PUF instances, DRILL is effective even when these biases exist.

The results show that the DRILL countermeasure provides protection despite variations in temperature and misalignments in age. Again it is important to note that the environmental effects are not a protection from an attack however this characterization is important knowledge for a designer.

7.4 Contributions

In this section countermeasures for power-based modeling attacks were discussed. Each countermeasure was characterized by the goal it was trying to achieve: either reducing the SNR to where the response could not be predicted from the traces and/or attempting to confuse or poison the modeling algorithm. Combinations of the countermeasure served to combine the advantages of the separate countermeasures. Tradeoffs between the countermeasures were also reviewed to provide guidance in their usage. Additionally, by using DRILL, a PUF's vulnerability to Cross-PUF attacks can be mitigated even in the presence of temperature variations and differing ages of the reference and target devices.

Chapter 8: Physical Implementation Results

The results presented thus far have pertained to attacks which occurred on simulated PUF instances. To add an element of realism to these results gaussian noise was added to the collected power traces from the simulations. This noise approximated the noise that is observed by the IC in which the PUF is implemented including the components which make up the PUF itself. These results showed that there exists a pervasive vulnerability in modeling the PUF behavior although it is supposed to be unclonable. Although these results are compelling, their real silicon implementation requires investigation. The transfer of these results to a physical platform and subsequent successful attack provides conclusive evidence that these attacks are feasible on real physical instances of PUF circuitry. To perform these attacks, we implemented multiple arbiter-PUFs on an FPGA.

This chapter shows the results of attacking real silicon instances of the arbiter-PUF within an FPGA for both Self-PUF and Cross-PUF attacks. Attacks on Multi-bit Parallel PUFs implemented in an FPGA are also investigated herein. These attacks are performed on unprotected and DRILL protected PUFs (see Chapter 7 Section 7.1.5 for an overview of DRILL). A revised methodology for performing these attacks is required for success. The attack does not change very much, however due

to some of the eccentricities of data collection trace alignment is required for a successful Cross-PUF attack. Note that this methodology is similar to the shifting of traces required previously when attacking the latch.

8.1 Attack Methodology for FPGA Implemented PUFs

To attack the traces collected from the FPGA implemented PUFs the methodology is somewhat similar to that performed in simulation, however there are some additional steps that are used to successfully execute the attacks. The traces are collected during the PUF's operation, that is the period of time in which the rising edge propagates through the PUF, the response is arbitrated, and then the response is registered in the Flip-Flop. The principle focus of the attacks, much like before, is during the period of time in which the Flip-Flop is active, therefore the only portion of the traces which are associated with the phenomenon are used. When performing the Cross-PUF attacks, the traces are assessed if they need to be shifted to align the principle leakages created by the PUF's Flip-Flop. At this point the traces can be attacked in the traditional manner in which they were attacked for the simulated traces. The process just described is shown in the block diagram of Figure 8.1.

The attacker for this threat operates much in the same way as was mentioned in Section 6.1. This threat model, as can be recalled, is that the attacker is capable of collecting the power traces of a device with the PUF implemented within the system. This attacker can create a model of the PUF within their possession and then use that model to attack a previously unseen PUF for unobserved challenges.

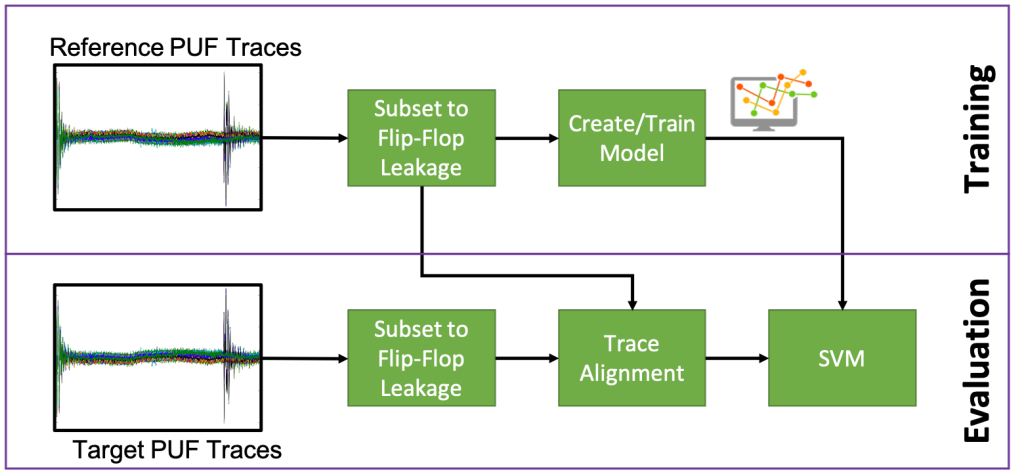


Figure 8.1: Block diagram of the attack methodology for the PUF instances in FPGAs.

When performing the aforementioned attack the inputs to training and the inputs for testing and evaluation are randomized to ensure that there are not artifacts from the order in which the challenges were given to the PUFs. Likewise none of the challenges used in training were used in performing the test and evaluation (i.e., attacking) the PUF. While not necessarily a component of the methodology it is important to note that the FPGA board used for the implementation has a built in amplification circuit. This circuit performs some analog filtering of voltage while also providing 20dB of gain to the recorded signal [4].

8.2 FPGA Implementation Overview

The utilized Sakura-G FGPA board contains a Xilinx Spartan-6 FPGA. This device is composed of many slices of FPGA logic which is shown in Figure 8.2. The structure of the slice consists of four identical Look-Up Table (LUT) structures which

implement the programmable logic for the device[1]. Each LUT structure shares a common clock (CLK), clear (CE), and set-reset (SR) signal.

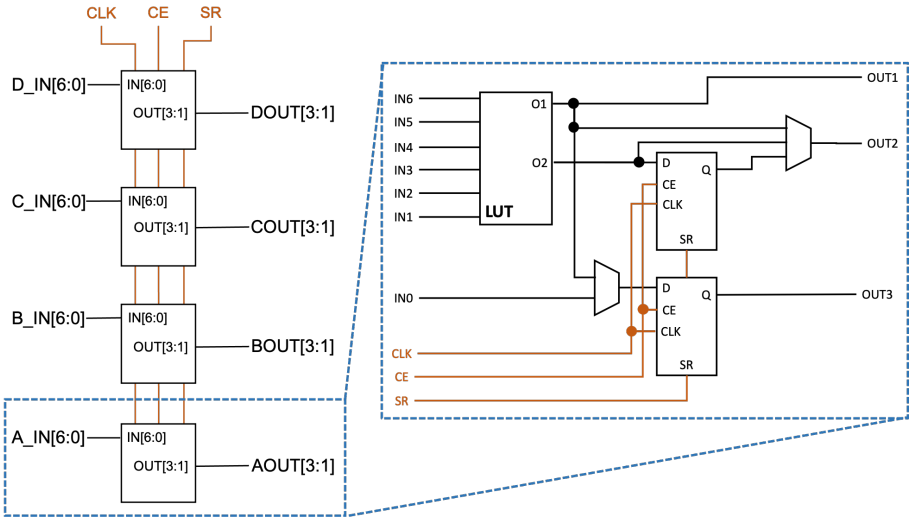


Figure 8.2: Structure of Xilinx Virtex-6 Slice. The LUT logic structures use shared CLK, CE, and SR inputs [5].

The structure of the arbiter-PUF must be carefully placed into these structures to ensure that all of the multiplexers (switches) and the interconnections within the arbiter-PUF chain are identical. The placement for the switch within the slice is shown in Fig. 8.3.

The implemented PUF was developed with consideration to balancing its uniformity [59] and its uniqueness from other PUF instances [58]. The arbiter-PUF FPGA implementation is further documented in Appendix A.4. Also note that the metrics of the implemented PUF are relayed in this appendix as well.

[1]There are many other structures within an FPGA. The slice and LUT structures are shown here as they are the primary means of implementing the PUF within the FPGA.

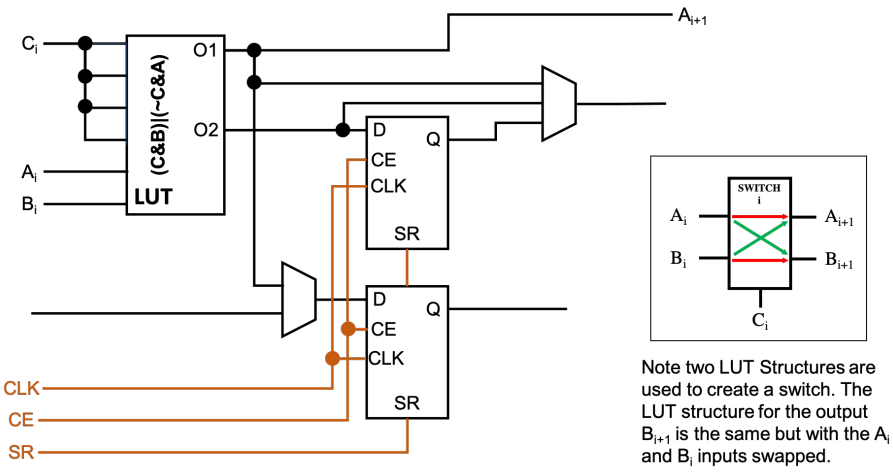
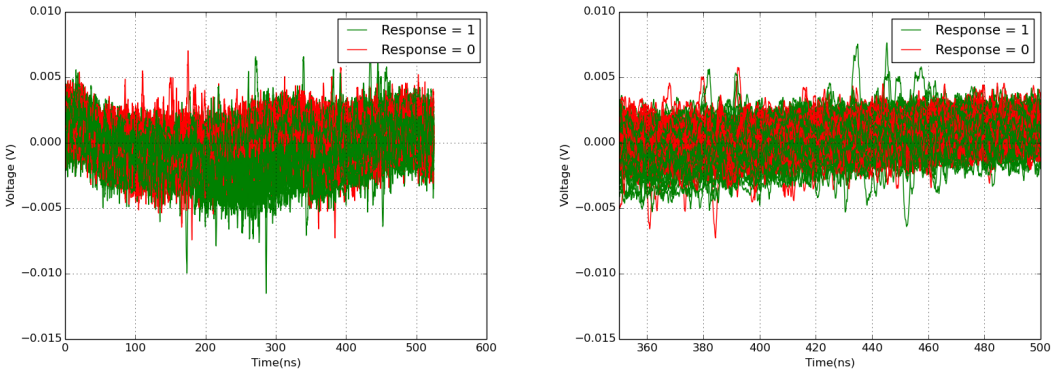


Figure 8.3: Placement of a switch structure within the LUT components of a slice.

8.3 Unprotected PUF

In this section the result of attacking the unprotected instances of the FPGA implemented PUF are detailed. For these results five identical PUFs were implemented within the FPGA albeit in different locations resulting in different physical characteristics to produce five unique PUF instances. The first observations to be made on the efficacy of the attack is the indistinctive nature of the PUF output, which can be seen in the traces presented in Figure 8.4. The traces presented here are unlike those of the simulation in the fact that the output of the traces cannot be discerned from mere observation the traces for a resulting 0 or 1. Nevertheless, by following the attack process described in Section 8.1, the five FPGA implemented PUFs were attacked. The results of these attacks are shown in Table 8.1.

It can be seen in the table that the success of the Self-PUF attack ranges from 92.62% to 94.72%, showing that the attack is successful when attacking a PUF with a model created from traces collected from itself. Furthermore the attack accuracy



(a) Full length of the recorded power traces.

(b) Power traces focused on the power consumption of the Flip-Flop.

Figure 8.4: Traces for the unprotected PUF instance. There are 200 total traces shown for PUF-1.

Table 8.1: Accuracy of the Self-PUF (**bold**) and Cross-PUF attacks for the unprotected PUFs by using SVM with the shift-based alignment. Training was performed with 1000 traces.

Modeled PUF	Attacked PUF				
	PUF-0	PUF-1	PUF-2	PUF-3	PUF-4
PUF-0	**0.9387**	0.9326	0.9115	0.9260	0.9235
PUF-1	0.9180	**0.9472**	0.9048	0.9360	0.9250
PUF-2	0.9038	0.9257	**0.9262**	0.9158	0.8938
PUF-3	0.9048	0.9343	0.9121	**0.9309**	0.9059
PUF-4	0.9393	0.9325	0.9069	0.9247	**0.9433**

of the Cross-PUF attack ranges from 89.38% to 93.93%. These accuracies also show that the Cross-PUF attack is just as successful as that of the Self-PUF attack, undermining the uncloneabilty concept associated with the properties of a PUF as one PUF should not reveal information about another.

