Secure and Unclonable Integrated Circuits

by

Kleber Hugo Stangherlin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Nicola Nicolici Professor, Dept. of Electrical and Computer Engineering, McMaster University
Supervisor(s):	Manoj Sachdev Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Catherine Gebotys Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal-External Member:	Alfred Menezes Professor, Dept. of Combinatorics and Optimization, University of Waterloo
Other Member(s):	Andrew Kennings Professor, Dept. of Electrical and Computer Engineering, University of Waterloo

Author's Declaration

Abstract

Semiconductor manufacturing is increasingly reliant in offshore foundries, which has raised concerns with counterfeiting, piracy, and unauthorized overproduction by the contract foundry. The recent shortage of semiconductors has aggravated such problems, with the electronic components market being flooded by recycled, remarked, or even out-of-spec, and defective parts. Moreover, modern internet connected applications require mechanisms that enable secure communication, which must be protected by security countermeasures to mitigate various types of attacks. In this thesis, we describe techniques to aid counterfeit prevention, and mitigate secret extraction attacks that exploit power consumption information.

Counterfeit prevention requires simple and trustworthy identification. Physical unclonable functions (PUFs) harvest process variation to create a unique and unclonable digital fingerprint of an IC. However, learning attacks can model the PUF behavior, invalidating its unclonability claims. In this thesis, we research circuits and architectures to make PUFs more resilient to learning attacks. First, we propose the concept of non-monotonic response quantization, where responses not always encode the best performing circuit structure. Then, we explore the design space of PUF compositions, assessing the trade-off between stability and resilience to learning attacks. Finally, we introduce a lightweight key based challenge obfuscation technique that uses a chip unique secret to construct PUFs which are more resilient to learning attacks.

Modern internet protocols demand message integrity, confidentiality, and (often) non-repudiation. Adding support for such mechanisms requires on-chip storage of a secret key. Even if the key is produced by a PUF, it will be subject to key extraction attacks that use power consumption information. Secure integrated circuits must address power analysis attacks with appropriate countermeasures. Traditional mitigation techniques have limited scope of protection, and impose several restrictions on how sensitive data must be manipulated. We demonstrate a bit-serial RISC-V microprocessor implementation with no plain-text data in the clear, where all values are protected using Boolean masking and differential domino logic. Software can run with little to no countermeasures, reducing code size and performance overheads. Our methodology is fully automated and can be applied to designs of arbitrary size or complexity. We also provide details on other key components such as clock randomizer, memory protection, and random number generator.

Acknowledgements

This PhD would not have been possible without the unconditional encouragement of my wife. Experience tells me she will support anything I believe in, except for Alien life, and the number of dogs we should have. For my entire family, thank your love and comprehension throughout these rough years I stayed so far away from you all.

To my supervisor, and friend, professor Manoj Sachdev, my deepest and sincere gratitude for believing in a young Brazilian, almost too old to be seen as a student. I will not forget how you always had a clever story to tell me at those moments when I had run out of energy to keep pushing, specially during the rigorous lockdown years. After all, the tail of the elephant is almost passing through.

Finally, I would like to thank the committee for revising this thesis. You have found time to look at my work, and I sincerely hope it does not disappoint. I will take your feedback, the good and the bad, to improve and grow as a possible scientist, and above all, as a human being.

Table of Contents

List of Abbreviations

AES	advanced encryption standard.
ALU	arithmetic logic unit.
AND	logic “and”.
APUF	arbiter PUF.
APUF-POP	PUF-on-PUF composition of APUFs.
ASIC	application specific integrated circuit.
BER	bit error rate.
BM	boolean masking.
BR-PUF	bistable ring PUF.
BUF	buffer register.
CASR	cellular automata shift register.
CMA	covariance matrix adaptation.
CMOS	complementary metal-oxide semiconductor.
CPA	correlation power analysis.
CPU	central processing unit.
CRP	challenge-response pair.

CSR	control and status register.
CTRL	control logic.
DDL	differential domino logic.
DNN	deep-neural network.
DTW	dynamic time warping.
EDA	electronic design automation.
ES	evolution strategies.
FPGA	field programmable gate arrays.
GBW	gain bandwidth product.
GPIB	general purpose interface bus.
GPU	graphics processing unit.
HD	hamming distance.
HW	hamming weight.
IC	integrated circuit.
ID	instruction decoder.
IO	input output.
JTAG	Joint Test Action Group.
LFSR	linear feedback shift register.
LR	logistic regression.
LSB	least significant bit.
MOSFET	metal-oxide-semiconductor field-effect transistor.