8.3.1 Discussions on attacking Unprotected FPGA PUFs

First, as previously noted, the attack is successful on physically implemented PUFs. This success is concerning as it undermines the security presented by the PUF. The results previously seen in simulation are confirmed by those seen in the real silicon implementation within the FPGA.

To properly compare to the previous results we observe and compare the SNR of the traces from each of the five implemented PUF instances. Recall that the SNR is the ratio of the signal over the noise present in the collected data and the equation for its calculation is shown in Equation 4.3. The maximum SNR for each of the PUFs is shown in Table 8.2.

Table 8.2: Maximum SNR of the five unprotected PUFs implemented within the FPGA

PUF Inst.	PUF-0	PUF-1	PUF-2	PUF-3	PUF-4	Avg. All
Max SNR	0.282	0.228	0.161	0.195	0.072	0.188

Recollect that the SNR experienced in the simulated traces with added noise (from Table 6.2) ranged from 12.32 to 0.023 for a σ =2.5e-4 to σ =64e-4. More specifically the average SNR appears to indicate that the appropriate added noise figure is between σ =16e-4 (SNR=0.309) and σ =32e-4 (SNR=0.080). This SNR range indicates that our simulations were in fact within the noise range of a real device.

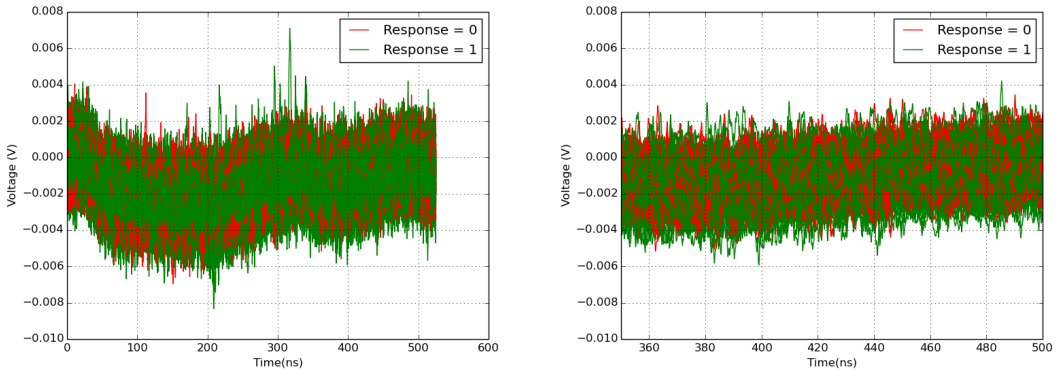
Again, these results indicate that the attack is successful for both the FPGA implementation confirming what is seen in the simulation. Comparing the FPGA implementation results seen in Table 8.1 with those of the simulation (seen in Fig-

ure 6.5), there is a slight decrease in expected accuracy for the SNR range indicated. The simulation results are near 100% for these SNR levels whereas the FPGA results are in the low 90% range. This discrepancy is likely due to differences between the simulation being perfect such as consistent sampling of the devices at a highly precise moments in time and the ability to sample the device at a higher frequency.

8.4 Protected PUF

Also implemented within an FPGA were protected instances of the PUF. The primary means of protection implemented were through the DRILL protection mechanism. The focus on DRILL is due to its shown effectiveness in simulation which is desirous of confirmation of these results in real-silicon implementations, such as the FPGA implementations here.

Similar to the unprotected results there were five FPGA implementations of the Arbiter-PUF protected with the DRILL countermeasure studied. The DRILL countermeasure is described in Section 7.1.5. This protection, as described before, utilizes Dual-Rail Logic on the arbiter primary output and its inverse as well as Random Initialization Logic on both of the implemented Flip-Flops to lower the SNR of the PUF response output. The traces resulting from one of the implementations are shown in Figure 8.5. As seen in the figure the response is not readily viewed from observing the trace alone which was similar to the observation made about the unprotected traces, therefore adding the protection did not incur any anomalies that would reveal the result.



(a) Full length of the recorded power traces.

(b) Power traces focused on the power consumption of the Flip-Flop.

Figure 8.5: Traces for the DRILL protected PUF instance. There are 200 total traces shown for PUF-1.

We followed the methodology discussed in Section 8.1 to launch Self-PUF attacks (using their own power traces for modeling) and Cross-PUF attacks (using the model of one PUF to attack another) on these PUFs. The results of these attacks are shown in Table 8.3.

Table 8.3: Accuracy of the Self-PUF (**bold**) and Cross-PUF attacks for the PUFs protected by DRILL. Training was performed with 1000 traces.

Modeled PUF	Attacked PUF				
	PUF-0	PUF-1	PUF-2	PUF-3	PUF-4
PUF-0	**0.6394**	0.4868	0.5819	0.5199	0.5386
PUF-1	0.5015	**0.6074**	0.4803	0.4525	0.5238
PUF-2	0.5739	0.4761	**0.7368**	0.5002	0.5002
PUF-3	0.5001	0.4528	0.4751	**0.5586**	0.4743
PUF-4	0.5444	0.5397	0.4714	0.5038	**0.6002**

Ideally if DRILL provides protection to the PUF the results of attacking should be no better than a best chance guess, that is 50% when choosing between a '1' or '0' for the response of the PUF. For the Self-PUF attacks the accuracy is slightly higher than would be expected for a protected PUF with the average being 62.84%.

However, this attack percentage would not yield enough accuracy from the attacked PUF to provide fruitful information to the attacker. Overall, the accuracy drops 30% from about 93% for attacking an unprotected PUF instance thus providing considerable vulnerability mitigation to the circuitry. The countermeasure did much better when launching Cross-PUF attacks where the average attack accuracy is 50.49% which is expected from a properly protected PUF.

8.4.1 Discussions on attacking Protected FPGA PUFs

The first takeaway of the results from DRILL protected PUFs is that DRILL provides protection for both the Self-PUF and Cross-PUF attacks. These results verify what is seen within the simulation results presented in Section 7.3, which is the fact that DRILL provides protection to the PUF circuitry in realistic scenarios.

The maximum SNR of the protected PUF is shown in Table 8.4. It can be seen that the DRILL countermeasure achieves its objective of lowering the SNR of the PUF's power traces. Furthermore it can be seen that the PUFs with a higher attack accuracy have a higher SNR, i.e., PUF-2 has a Self-PUF accuracy of 73.68% and an SNR of 0.055, both of which are considerably higher than the other PUFs. There are two points that can be made from this observations; first the SNR is a reasonable estimate of the attackability of the PUFs, and second that a high SNR of a single PUF does not compromise other PUFs.

Table 8.4: Maximum SNR of the five DRILL Protected PUFs implemented within the FPGA

PUF Inst.	PUF-0	PUF-1	PUF-2	PUF-3	PUF-4	Avg. All
Max SNR	0.0377	0.0034	0.0549	0.0070	0.0149	0.02358

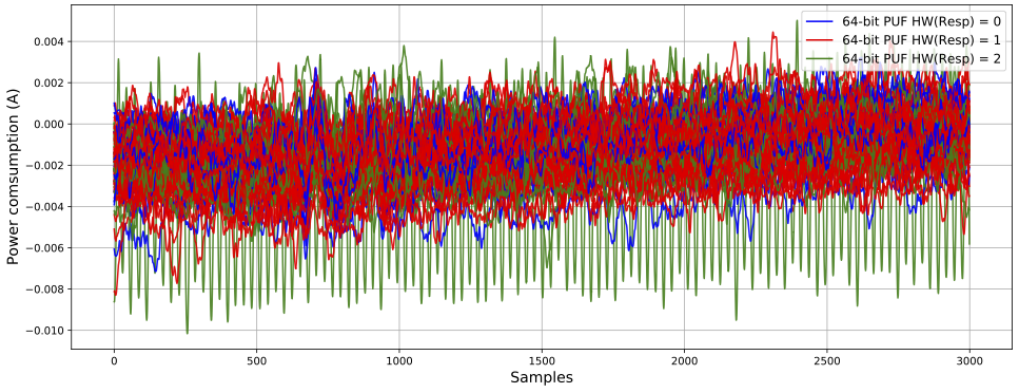
8.5 Multi-bit Parallel PUFs

In Chapter 5 attacks on Multi-bit Parallel PUFs were investigated and discussed from implementations based in SPICE simulations. Correspondingly the countermeasures for power based modeling attacks on parallel PUFs were shared in Chapter 7 Section 7.2. To complete the investigation of these sorts of PUFs they were also implemented within the fabric of the FPGA. Two instances of unprotected and two instances of DRILL protected Multi-bit Parallel PUFs were implemented for investigation. Note that the implementation is the same as the single bit implementation however two simultaneous PUFs are triggered at the same time and produce their responses within the single clock cycle. Therefore, the power recorded during this time corresponds to the operation of these two PUFs with the naturally occurring systemic noise.

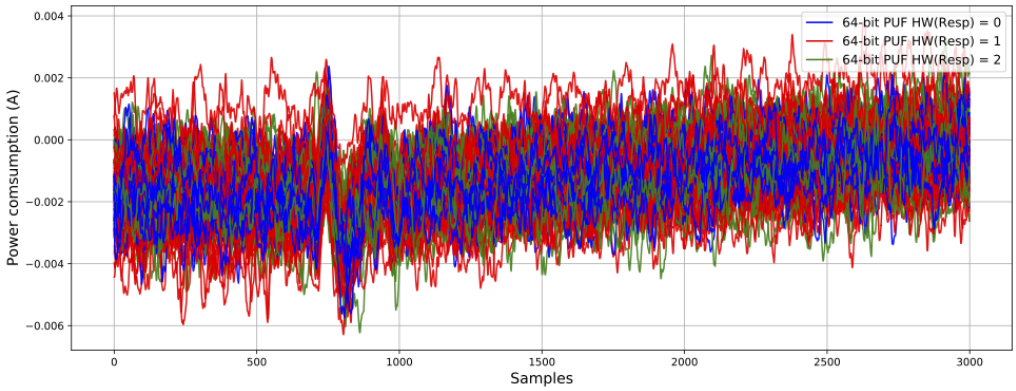
8.5.1 Results of attacking Multi-bit Parallel PUFs within FPGAs

Much like before it is important to observe the power traces of the investigated PUF. The power traces for the unprotected PUF are shown in Figure 8.6(a). This figure focuses on the period of time in which the response Flip-Flops are active and the responses cannot be distinguished from pure observation of the power traces. In

Chapter 5 we also noted that depending on the level of noise the Hamming Weight of the response may be readily observed in the traces. However it is not apparent in the observed traces from the FPGA. The traces shown in Figure 8.6(b) are for the DRILL protected PUF instance and display a similar phenomenon to those of the unprotected parallel PUF. The greatest difference is in the face that at around 750 samples there is a clear perturbation amongst all responses for the PUF. This event does not reveal the response as all of the traces are the same. It is likely due to the dual Flip-Flops operating simultaneously increasing the leakage during the registration of response.



(a) Power traces of the unprotected PUF-0 instance (≈100 traces).



(b) Power traces of the DRILL protected PUF-0 instance (≈100 traces)

Figure 8.6: Power traces for the PUF-0 Multi-bit Parallel PUF instance zoomed to highlight the Flip-Flop operation.

The results of Self-PUF attacks on Multi-bit Parallel PUFs are shown in Figure 8.7. In these attacks 5000 traces were used for modeling the PUF and the model was validated on 7000 traces to get the accuracies displayed here. The first thing that can be observed is the high rate of attack accuracy on the unprotected PUF at 83.5% and 91.8% for PUF-0 and PUF-1 respectively. This result shows that the Multi-bit Parallel PUF is indeed vulnerable to the same sorts of power based modeling attacks as its unparallel counterpart. The second thing to note is that there is a large decrease in accuracy for the PUFs protected with the DRILL circuitry. Here the accuracy for attacks on the protected PUFs drops to 32.7% and 42.22% for Protected PUF-0 and Protected PUF-1 respectively. This sharp decrease in the accuracy of the attack shows that the DRILL countermeasure is effective in protecting against power-based modeling attacks on parallel PUFs implemented within FPGAs.

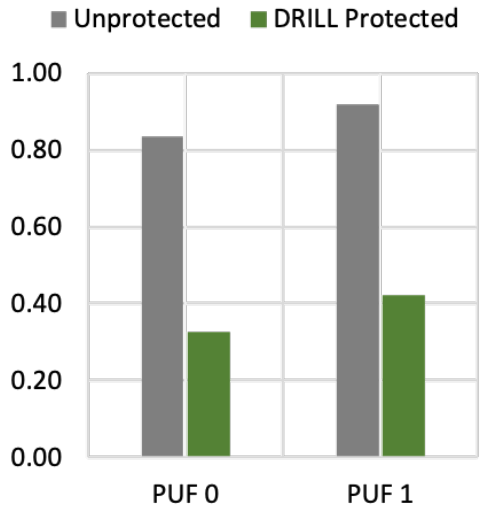


Figure 8.7: Results of attacking unprotected and DRILL protected Multi-bit Parallel PUFs. 5000 traces were used for training.

The accuracy of the attack for each individual two-bit response is shown in Table 8.5. The individual responses for the unprotected PUFs range from 62.58% to 96.72% for PUF-0 and for PUF-1 they range from 88.25% to 94.64%. These results show that it is easier to infer the responses of 00 and 11 than 01 and 10 which is somewhat expected, as these values will produce the highest and lowest leakages. This observation is not true for the protected PUFs where the Dual Rail Logic portion of DRILL levels out the leakages with the inverted value Flip-Flop. From Table 8.5 it can be seen that the protection holds for the multi-bit PUF instance as there is not any response representation that is predicted abnormally more than another. The accuracy from each response ranges from 29% to 36.41% for the DRILL Protected PUF-0 and from 37.91% to 48.26% for PUF-1.

Table 8.5: Multi-bit Parallel PUF predictions of individual responses from implemented PUFs for when 5000 traces were used for training.

PUF Inst.	Response	Percent Correct	
		Unprotected	Protected
PUF-0	00	0.9439	0.3349
	01	0.7039	0.3641
	10	0.6258	0.3255
	11	0.9672	0.29
PUF-1	00	0.9308	0.4826
	01	0.8825	0.3791
	10	0.9021	0.4201
	11	0.9464	0.4097

The takeaway from these results is that the parallel PUF instances are attackable in real-silicon however DRILL is an effective countermeasure for these sorts of attacks.

8.5.2 Discussions on attacking Parallel PUFs

Power based modeling attacks on unprotected parallel PUFs were shown to be effective. This high rate of success occurs despite the fact that there was no power trace averaging which was used in the simulated PUF results in Section 5.3. This difference shows that the real implementation of the PUF is more vulnerable than expected and is likely due to the eccentricities of real circuitry. Here each of the parallel PUFs implemented together will have variations which if not paired or matched will reveal which bit is producing certain artifacts of the power consumption observed by the device. The attacker can take advantage of these variations to determine what the response bits of the parallel PUF are, and indeed does by performing the attack.

Table 8.6 shows the SNR of the parallel PUFs. In this table it can be seen that the average SNR of the unprotected PUF is 0.147 and the average of the DRILL protected PUF is 0.026. For each bit of each PUF the SNR was lower than its unprotected counterpart showing that the DRILL protection reduced the SNR as was expected. In simulation the SNR of the traces for a parallel PUF were shown in Table 7.3. From this table the unprotected PUF the SNR of the PUF was 0.233 when the noise was σ =16e-4 and 0.062 when σ =32e-4. Therefore, the level of noise in the simulation was similar to that shown in the real implementation. Also note that this similarity existed for the FPGA implementation of the single bit PUF previously shared.

From the results on the Parallel PUF it can be seen that it is more difficult

Table 8.6: Maximum SNR of Multi-bit Parallel PUFs implemented within the FPGA

PUF Inst.		Unprotected	DRILL Protected
PUF-0	Resp. Bit 0	0.175	0.021
	Resp. Bit 1	0.136	0.011
PUF-1	Resp. Bit 0	0.138	0.069
	Resp. Bit 1	0.138	0.003
Average		0.147	0.026

to attack however it is still vulnerable to power-based modeling attacks in its unprotected state. Therefore, to thwart an attacker some countermeasure for these attacks should be considered by the designer. As shown here, DRILL certainly provides some mitigation to the overall attack.

8.6 Contributions

The primary contribution of the FPGA implemented PUF results is the confirmation that the simulation correctly identified a critical vulnerability in PUF circuitry. The FPGA implementation results confirm that PUFs can be attacked though both Self-PUF attacks as well as by Cross-PUF attacks utilizing the power consumption of the device. The FPGA implementation of the Multi-bit Parallel PUF was also confirmed to be vulnerable in its real-silicon implementation. The SNR is confirmed as an indicator that the a device may be vulnerable to power consumption attacks. Finally, it was confirmed that the DRILL countermeasure lowers the SNR of the PUF's power consumption and ultimately protects the device from Self-PUF and Cross-PUF attacks as well as attacks on Multi-bit Parallel PUFs.

Chapter 9: Summary and Conclusions

This chapter is a review of the previous chapters, reminder of the contributions of this work, and a summation of the conclusions of the presented investigations.

9.1 Recapitulation

This research investigated the susceptibility of PUFs to power based side-channel attacks and determine the extents of its vulnerability. Initial investigations were performed on Arbiter PUFs to compare with traditional attacks based on CRP modeling. These results showed that power based side-channel modeling was more effective and required less information to perform the modeling. These investigations were expanded to supposed model resistant PUFs (a challenge obfuscated PUF and the VTC PUF) showing that although they protect against CRPs they do not prevent power based modeling attacks.

The work presented here focuses on the registration of the response bit in a Flip-Flop, a critical component required for using the PUF's response. Performing the investigations on a PUF's power consumption exposed a new vulnerability wherein the attacker can attack one PUF with the information of another by modeling the power consumption related to the operation of the Flip-Flop responsible

for registering the PUF's response. This threat model is an innovative way to exploit PUFs, which extends the list of threats regarding "physical unclonability" described in Clause 5.5.7 of ISO/IEC 20897-1:2020 [42]. The simulation results were very compelling showing a high level of accuracy in all modes of the attacks performed. These attacks were furthermore investigated by varying in age as well as temperature variation and the results showed that there is still a high level of success despite this misalignments. This finding is a crucial point in understanding as an attacker does not need to worry about such factors to perform a successful attack. Also investigated in simulation were Multi-bit Parallel PUFs. The characteristics of their responses were analyzed and discussed along with showing that these types of PUFs are also vulnerable to these power based modeling attacks as well.

Refocusing the effort from attacking to defending, several countermeasures were investigated and discussed showing varying degrees of protection. These countermeasures were based either on reducing the SNR seen in the power traces or confusing/poisoning the model with erroneous values. A proposed countermeasure was more heavily investigated, which combined the Dual Rail Logic and Random Initialization Logic, coined DRILL, to significantly reduce the SNR and mitigate the attack on the PUF. The lightweight countermeasure DRILL can be added to all of the PUF instances investigated including the Multi-bit parallel PUFs. In simulation DRILL sufficiently accomplishes the aforementioned goal of reducing the SNR and mitigating the attack in realistic scenarios i.e., situations in which there is systemic noise in the system like those seen in real devices. Furthermore DRILL performs this goal despite the temperature and age variation of the devices. In the evaluation of

Multi-bit Parallel PUF the effectiveness of Random Arbiter Swapping and Random Response Masking were also useful in protecting the PUF from attack. Utilization of the DRILL and RAS countermeasures together were particularly effective in protecting the PUFs. This combination showed that the combination of poisoning the modeling algorithm and SNR reduction was effective.

To support the simulation results and provide verification of their efficacy the arbiter-PUF was implemented in hardware via an FPGA. Multiple PUF instances were implemented on a Sakura board and the power consumption of the device was recorded during the PUF's operation. The results confirm that these power based modeling attacks do in fact occur on real devices. The results show that SNR is also an indicator of how successful an attack of this nature will be. Investigation of the DRILL countermeasure was also performed in the FPGA. These results confirmed that the countermeasure decreased the SNR of the device and in doing so thwarted the attack on the implemented PUF.

9.2 Review of major Contributions

These are the contributions of this work, reiterated from the introduction to this work.

- Show the vulnerability of PUFs (in particular the arbiter-PUF and its family) to power side-channel based modeling attacks;
- Compare the effectiveness of the power side-channel based modeling attacks and the CRP-based modeling attacks;

- Show the effect of environmental conditions (e.g., temperature) as well as aging on the power side-channel based modeling attacks;
- Show the ineffectiveness of countermeasures tailored against CRP-based modeling in preventing power-based modeling attacks and develop appropriate countermeasures against such attacks;
- Develop Cross-PUF attacks for the first time and show their efficiency in modeling the PUFs behavior;
- Investigate the effect of temperature and aging misalignments in the efficiency of the Cross-PUF attacks for both unprotected and protected variants;
- Launch successful power side-channel based modeling attacks on multi-bit parallel PUFs;
- Launch power-based attacks on the unprotected variants of single-PUFs, Cross-PUFs, and parallel PUFs realized in FPGA as well as their protected counterparts equipped with the proposed DRILL countermeasure and show the effectiveness of the proposed countermeasure.

9.3 Directions of Future Work

Protecting and securing hardware is an unending challenge much as it is for the investigation shared here. There are some suggested expansions of the research presented in this work. The next exploration which would be investigations of the Arbiter-PUF presented here within ASICs. An ASIC implementation would be ideal

as the PUF would be implemented with the actual transistors that are expected to be used for each component instead of the lookup tables that are used within FPGAs. Realistically this investigation would provide a comprehensive look at these sorts of attack as designers really only have these two avenues, FPGAs and ASICs, in which to implement the PUF.

In this work only power based modeling attacks were investigated however Electromagnetic (EM) attacks could also be of interest. These EM attacks could be used to show if the response storage Flip-Flop can be compromised in a similar fashion and determine if this component poses an even greater vulnerability. Moreover it would be interesting to determine if Cross-PUF attacks can be perpetrated with using the EM signatures of the device.

Tangentially related to this research an in depth comparison between realistic power measurements and those performed in the SPICE simulations would help future research. An in depth comparison will provide more confidence in the simulation results and in particular a comparison of the SNR levels of the device with systemic added noise. If a characterization can be made then simulation results can be used more readily for research.

9.4 Conclusion

This work provided investigations on the resiliency of PUFs to power based modeling attacks. CRP focused countermeasures require further mitigations to prevent these sorts of attacks. Furthermore these sorts of power based modeling attacks

expose a vulnerability previously thought impossible, that is the ability to attack one PUF with information collected from another. This sort of vulnerability should be cognizant to designers and end-users of PUFs. Lightweight countermeasures launch as those proposed in this work should be used to prevent attackers from successfully launching these attacks, especially when investigations suggest that varying temperature and differing device age have little effect on these attacks.

Appendix A: Simulation Environment and Data Collection

The following describes the simulation environment, data processing, and analysis techniques.

A.1 Simulation Environment

The various circuit designs for this work are implemented at the transistor level utilizing an open-source NANGATE library for 45 nm technology [2]. The software used to perform the simulations is Synopsys HSPICE.

Within the simulations it was necessary to create variations in the transistor instantiations that would replicate the manufacturing process variations that would occur in a 45 nm process in commercial use. These variations were performed via Monte-Carlo simulations with gaussian distributions for the following: transistor gate length L : $3\sigma = 10\%$, threshold voltage V_{TH} : $3\sigma = 30\%$, and gate-oxide thickness t_{OX} : $3\sigma = 3\%$. Unless otherwise noted all simulations took place with the operating temperature of the device at $80°C$.

To perform the simulation, HSPICE requires the development of a Circuit Netlist, the actual circuit design for which the simulation is desired. Also required is the target Technology Library as well as the Input and Environmental Parameters

(operating temperature, simulation time, input signals, etc.). These three items feed the HSPICE simulation which runs the circuit for the desired time and provides the requested circuit signal outputs. Running this type of simulation is equivalent to simulating the operation of a fresh device (no-aging). This process is presented in the flowchart shown in Figure A.1 [45].

A.1.1 Aging Simulations

Synopsis HSpice has a built in capability known as MOSFET reliability analysis (MOSRA) which is used to induce the effects of HCI and BTI, the primary aging effects of concern, within the simulation tool. Specifically the MOSRA Level 3 is used to capture aging effects. This type of simulation is an empirical model of the devices and captures the effects of HCI and BTI on the threshold voltage V_{TH} and drain-source current I_{ds} [80].