MPUF	multi PUF.
NAND	logic “and” complemented.
NCM	no countermeasures.
NIST	National Institute of Standards and Technology.
NLFSR	non-linear feedback shift register.
NMQ	non-monotonic response quantization.
NMQ-PUF	non-monotonically quantized strong PUF.
NMQ-RO	ring oscillator based non-monotonically quantized PUF.
OR	logic “or”.
OTP	one time programmable.
PC	program counter.
POP	PUF-on-PUF.
PRNG	pseudo random number generator.
PUF	physical unclonable function.
RD	destination register.
RF	register file.
RISC	reduced instruction set computer.
RNG	random number generator.
RO	ring oscillator.
RO-PUF	ring oscillator PUF.
RS	source register.
RTL	register transfer level.

SABL sense amplifier based logic.

SAC strict avalanche criterion.

SBOX substitution box.

SIA Semiconductor Industry Association.

SRAM static random access memory.

TI threshold implementation.

TMV temporal majority voting.

TVLA test vector leakage assessment.

UP microprocessor.

WDDL wave dynamic differential logic.

XNOR logic “exclusive or” complemented.

XOR logic “exclusive or”.

XOR-APUF xored composition of APUFs.

XOR-NMQ-RO xored composition of NMQ-ROs.

Chapter 1

Introduction

Semiconductor manufacturing is increasingly reliant in offshore foundries, which has raised concerns with counterfeiting, piracy, and unauthorized overproduction by the contract foundry [40]. The revenue lost by legitimate companies was estimated in \$100 Billion per year [70]. The recent shortage of semiconductors has aggravated such problems, with the electronic components market being flooded by recycled, remarked, or even out-of-spec, and defective parts [120]. The Semiconductor Industry Association (SIA) estimates that 15% of all spare and replacement semiconductors purchased by the Pentagon are counterfeit [5]. Quoting general Patrick O'Reilly in 2011, director of the missile defense agency, "We do not want a \$12 million missile defense interceptor's reliability compromised by a \$2 counterfeit part" [68].

Counterfeit prevention requires simple and trustworthy identification. Traditional solutions would selectively burn polysilicon fuses that modify the value of a publicly readable identifier. Attackers were able to inspect, probe, and reconnect fuses using easily accessible laboratory equipment [49]. Memory technologies like flash, offer a safer storage alternative, where logic values are represented by the amount of trapped charges inside the floating gate of a memory cell. When using flash memories, a particular sector may be reserved for one-time programmable data—which is easily achieved using digital logic. Probing techniques still pose risks for flash memories, but it is considerably harder to tamper or passively read the stored values [4, 109].

Having a publicly readable identifier stored in flash does not stop other circuits from impersonating (cloning) an authentic integrated circuit (IC). Therefore, designers often implement a challenge-response system, where a chip unique secret value is used as key for a cryptographic operation. In this context, the device acts as an encryption oracle, producing encrypted responses for externally provided challenges. The external verifier knows the secret key stored in memory, so it can validate the produced responses for each of the inquired challenges. To avoid replay attacks, a challenge is never be used more than once.

The challenge-response system described above has two main weaknesses, i) the extra cost of the encryption hardware might be prohibitive for some resource constrained applications; and ii) if the secret key leaks, it is not possible to differentiate between genuine and cloned devices. While the extra cost is relevant only for certain niches, the threat of exposed keys affects all applications. For example, overproduced ICs have a blank flash memory, therefore, if the correct key is known, they can be programmed to behave like any other authentic device. Moreover, manipulating external data (challenge) with a secret key leaks sensitive information in the power consumption, which can lead to key extraction using power analysis attacks [48].

Distinguishing between genuine and cloned devices requires an identification method similar to human fingerprints, where uniqueness and unclonability emerge from very complex biological processes. In the context of semiconductor manufacturing, the unique and unclonable properties must emerge from the inherent variability of the manufacturing process. Objects with unclonable properties were already used during the cold war to identify nuclear weapons [35]. They sprayed a thin coating of light-reflecting particles onto the weapon's surface. When it was illuminated from various angles, the randomly distributed particles generated unique inference patterns that are not easily reproducible.

The first known physical unclonable function (PUF) was introduced in [69]. The device used a laser pointed at a stationary scattering medium to observe the unique speckle pattern exiting the structure. The first silicon implementation of PUFs was proposed in [33], and named arbiter PUF (APUF). The APUF has two signals racing through identically designed delay paths. The time it takes to traverse the paths depends on the manufacture variability of each delay cell. The input challenge selects two unique, nominally identical, delay paths. At the end, an arbiter determines the response based on which signal arrived first.

As initially proposed in [33], the APUF has one main advantage over key based challenge-response systems: the device's memory does not contain secret information. PUFs give ICs the capability of unique responses, irrespective of how their memory is programmed. Probing attacks are not able to extract individual gate delays. Exhaustive enumeration is impractical for challenge spaces of 64, or 128 bits. Moreover, since responses are taken from a differential measurement, they are mostly stable under different environmental conditions—reported APUF bit error rates are typically below 5% [87]. Achieving low bit error rate in circuits that measure manufacture variability is hard, therefore, practical solutions often use error correction, or tolerance margins.