The aging simulation differs from the normal simulation process in that it requires two steps where in the first step (so-called pre-stress) the aging-induced change of the electrical characteristics (e.g., threshold voltage) of the transistors are extracted and in the second step (i.e., post-stress) these degradation of device characteristics are translated to performance degradation at the circuit level. Aging simulation also requires a few more input parameters (the simulation time-steps, the longevity of the simulation, etc.). In the pre-stress simulation MOSRA assesses and calculates the electrical stress placed on the simulated transistors at each of the requested time intervals. In the post-stress simulation the device is simulated running

at the previously calculated stress level providing information on operation changed due to said stress [45]. Note that the simulation times for aging are discussed when those results are presented.

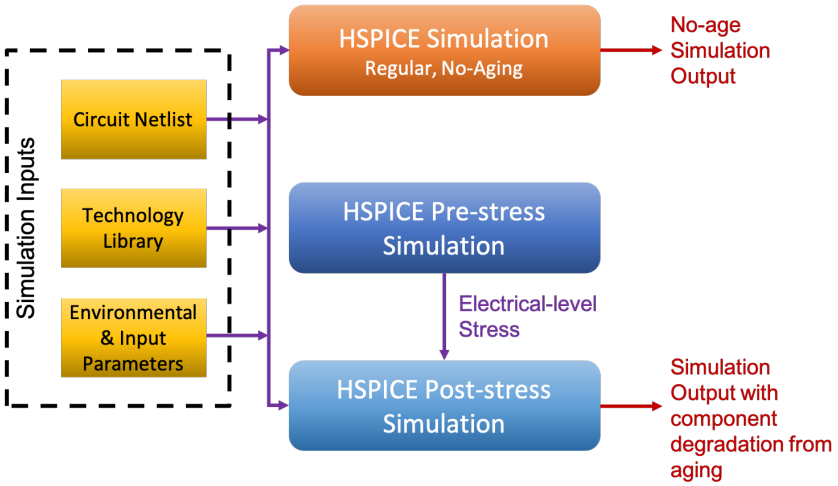


Figure A.1: Simulation Process Block Diagram for performing both normal (no-age) and aging simulations [45].

A.1.2 Simulation Processes

In the simulation of each PUF, the input parameters are carefully controlled to allow the proper operation of the PUF implementation. Each simulation is set up to systematically query the PUF with challenges one after another until the desired number of challenge queries have been simulated. This querying is executed in the form of a loop where the circuit is first provided the challenge, then, after the circuit reaches a stable state, the PUF is fed with a transition (a rising edge in our case) to propagate to the arbiter. The simulation continues until a stable response has reached the end of the circuit and been registered in the output Flip-Flop (if present).

A.2 Data Extraction

The current and voltage of the simulated circuit is provided by Synopsis HSPICE as a continuous set of values, one challenge query after another. It is from these results that the data has to be extracted for each challenge query of the PUF. Figure A.2 shows the sampling window for the data extraction. The samples for each query are extracted starting when the rising edge is given to the PUF. Sampling of the current continues through the entirety of edge's propagation through the PUF and, if present, continues as the response is registered in the output Flip-Flop. When it is clear that the response is valid its voltage value is recorded and converted to its digital equivalent ('0' or '1') in tandem with the sampled power trace. This produces a tuple of data for each query of the PUF:

$\{Challenge, PowerTrace, Response\}$

These tuples make up the data sets. The challenge consists of the string of bits that were provided as to the PUF for each query and the response is the eventual output. The power trace consists of the operating current of PUF during its operation and consists of an array of floating point values. The number of collected samples of the power trace varies based on the size of the PUF due to the longer delay path.

The first challenge query is unused to avoid any anomalies associated with simulation/device start up. When performing the attacks and developing the models only the power trace and the associated response are used.

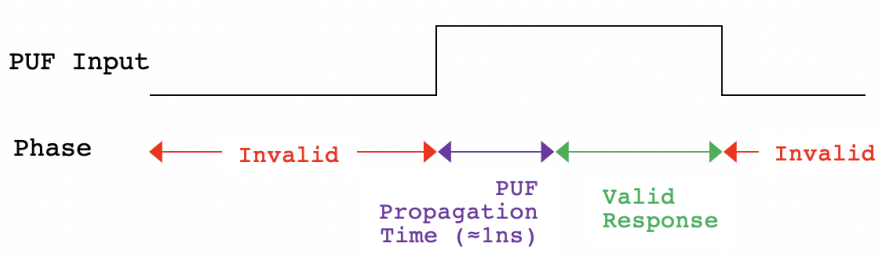


Figure A.2: Power trace and response sampling window.

A.3 Simulated Noise

To provide realism to the simulations and account for the effect that occur in real silicon noise was added to the extracted power traces post simulation. To obtain noisy power traces Y, varying levels of gaussian noise N were added to to the original power trace X

$$Y = X + N \qquad \text{where } N \sim \mathcal{N}(0,\, \sigma^2) \tag{A.1}$$

The standard deviations (σ) for the gaussian noise levels are discussed with the results.

A.4 Support Vector Machine Algorithmic Implementation

The SVMs were implemented in Python using the library scikit-learn. The Radial Basis Function (RBF) kernel is used to manipulate the data. This method is used because it is effective in modeling data that has gaussian properties such as those present in the random variations in manufacturing processes, the exact properties being exploited by PUFs.

If the intention is to perform CRP-based modeling then the ϕ for each challenge is calculated from Equation 3.2. In machine learning there are parameters that need to be controlled to develop the model. These parameters control the learning rate and regularization of the model fitting. For the used RBF kernel the parameters are γ and C. The γ parameter controls the weight that an individual sample holds on the model fitting, if γ is high then the training samples will have a large impact on the fitting of the model. C serves as a regularization parameter; this parameter is used to control the margin in the decision boundary. If C is large then the model is trained to fit the training data more tightly (there is smaller margin around the data points) and if C is lower then there is more margin. Balancing these two parameters aids in ensuring that the model is not over-fitting (only correctly classifies the training data while failing to other data points) or under-fitting (in properly classifying data due to a poor decision boundary) [3].

Figure A.3 and Figure A.4 show examples of the parameter testing graph for the SVM classification of CRPs and power traces respectfully. The testing consisted of running model accuracy tests for varying ranges of γ and C, with the model accuracy being plotted as a function of the two. The chart shows that manipulation of the parameters has a large effect on the model's performance.

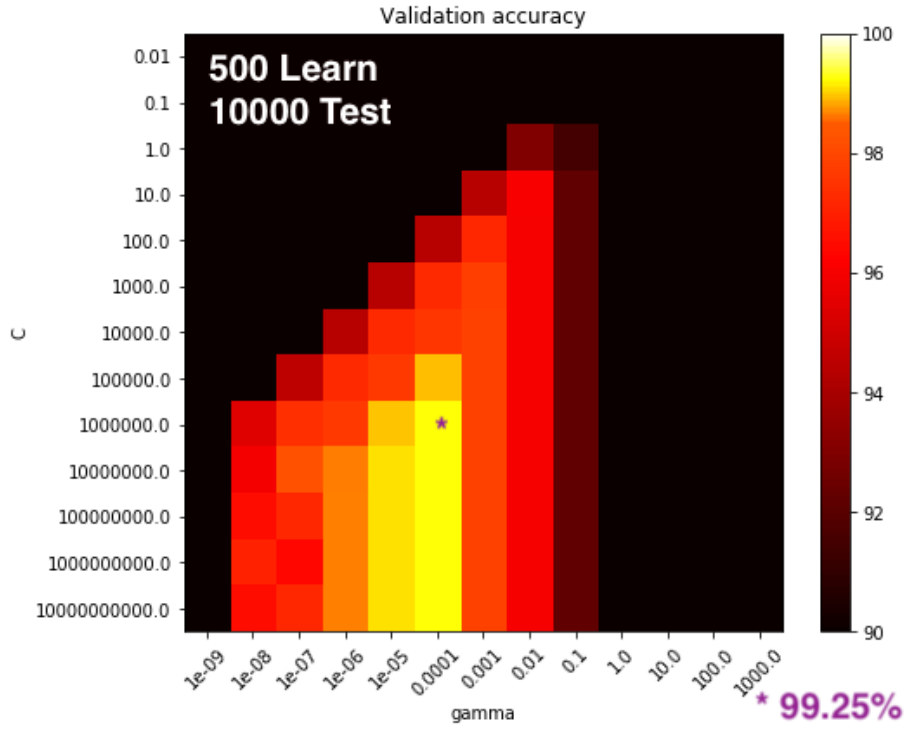


Figure A.3: Parameter testing for the SVM model of CRPs.

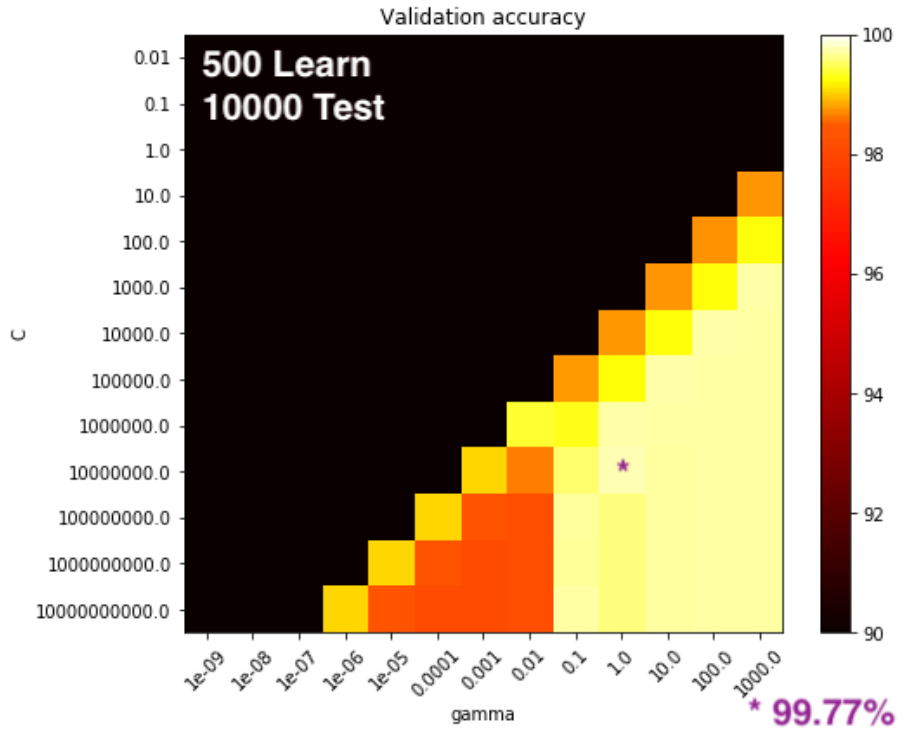


Figure A.4: Parameter testing for the SVM model of Power Traces.

Appendix B: FPGA Implementation of Arbiter-PUF

This appendix details the implementation of the Arbiter-PUF on the Sakura FPGA Board. This is used in producing the results within Chapter 8. The physical investigations for this work are performed within a Field Programmable Gate Array (FPGA). These reconfigurable devices allow for a hardware implementation of a digital circuit without the costly price of creating an Application Specific IC (ASIC). FPGAs include various efficient hardware components such as multiplexers, registers, latches, Lookup Tables (LUTs) amongst others [25].

B.1 Sakura FPGA Board

To implement the PUF the Sakura-G FGPA board was used (shown in Figure B.1) [4]. The Sakura-G FPGA board is programmed using the Xilinx ISE tool [1][1].

This board contains circuitry in conjunction with a Xilinx Spartan 6 FPGA (specifically the Spartan 6 XC6SLX75-2CSG484C) to make collection of its power consumption during operation convenient. The main FPGA on the board is supplied with power from a source which contains measurement points of the operating

[1]Implemented in Xilinx ISE Version 14.7 on Windows 10

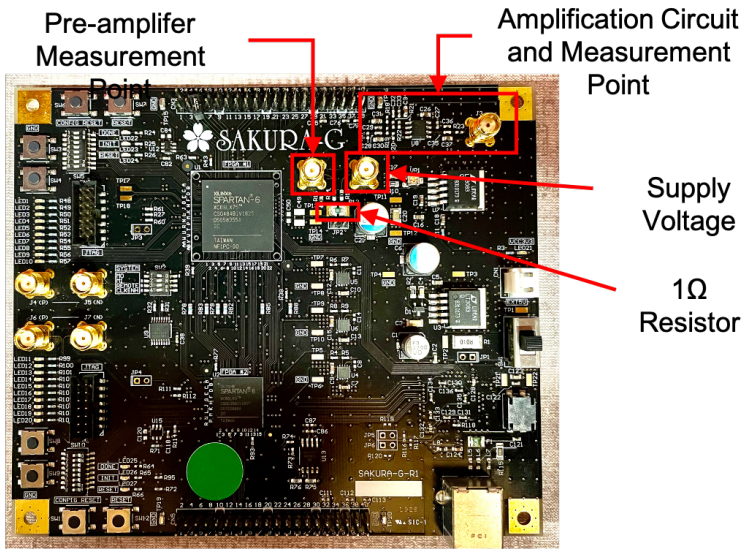


Figure B.1: The Sakura-G FGPA board with pertinent measurement components highlighted.

current through a 1Ω resistor as shown in Figure B.2. As shown in the figure, there is also a built in amplification circuit which allows for the voltage variations to be observed with 20dB of gain [6].

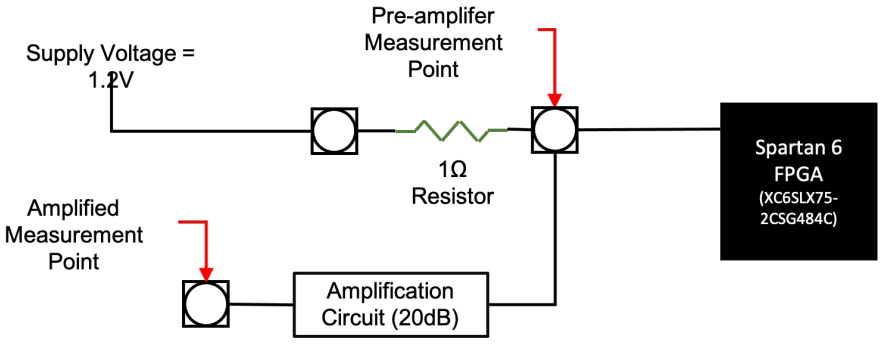


Figure B.2: Measurement setup of the Sakura-G FPGA board.

Note that the power consumption of the device is proportional the voltage due to the fact that the power (P) is defined as:

$$P = V * I \tag{B.1}$$

where I is current and V is voltage. Therefore the change in voltage seen over the resistor on the power supply line is proportional to the power consumption of the device.

B.2 FPGA Implementation

This section details the FPGA implementation of the arbiter-PUF. The implementation consists of a UART module running at 115200 Baud for the challenge input and response reporting to an external computer. The data transfers from the UART are collected by a state-machine which provides the transferred challenge to the PUF instance. After the challenge is supplied to the PUF the state-machine provides the input edge to query the PUF. The PUF executes in a single clock cycle and provides its response to the state-machine which then sends said response to the external computer. To allow the PUF to execute in a single clock cycle a PLL is used to produce a 3.1875MHz clock from the 48MHz clock available to the FPGA. All of the systems run off of this 3.1875MHz clock. The system diagram for the described implementation is shown in Figure B.3.

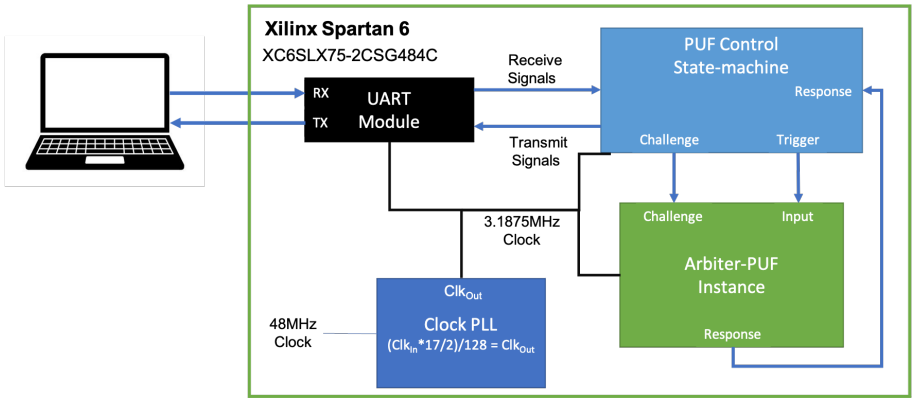


Figure B.3: System diagram for the arbiter-PUF implementation.

B.2.1 Arbiter-PUF FPGA Structure

The standard implementation of the arbiter-PUF is shown in Figure 4.5. This implementation has to be adjusted for successful operation within an FPGA. The adjusted arbiter-PUF is shown in Figure B.4.

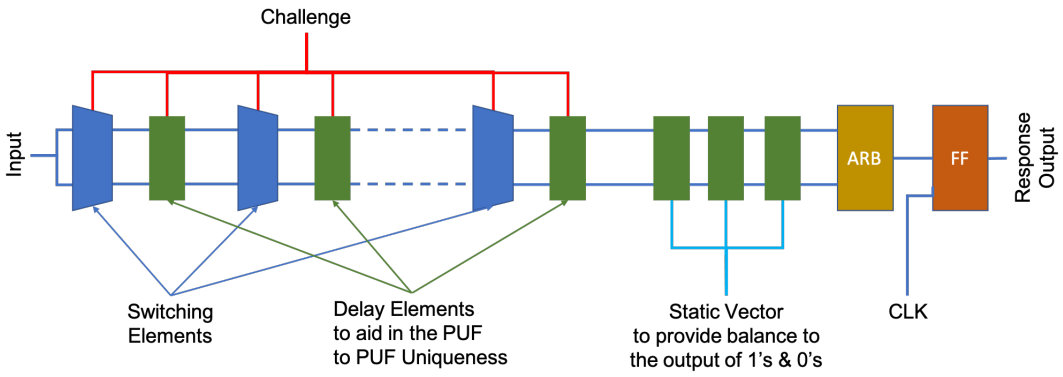


Figure B.4: Arbiter-PUF implementation for operation within an FPGA.

There are two main adjustments made to the implementation for successful operation within the FPGA. First there are added delay elements between each stage. These delay elements do not swap the top and bottom paths rather they add an inline delay to the path for more variability from one PUF instance to another, i.e., these delays aid in preserving the uniqueness of the PUF [58]. The second adjustment is the addition of delays at the end of each chain. These delays are static and do not change with the inputs to the PUF. The purpose of these delays is to balance the amount of '0's and '1's produced by the PUF instance thereby providing the correct uniformity for the PUF's responses [59]. Other than these adjustments the PUF operates the same as before with a rising input edge which propagates through the chain of switches (and the added delays in this instance)

follow by adjudication by the arbiter and finally registration of the response in the Flip-Flop.

B.2.2 Arbiter-PUF FPGA Layout

In this section the details of the arbiter-PUF layout for the FPGA are discussed. Due to the complexity of producing randomized response from the PUF instance the components of the PUF are required to be manually layed out within the FPGA and the components place in what is known as a hard macro. These hard macros can be relocated on the FPGA in relation to a reference component as long as the relative structure of the hard macro can be placed (the components in the new location are the same as those in the original hard macro location).

There are four components that make up the arbiter-PUF FPGA instance: the switch, the variable stage delay component, the static delay component, and the arbitration latch. These components should be aligned sequentially within the slices of the FPGA as shown in Figure B.5 [5]. This sequential alignment allows for the routing of the connections between each of the components to be consistent as possible. The components are placed via a X-Y coordinate system. For a successful PUF implementation the X coordinate of the slice should remain the same while the Y coordinate is incremented.

Figure B.6 shows the layout of the switch and the variable delay components. For efficiency purposes these are implemented within the same slice for each stage. The switching component is implemented within one LUT which selects which input

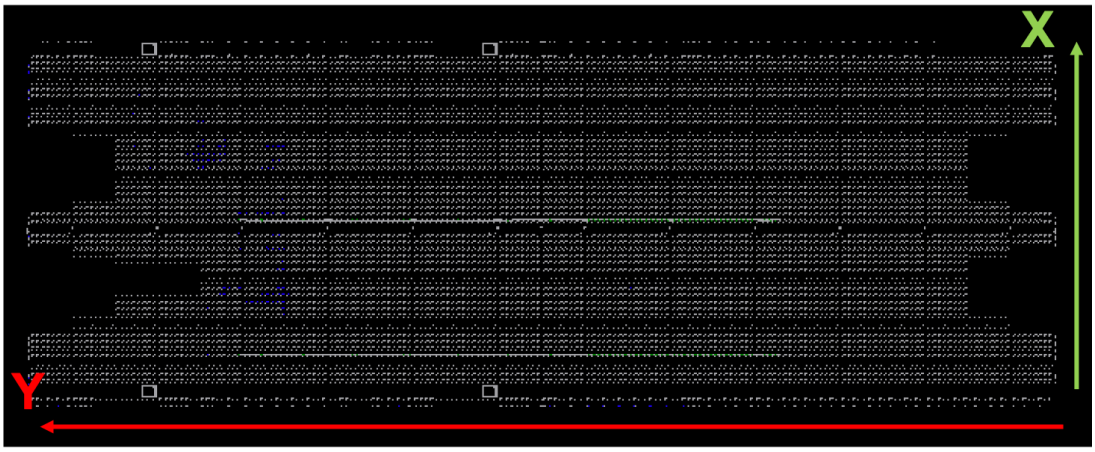


Figure B.5: Physical internal layout of the Spartan-6 FPGA each dot is a different component integrated on the chip. The directional layout for a successful PUF is shown by the red arrow.

should pass through on the path before the signal is routed out of the slice and back to the input of the variable delay LUT. The output of the variable delay is then passed to the next component (either another stage or the final static delay chain). There are two switch components for each stage of the arbiter-PUF one for the top path and one for the bottom path.

The static delay component is very similar to the variable delay however its input is set to a '0' or '1' during the build of the FPGA such that the input value does not change during operation. Figure B.7 shows the input to the component is passed through either the short delay or the long delay through the LUT before being passed on to either another static delay component or the arbitration latch.

The final component is the arbitration latch. This component is responsible for determining the response bit based on the difference in delays to its inputs. The arbitration latch is an S-R Latch consisting of two NAND gate implementations. The two NAND gates are implemented within separate LUTs within the same slice.

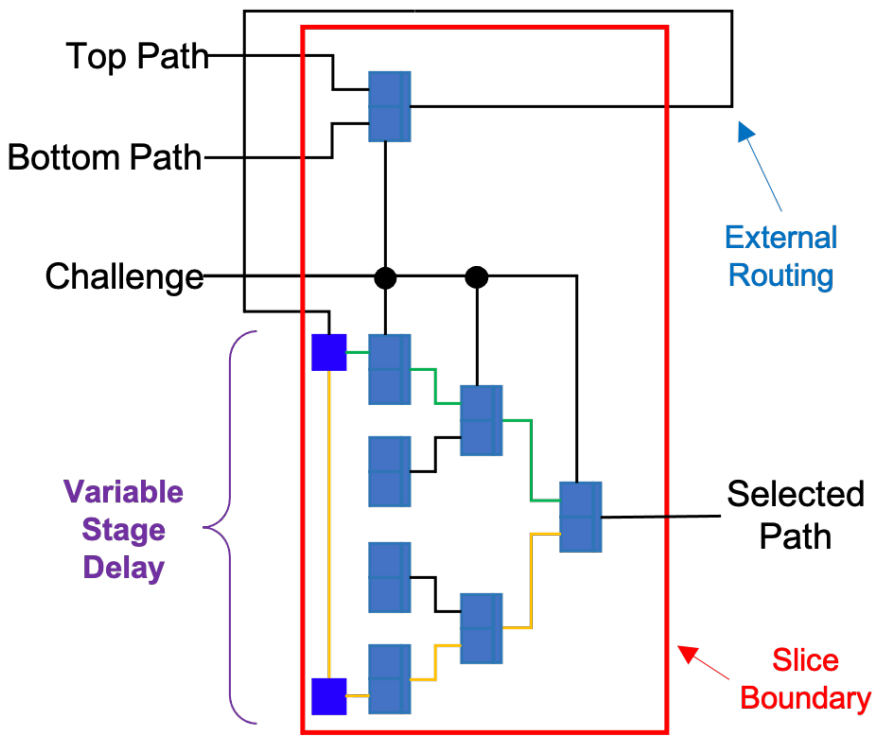


Figure B.6: Layout of the switching component and the variable stage delay within the FPGA. Note that the LUT implemented delay uses the first level to select between 0 and 1 of the selected path (dark blue component). The short delay through the LUT is shown by the green path whereas the long delay path is denoted by the orange path.

The outputs of the NAND gates (Q and $\bar{Q}$) exit the slice and return to the input completing the connections for the S-R Latch. The layout of the arbiter is shown in Figure B.8. This layout is important because the delays of the outputs (Q and $\bar{Q}$) must be matched otherwise there are an unreasonable amount of metastable outputs (outputs which oscillate between '0' and '1'). Another complication is that the delay is such that the output remains at the steady state of the arbiter, which is an output of 0. This phenomenon creates a disproportionate amount of '0's in the response output.

Recall that the overall layout of the arbiter-PUF instance is shown in Fig-

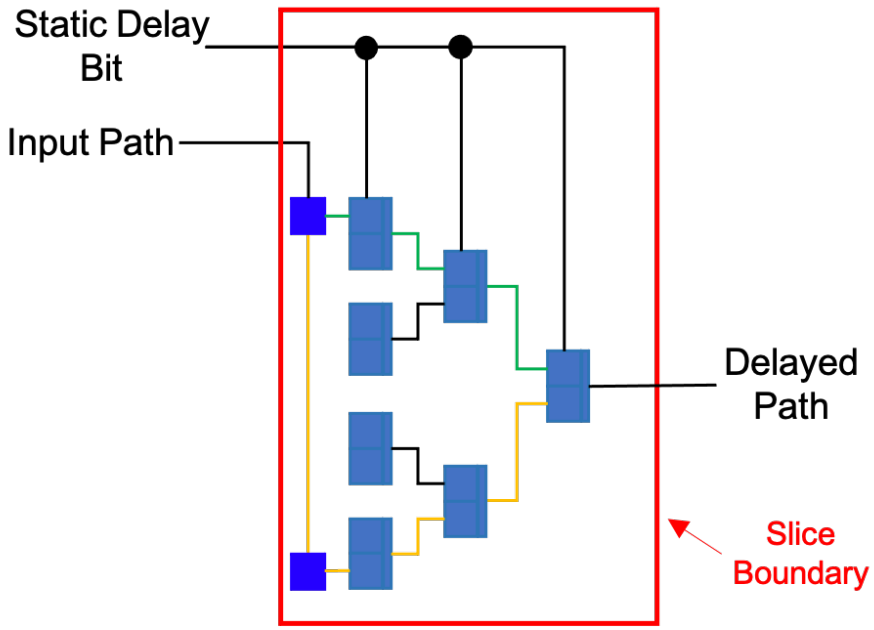


Figure B.7: Layout of the static delay component within the FPGA. Note that the LUT implemented delay uses the first level to select between 0 and 1 of the input path (dark blue component). The short delay through the LUT is shown by the green path whereas the long delay path is denoted by the orange path.

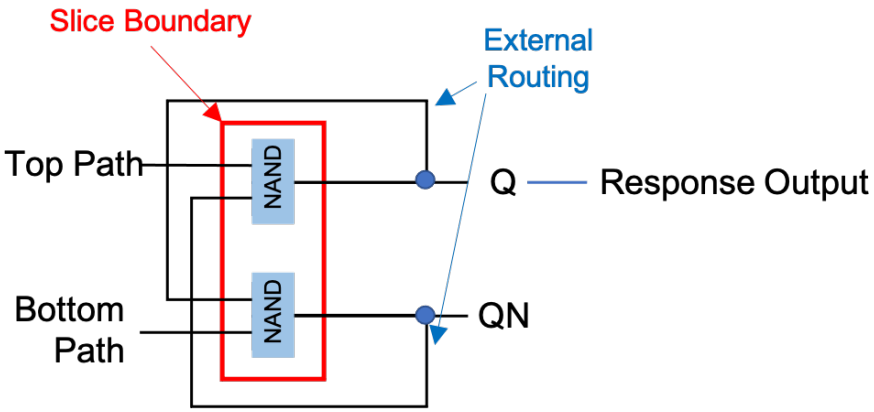


Figure B.8: Layout of the arbitration latch component within the FPGA.

ure B.4. This shows how the components are interconnected to produce the PUF within the FPGA. The Flip-Flop component, for the response output, within this figure is manually placed and its implementation is straightforward.

B.3 Required Metrics for Each PUF Instance

The metrics for a PUF implementation are discussed in Chapter 2 Section 2.1.2. The FPGA implementation was assessed for these metrics to ensure that it is a reasonable implementation.

Randomness

The indication that the response bits are random is defined by NIST standards [14]. The goal of these tests are to determine if there is any pattern to the output of the value generated by the PUF. The randomness values for a single PUF instance for 100,000 responses are shown in Table B.1. Notably the Random Excursion and Random Excursion Variants tests optimally require over one million input values [14].

Table B.1: NIST Randomness Test Results

Test	**Passed Percent**	**P-value**
Frequency	97%	0.416
Frequency Block	100%	0.494
Runs	99%	0.474
The Longest Run	97%	0.479
Binary Matrix Rank	100%	0.450
FFT	100%	0.511
Non-overlap. Template	97%	0.522
Overlapping Template	100%	0.999
Universal	100%	0.968
Linear Complexity Test	87.5%	0.325
Serial	100%	0.497
Approx. Entropy	100%	0.482
Cumulative Sums	96%	0.435
Random exc.	75%	0.532
Random exc. var.	87.5%	0.465

Uniqueness

The uniqueness metric ensures that there is significant variation across PUF designs to distinguish one PUF from another. Ideally there should be a 50% difference between the responses of different PUFs. The results for five implemented PUFs are shown in Table B.2. Here the average percentage of similar responses between PUF instances is 51.19%.

Table B.2: Uniqueness of the five PUF FPGA instances.

	PUF-5	PUF-4	PUF-3	PUF-2
PUF-1	37.13%	39.98%	64.09%	56.86%
PUF-2	46.93%	58.43%	54.25%	—
PUF-3	47.88%	47.46%	—	—
PUF-4	58.87%	—	—	—

Uniformity

The uniformity of a PUF is concerned with the balance of '1's and '0's on the output. Ideally there should be an equal balance of '0's and '1's in the response set. The uniformity of five PUF instances are shown in Table B.3. The average percent of '1's for all five PUF instances is 53.10%.

Table B.3: Uniformity of five PUF FPGA instances.

	Percent of Response = 0	Percent of Response = 1
PUF-1	50.90%	49.10%
PUF-2	35.86%	64.14%
PUF-3	51.48%	48.52%
PUF-4	45.00%	55.00%
PUF-5	51.25%	48.75%

Reliability/Stability

To assess the reliability, the PUF implementation is run multiple times and the percentage of differences for 12,000 responses are shown in Table B.4. The low

percentage difference between the runs shows that the PUFs can reliably produce the same response.

Table B.4: Reliability of five PUF FPGA instances replayed twice for 12,000 responses. Percentage shown is the difference between the two runs.

	Percent Difference
PUF-1	0.98%
PUF-2	0.88%
PUF-3	1.58%
PUF-4	0.52%
PUF-5	0.99%

B.4 Measuring Power Consumption

Power from the Sakura board was collected via a Teledyne Lecroy Waverunner 8254M Oscilloscope at 40GS/s via the amplified measurement port shown in Figure B.1. The voltage is amplified by 20dB through the integrated amplification circuit. The traces were collected from the rising input edge to the PUF through the output of the response Flip-Flop. This creates a power trace comprised of 16000 samples per collect for 400ns. The traces are shown in Figure 8.4 and Figure 8.5 of Chapter 8. Note that the PUF was designed to operate within a single clock cycle at a rate of 3.1875MHz. The physical collection from the board are shown in the photo of Figure B.9. Physically connected to the board is the JTAG programmer, a USB UART module, power to the board, the oscilloscope connection to the power amplification circuit, and a synchronization trigger output also connected to the oscilloscope.

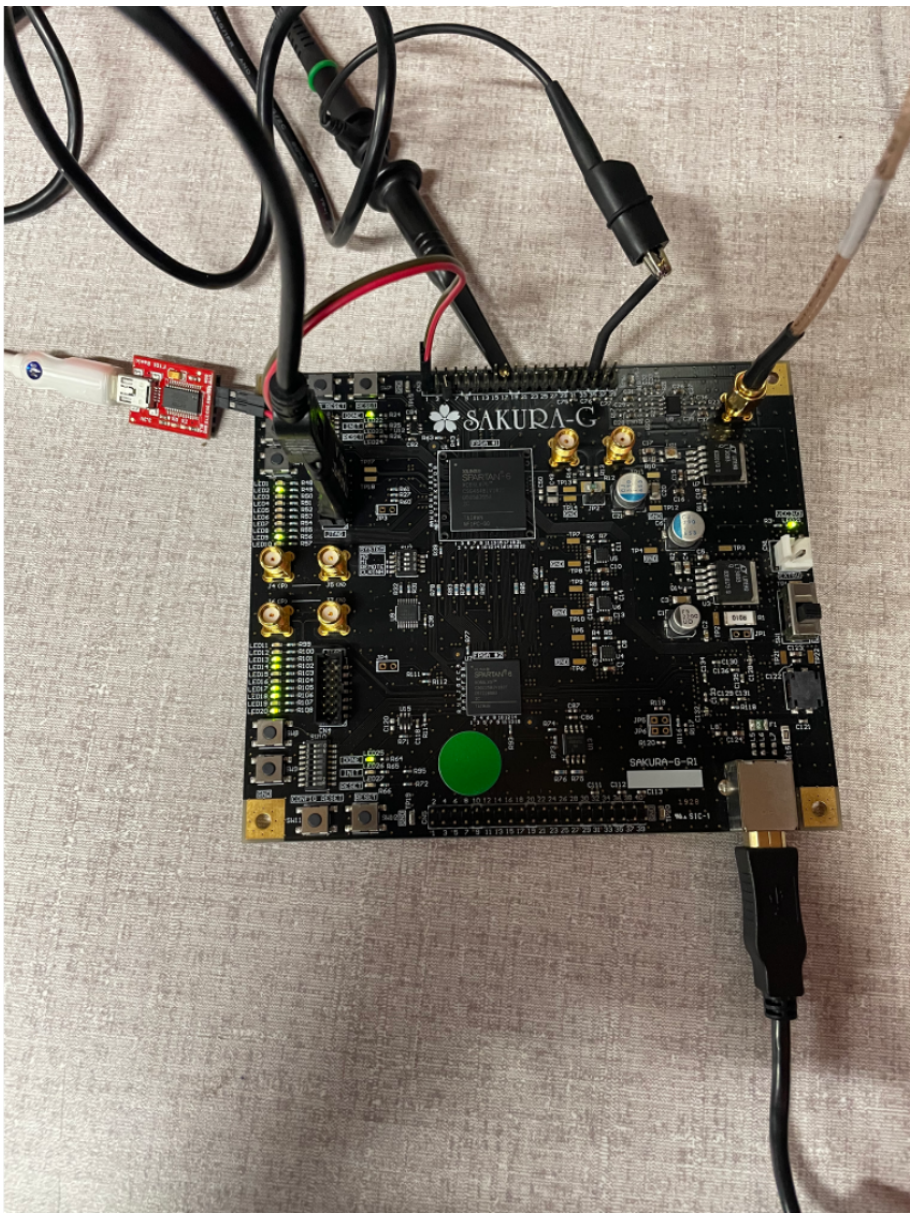


Figure B.9: Connected Sakura FPGA Board

Appendix C: Investigation Nuances

This appendix is used to catalog nuance investigations on the Cross-PUF attack presented within this work. There are two concerns:

1. What about arbiter-PUFs of larger size?

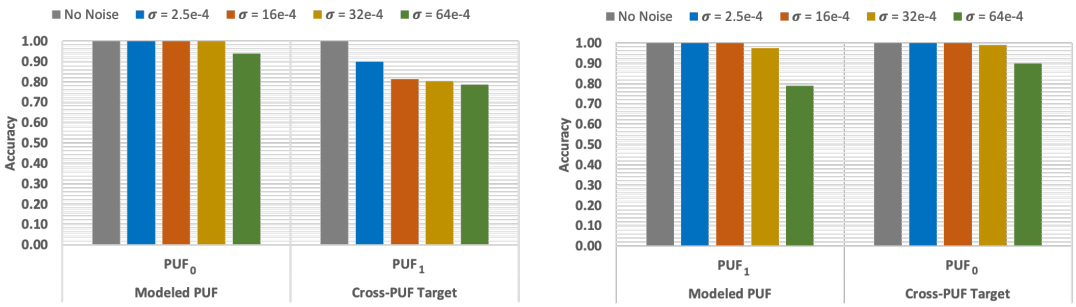
2. Does this really apply to derivatives of the arbiter-PUF?

Therefore this section is added to directly address these comments.

C.1 Investigations on PUF Size

When presenting the Cross-PUF attacks (Chapter 6) the size of the PUFs being investigated are 16-bits (meaning they have a 16-bit challenge and correspondingly 16 switch stages). Therefore to address these comments a 64-bit PUF implementation is investigated for its vulnerability to the Cross-PUF attack. The power traces for various sized PUFs are presented in Figure 4.16 as the figure shows the power consumption presents the same phenomenon regardless of the size of the PUF. Since the Cross-PUF attack focuses on attacking the response Flip-Flop which is present at the end of arbitration chain regardless of the size of the PUF it stands that the attack would be successful in this instance. Nonetheless, Cross-PUF at-

tacks for the 64-bit instances of the arbiter-PUF were performed and the results are shown in Figure C.1. These results indicate that the 64-bit PUF is similarly vulnerable to the the Cross-PUF attack displaying result with similar accuracies to the 16-bit PUF attacks.



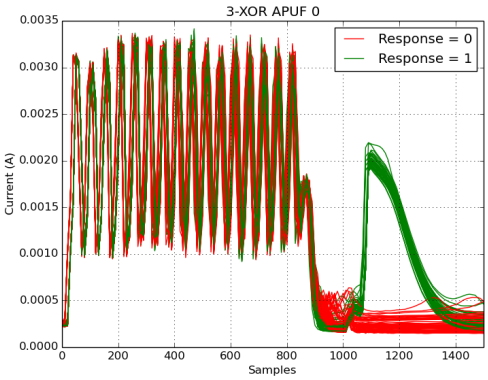
(a) 64-bit PUF-0 used for training.

(b) 64-bit PUF-1 used for training.

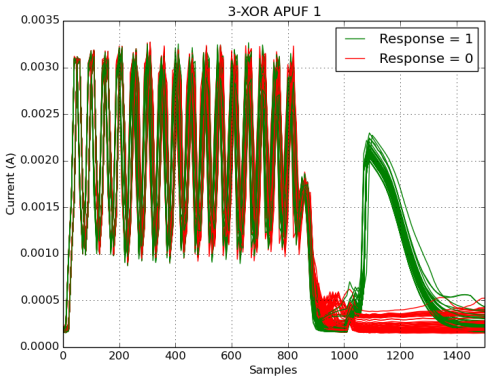
Figure C.1: Results of Cross-PUF attacks on 64-bit PUFs

C.2 Investigations on Arbiter-PUF Derivatives

A similar argument to the PUF size can be made for the Cross-PUF attack's potential success in attacking the derivatives of the arbiter-PUF. Namely the principle point of attack is the response Flip-Flop which is present at the output of every PUF instance. To investigate the efficacy of the Cross-PUF attack on arbiter-PUF derivatives investigations were performed on 3-XOR PUFs. The first observation to be made is that the traces of the 3-XOR PUFs have the same characteristics created by the response Flip-Flop as those of the original arbiter PUF shown in the traces displayed in Figure C.2. The results of performing the Cross-PUF attacks on the 3-XOR PUFs are shown in Figure C.3. These results indicate that the XOR PUF has the same vulnerability as its parent architecture, the arbiter-PUF.

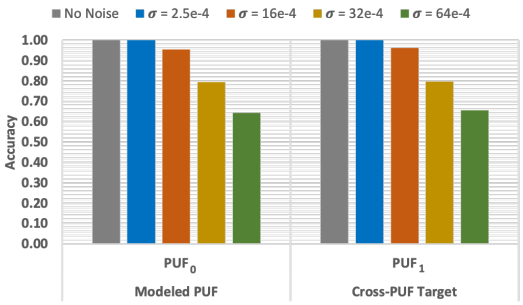


(a) Traces from 3-XOR PUF-0.

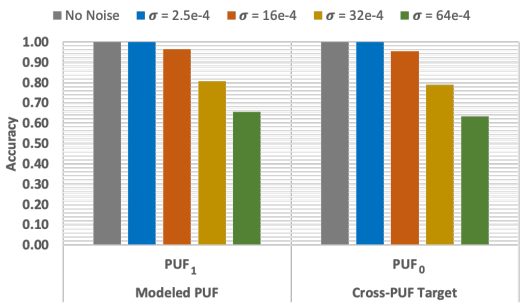


(b) Traces from 3-XOR PUF-1.

Figure C.2: Power traces (= 200) of 3-XOR PUFs.



(a) 3-XOR PUF_0 used for training.



(b) 3-XOR PUF_1 used for training.

Figure C.3: Results of Cross-PUF attacks on 3-XOR PUFs.

Bibliography

[1] Ise Design Suite. URL https://www.xilinx.com/products/design-tools/ise-design-suite.html

[2] Nangate 45nm open cell library. "http://www.nangate.com"

[3] RBF SVM parameters. "https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html#sphx-glr-auto-examples-svm-plot-rbf-parameters-py"

[4] SAKURA-G. URL https://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html

[5] Spartan-6 fpga family. URL https://www.xilinx.com/products/silicon-devices/fpga/spartan-6.html

[6] SAKURA-G: Side-channel AttacK User Reference Architecture - Specification. SAKURA Hardware Security Project (2013). URL https://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G_Spec_Ver1.0_English.pdf

[7] Aghaie, A., Moradi, A.: Ti-puf: Toward side-channel resistant physical unclonable functions. IEEE Transactions on Information Forensics and Security **15**, 3470–3481 (2020). DOI 10.1109/TIFS.2020.2986887

[8] Jiang et al., Q.: Two-Factor Authentication Protocol Using Physical Unclonable Function for IoV. In: IEEE/CIC ICCC, pp. 195–200 (2019)

[9] Elnaggar et al., R.: Machine learning for hardware security: Opportunities and risks. Journal of Electronic Testing **34**(2), 183–201 (2018). DOI 10.1007/s10836-018-5726-9

[10] Alkabani, Y., Koushanfar, F.: Active hardware metering for intellectual property protection and security **291-306**, 20 (2007)

[11] Alkatheiri, M.S., Zhuang, Y.: Towards fast and accurate machine learning attacks of feed-forward arbiter pufs. In: IEEE Conference on Dependable and Secure Computing, pp. 181–187 (2017). DOI 10.1109/DESEC.2017.8073845

[12] Association, A.I.: Counterfeit parts: Increasing awareness and developing countermeasures (2011). URL https://www.aia-aerospace.org/report/counterfeit-parts-increasing-awareness-and-developing-countermeasures/

[13] Ayodele, T.O.: Machine learning overview. New Advances in Machine Learning pp. 9–19 (2010)

[14] Bassham, L., Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Leigh, S., Levenson, M., Vangel, M., Heckert, N., Banks, D.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2010). URL https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762

[15] Becker, G.T., Kumar, R.: Active and passive side-channel attacks on delay based puf designs. IACR Cryptology ePrint Archive **2014**, 287 (2014). URL https://eprint.iacr.org/2014/287

[16] Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: M. Joye, J.J. Quisquater (eds.) Cryptographic Hardware and Embedded Systems - CHES 2004, pp. 16–29. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)

[17] Bruneau, N., Danger, J.L., Facon, A., Guilley, S., Hamaguchi, S., Hori, Y., Kang, Y., Schaub, A.: Development of the unified security requirements of pufs during the standardization process (2018). URL https://link.springer.com/chapter/10.1007/978-3-030-12942-2_24

[18] Chatterjee, D., Mukhopadhyay, D., Hazra, A.: Interpose PUF can be PAC Learned. IACR Cryptol. ePrint Arch. p. 471 (2020)

[19] Chatterjee, U., Govindan, V., Sadhukhan, R., Mukhopadhyay, D., Chakraborty, R.S., Mahata, D., Prabhu, M.M.: Building puf based authentication and key exchange protocol for iot without explicit crps in verifier database. IEEE Transactions on Dependable and Secure Computing **16**(3), 424–437 (2019). DOI 10.1109/TDSC.2018.2832201

[20] Cherif, Z., Danger, J.L., Guilley, S., Bossuet, L.: An easy-to-design puf based on a single oscillator: The loop puf (2012). DOI 10.1109/DSD.2012.22

[21] Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)

[22] Cui, X., Zhang, J., Wu, K., Garg, S., Karri, R.: Split manufacturing-based register transfer-level obfuscation. ACM Journal on Emerging Technologies in Computing Systems **15**(1) (2019). DOI 10.1145/3289156

[23] Delvaux, J., Verbauwhede, I.: Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise. In: IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 137–142 (2013). DOI 10.1109/HST.2013.6581579

[24] Devadas, S., Suh, E., Paral, S., Sowell, R., Ziola, T., Khandelwal, V.: Design and implementation of puf-based" unclonable" rfid ics for anti-counterfeiting and security applications. In: 2008 IEEE international conference on RFID, pp. 58–64. IEEE (2008)

[25] Eastland, N.: Structure of an fpga (2021). URL `https://digilent.com/blog/structure-of-an-fpga/`

[26] Ebrahimabadi, M., Lalouani, W., Younis, M., Karimi, N.: Countering puf modeling attacks through adversarial machine learning. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 356–361 (2021)

[27] Ebrahimabadi, M., Younis, M., Karimi, N.: Hardware assisted smart grid authentication. In: ICC 2021 - IEEE International Conference on Communications, pp. 1–6 (2021)

[28] Esbach, T., Fumy, W., Kulikovska, O., Merli, D., Schuster, D., Stumpf, F.: A new security architecture for smartcards utilizing pufs (2012). URL `https://link.springer.com/chapter/10.1007/978-3-658-00333-3_18`

[29] Fukushima, K., Souissi, Y., Hidano, S., Nguyen, R., Danger, J., Guilley, S., Nakano, Y., Kiyomoto, S., Sauvage, L.: Delay puf assessment method based on side-channel and modeling analyzes: The final piece of all-in-one assessment methodology. In: 2016 IEEE Trustcom/BigDataSE/ISPA, pp. 201–207 (2016). DOI 10.1109/TrustCom.2016.0064

[30] Ganji, F., Tajik, S.: Physically unclonable functions and ai: Two decades of marriage. ArXiv **abs/2008.11355** (2020)

[31] Gao, Y., Li, G., Ma, H., Al-Sarawi, S.F., Kavehei, O., Abbott, D., Ranasinghe, D.C.: Obfuscated challenge-response: A secure lightweight authentication mechanism for puf-based pervasive devices. In: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), pp. 1–6 (2016). DOI 10.1109/PERCOMW.2016.7457162

[32] Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS '02, p. 148–160. Association for Computing Machinery, New York, NY, USA (2002). DOI 10.1145/586110.586132. URL `https://doi.org/10.1145/586110.586132`

[33] Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Delay-based circuit authentication and applications. In: Proceedings of the 2003 ACM Symposium on Applied Computing, SAC '03, p. 294–301. Association for Computing Machinery, New York, NY, USA (2003). DOI 10.1145/952532.952593. URL https://doi.org/10.1145/952532.952593

[34] Genkin, D., Shamir, A., Tromer, E.: Rsa key extraction via low-bandwidth acoustic cryptanalysis. In: Annual Cryptology Conference, pp. 444–461. Springer (2014)

[35] Gu, C., Chang, C.H., Liu, W., Yu, S., Ma, Q., O'neill, M.: A modeling attack resistant deception technique for securing puf based authentication. In: 2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 1–6 (2019). DOI 10.1109/AsianHOST47458.2019.9006710

[36] Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: Fpga intrinsic pufs and their use for ip protection. In: P. Paillier, I. Verbauwhede (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, pp. 63–80. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[37] Guerin, C., Huard, V., Bravaix, A.: The energy-driven hot-carrier degradation modes of nmosfets. IEEE Transactions on Device and Materials Reliability **7**(2), 225–235 (2007). DOI 10.1109/TDMR.2007.901180

[38] Guin, U.: Establishment of trust and integrity in modern supply chain from design to resign. Ph.D. thesis (2016). URL https://opencommons.uconn.edu/dissertations/1063/

[39] Guin, U., Huang, K., DiMase, D., Carulli, J.M., Tehranipoor, M., Makris, Y.: Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. Proceedings of the IEEE **102**(8), 1207–1228 (2014). DOI 10.1109/JPROC.2014.2332291

[40] Haber, H.: The expected value and variance of an average of iid random variables (2012). URL http://scipp.ucsc.edu/~haber/ph116C/iid.pdf

[41] Herder et al., C.: Physical unclonable functions and applications: A tutorial. Proc. of the IEEE **102**(8), 1126–1141 (2014)

[42] ISO/IEC 20897-1:2020: Information security, cybersecurity and privacy protection — Physically unclonable functions — Part 1: Security requirements. Standard, International Organization for Standardization (2020)

[43] Joshi, S., Mohanty, S., Kougianos, E.: Everything you wanted to know about pufs. IEEE Potentials **36**, 38–46 (2017). DOI 10.1109/MPOT.2015.2490261

[44] Karimi, N., Danger, J., Guilley, S.: Impact of aging on the reliability of delay pufs. J. Electron. Test. **34**(5), 571–586 (2018). DOI 10.1007/s10836-018-5745-6. URL https://doi.org/10.1007/s10836-018-5745-6

[45] Karimi, N., Huang, K.: Prognosis of NBTI aging using a machine learning scheme. In: 2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2016, Storrs, CT, USA, September 19-20, 2016, pp. 7–10. IEEE Computer Society (2016). DOI 10.1109/DFT.2016.7684060. URL https://doi.org/10.1109/DFT.2016.7684060

[46] Khan, S., Haron, N.Z., Hamdioui, S., Catthoor, F.: Nbti monitoring and design for reliability in nanoscale circuits. In: 2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, pp. 68–76 (2011). DOI 10.1109/DFT.2011.49

[47] Koushanfar, F.: Hardware metering: A survey (2010). DOI 10.1007/978-1-4419-8080-9_5

[48] Koushanfar, F.: Integrated circuits metering for piracy protection and digital rights management: An overview. pp. 449–454 (2011). DOI 10.1145/1973009.1973110

[49] Kroeger, T., Cheng, W., Guilley, S., Danger, J.L., Karimi, N.: Cross-puf attacks on arbiter-pufs through their power side-channel. In: IEEE International Test Conference (ITC), pp. 1–5 (2020). DOI 10.1109/ITC44778.2020.9325241

[50] Kroeger, T., Cheng, W., Guilley, S., Danger, J.L., Karimi, N.: Effect of aging on puf modeling attacks based on power side-channel observations. In: Design, Automation Test in Europe Conference Exhibition (DATE), pp. 454–459 (2020). DOI 10.23919/DATE48585.2020.9116428

[51] Kroeger, T., Cheng, W., Guilley, S., Danger, J.L., Karimi, N.: Making obfuscated pufs secure against power side-channel based modeling attacks. In: Design, Automation Test in Europe Conference Exhibition (DATE) (2021)

[52] Kumar, R., Jovanovic, P., Polian, I.: Precise fault-injections using voltage and temperature manipulation for differential cryptanalysis. In: 2014 IEEE 20th International On-Line Testing Symposium (IOLTS), pp. 43–48. IEEE (2014)

[53] La Rosa, G., Guarin, F., Rauch, S., Acovic, A., Lukaitis, J., Crabbe, E.: Nbti-channel hot carrier effects in pmosfets in advanced cmos technologies. In: 1997 IEEE International Reliability Physics Symposium Proceedings. 35th Annual, pp. 282–286 (1997). DOI 10.1109/RELPHY.1997.584274

[54] Lalouani, W., Ebrahimabadi, M., Younis, M., Karimi, N.: Countering Modeling Attacks in PUF-based IoT Security Solutions. In: ACM Journal on Emerging Technologies in Computing Systems (JETC) (2021)

[55] Lalouani, W., Younis, M., Ebrahimabadi, M., Karimi, N.: Robust and efficient data security solution for pervasive data sharing in iot (2022)

[56] Lopez, T.: Dod adopts 'zero trust' approach to buying microelectronics. "https://www.defense.gov/Explore/News/Article/Article/2192120/dod-adopts-zero-trust-approach-to-buying-microelectronics/" (2020)

[57] Mahmoud, A., Rührmair, U., Majzoobi, M., Koushanfar, F.: Combined Modeling and Side Channel Attacks on Strong PUFs. IACR Cryptology ePrint Archive **2013**, 632 (2013)

[58] Maiti, A., Gunreddy, V., Schaumont, P.: A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions, pp. 245–267. Springer New York, New York, NY (2013). DOI 10.1007/978-1-4614-1362-2_11

[59] Majzoobi, M., Koushanfar, F., Devadas, S.: Fpga puf using programmable delay lines. In: 2010 IEEE International Workshop on Information Forensics and Security, pp. 1–6 (2010). DOI 10.1109/WIFS.2010.5711471

[60] Majzoobi, M., Koushanfar, F., Potkonjak, M.: Lightweight secure pufs. In: 2008 IEEE/ACM International Conference on Computer-Aided Design, pp. 670–673 (2008). DOI 10.1109/ICCAD.2008.4681648

[61] Mak, M.A.: Assessing DODs assured access to microelectronics in support of U.S. national security requirements: hearing before the Subcommittee on Oversight and Investigations of the Committee on Armed Services, House of Representatives, One Hundred Fourteenth Congress, first session, hearing held October 28, 2015 (2015)

[62] Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer (2006)

[63] Mansouri, S.S., Dubrova, E.: Ring oscillator physical unclonable function with multi level supply voltages. In: 2012 IEEE 30th International Conference on Computer Design (ICCD), pp. 520–521 (2012). DOI 10.1109/ICCD.2012.6378703

[64] Mars, A., Adi, W.: New Concept for Physically-Secured E-Coins Circulations. In: Adaptive Hardware and Systems, pp. 333–338 (2018)

[65] Menhorn, N.: External secure storage using the puf. Application Note: Zynq UltraScale Devices, XAPP1333 (2018). URL https://www.xilinx.com/support/documentation/application_notes/xapp1333-external-storage-puf.pdf

[66] Mohan, K.M.: Outsourcing trends in semiconductor industry. Ph.D. thesis, Massachusetts Institute of Technology (2010)

[67] Nguyen et al., P.H.: The Interpose PUF: Secure PUF Design against State-of-the-art Machine Learning Attacks. IACR Transactions on Cryptographic

Hardware and Embedded Systems (CHES) **2019**(4), 243–290 (2019). DOI 10.13154/tches.v2019.i4.243-290

[68] Nozaki, Y., Yoshikawa, M.: Em based machine learning attack for xor arbiter puf. In: Proceedings of the 2nd International Conference on Machine Learning and Soft Computing, ICMLSC '18, p. 19–23. Association for Computing Machinery, New York, NY, USA (2018). DOI 10.1145/3184066.3184100. URL https://doi.org/10.1145/3184066.3184100

[69] Oboril, F., Tahoori, M.B.: Extratime: Modeling and analysis of wearout due to transistor aging at microarchitecture-level. In: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), pp. 1–12 (2012). DOI 10.1109/DSN.2012.6263957

[70] Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science **297**(5589), 2026–2030 (2002). DOI 10.1126/science.1074376

[71] Pouyan, P.: Reliability-aware memory design using advanced reconfiguration mechanisms. Ph.D. thesis (2015)

[72] Ravi, S.: Semiconductor industry association - 2020 factbook. "https://www.semiconductors.org/resources/factbook/" (2020)

[73] Rohatgi, P.: Electromagnetic attacks and countermeasures. In: Cryptographic Engineering, pp. 407–430. Springer (2009)

[74] Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: CCS, pp. 237–249 (2010). DOI 10.1145/1866307.1866335

[75] Rührmair, U., Sölter, J.: PUF modeling attacks: An introduction and overview. In: DATE, pp. 1–6 (2014). DOI 10.7873/DATE.2014.361

[76] Samonas, S., Coss, D.: The cia strikes back: Redefining confidentiality, integrity and availability in security. Journal of Information System Security **10**(3), 21–45 (2014)

[77] Scholz, A., Zimmermann, L., Sikora, A., Tahoori, M.B., Aghassi-Hagmann, J.: Embedded analog physical unclonable function system to extract reliable and unique security keys (2020). URL https://www.mdpi.com/2076-3417/10/3/759/htm

[78] Sehwag, V., Saha, T.: Tv-puf: A fast lightweight analog physical unclonable function. In: 2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS), pp. 182–186 (2016). DOI 10.1109/iNIS.2016.049

[79] Speers, T.: Polarfire non-volatile fpga family delivers ground breaking value: Best-in-class security (2018). URL https://www.microsemi.com/blog/2018/04/10/polarfire-non-volatile-fpga-family-delivers-ground-breaking-value-best-in-cla

[80] Synopsys: HSPICE User Guide: Basic Simulation and Analysis (2016)

[81] Tehranipoor, F.: Design and architecture of hardware-based random function security primitives. Ph.D. thesis (2017). URL `https://opencommons.uconn.edu/dissertations/1512`

[82] Vatajelu, E.I., Natale, G.D., Mispan, M.S., Halak, B.: On the encryption of the challenge in physically unclonable functions. In: 2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 115–120 (2019). DOI 10.1109/IOLTS.2019.8854387

[83] Verbauwhede, I.: Secure Integrated Circuits and Systems (2010). DOI 10.1007/978-0-387-71829-3

[84] Vijayakumar, A., Kundu, S.: A novel modeling attack resistant puf design based on non-linear voltage transfer characteristics. In: 2015 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 653–658 (2015). DOI 10.7873/DATE.2015.0522

[85] Wang, Q., Gao, M., Qu, G.: A machine learning attack resistant dual-mode puf. pp. 177–182 (2018). DOI 10.1145/3194554.3194590

[86] Wang, S., Chen, Y., Li, K.S.: Adversarial attack against modeling attack on pufs. In: 2019 56th ACM/IEEE Design Automation Conference (DAC), pp. 1–6 (2019)

[87] Wisiol et al., N.: Splitting the Interpose PUF: A Novel Modeling Attack Strategy. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(3), 97–120 (2020). DOI 10.13154/tches.v2020.i3.97-120

[88] Xu, T., Wendt, J.B., Potkonjak, M.: Security of iot systems: Design challenges and opportunities. In: 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 417–423 (2014). DOI 10.1109/ICCAD.2014.7001385

[89] Yu, M., Sowell, R., Singh, A., M'Raïhi, D., Devadas, S.: Performance metrics and empirical results of a puf cryptographic key generation asic. In: 2012 IEEE International Symposium on Hardware-Oriented Security and Trust, pp. 108–115 (2012). DOI 10.1109/HST.2012.6224329

[90] Yue, S., Li, P., Hao, P.: Svm classification: Its contents and challenges. Applied Mathematics-A Journal of Chinese Universities **18**(3), 332–342 (2003)

[91] Zalivaka, S.S., Ivaniuk, A.A., Chang, C.: Reliable and modeling attack resistant authentication of arbiter puf in fpga implementation with trinary quadruple response. IEEE Transactions on Information Forensics and Security **14**(4), 1109–1123 (2019). DOI 10.1109/TIFS.2018.2870835

[92] Zhou, C., Jenkins, K.A., Chuang, P.I., Vezyrtzis, C.: Effect of hci degradation on the variability of mosfets. In: 2018 IEEE International Reliability Physics Symposium (IRPS), pp. P–RT.1–1–P–RT.1–4 (2018). DOI 10.1109/IRPS.2018.8353684

[93] Zhou, C., Parhi, K.K., Kim, C.H.: Secure and reliable xor arbiter puf design: An experimental study based on 1 trillion challenge response pair measurements. In: Proceedings of the 54th Annual Design Automation Conference 2017, DAC '17. Association for Computing Machinery, New York, NY, USA (2017). DOI 10.1145/3061639.3062315. URL https://doi.org/10.1145/3061639.3062315

[94] Öztürk, E., Hammouri, G., Sunar, B.: Towards robust low cost authentication for pervasive devices. In: 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 170–178 (2008). DOI 10.1109/PERCOM.2008.54