Authenticating a PUF enabled device requires an enrollment phase, which occurs in a secure environment. During enrollment, a randomly selected set of challenges is evaluated and responses are stored in a secure database. When verifying the authenticity of a device, the PUF is inquired with a subset of the enrolled challenge-response pairs (CRPs). If responses match the stored values, the device is deemed authentic. An error margin is typically acceptable, since responses for some challenges are less stable than others. Challenges are never used more than once to avoid replay attacks. Notice that practical use of PUFs still requires the programming of a public chip identifier, which is used to associate an enrolled CRP database to a particular device.

From their initial introduction in [33], PUFs attracted considerable attention from the security community. The possibility of dismissing secret keys and use a lightweight, secure identification system capable of distinguishing between genuine and cloned devices is very appealing. Nevertheless, researchers later discovered that PUF security is significantly more fragile than initially thought. Using a subset of challenge-response pairs, attackers can model the PUF entropy source, and predict responses for unseen challenges with high accuracy. Early modeling work was done in [55], and later consolidated in [76], where authors report 95% prediction accuracy for an APUF, using only 640 CRPs.

Following the seminal work in [33], PUFs were divided in two categories: strong and weak PUFs. The classification criterion is based on challenge space size. Strong PUFs have a large challenge space, such that exhaustive enumeration is impractical. Weak PUFs, on the other hand, have a much smaller challenge space. Therefore, weak PUF responses are not returned to the external user, but used as secret key material for other cryptographic engines. In such key generation applications, response stability is crucial. A single wrong key bit results in an incomprehensible output, and possibly lead to analytical key extraction attacks known as differential fault analysis [18]. Weak PUF implementations solve the response stability problem using error correction data which is programmed into a one time programmable (OTP) flash during test.

Weak PUFs are effective at preventing the foundry from overproducing ICs. Even though the overproduced devices have the same design, their embedded weak PUF ensures that two ICs will not behave alike. However, similarly to flash memories, weak PUFs are still susceptible to key extraction attacks, including probing, and possibly power analysis attacks as well [43, 48, 109]. If the weak PUF responses are leaked, impersonating devices may be created. The first weak PUF publication is a patent which dates back from 2002 [51]. Implementation results first appeared in [96], with an SRAM based design in

130 nm technology, and 90% bit stability. Pioneering work on SRAM based weak PUFs was also performed in [41, 42].

The development of learning resistant strong PUFs has been an active field of research for the past two decades. The first attempt at improving the security of strong PUF was to add "control" logic to the APUF, and was later referred to as controlled PUF [32, 33]. The controlled PUF surrounds the APUF with hash operations, which are applied to the input challenge and output responses. External helper data is used to correct errors from unstable responses. In addition to the added cost of the hash, controlled PUFs were shown vulnerable to attacks that manipulate the external helper data [26, 95].

Other strong PUF architectures use compositions to create a strong PUFs with enhanced security properties [52, 66, 78, 80, 116]. Similarly to the mathematical notion of composite functions, PUF compositions use the output of PUF instances as input to other instances. The resulting PUF has increased resilience to learning attacks, however, response bit error rate of the overall composition limits the achievable benefits.

In this thesis, we research circuits and architectures to make PUFs more resilient to learning attacks. Our results are presented in chapters 3, 4, and 5.

In chapter 3, we propose the concept of non-monotonic response quantization for strong PUFs [91]. Responses depend not only on which path is faster, but also on the distance between the arriving signals. Our experiments show that the resulting PUF has increased security against learning attacks. We report uniformity, uniqueness, and bit error rate measurements from a testchip fabricated in 65 nm technology.

In chapter 4, we explore the design space and assess the security of several PUF compositions [90, 116]. We extend previous techniques of influential bits to assess stage bias in APUF instances. Our data shows that compositions do not always preserve the security properties of PUFs. We report uniformity, uniqueness, and bit error rate measurements from a testchip fabricated in 65 nm technology.

In chapter 5, we propose a secure and lightweight key based challenge obfuscation for strong PUFs [93]. Our obfuscation mechanism uses non-linear feedback shift registers. Responses are directly provided to the user, without error correction or extra post-processing steps. We also discuss the cost of protecting our architecture against power analysis attacks. Security against learning attacks is assessed using avalanche criterion, and deep-neural networks.

Counterfeit prevention establishes trust between the user and the hardware. Most modern internet connected applications, however, requires the implementation of mechanisms for message integrity, confidentiality, and (often) non-repudiation. Adding support for such mechanisms requires on-chip storage of a secret key. Even if the key is produced by a weak PUF, it will be subject to key extraction attacks. In particular, power analysis attacks offer, perhaps, the greatest threat since it is non-invasive and can be performed with inexpensive laboratory equipment [48].

Power analysis attacks use a set of power consumption traces to extract secret information from a device. Traces record the device manipulating the secret value with different input data. For example, recorded power traces may include AES encryptions using the same key but different plaintexts. A few key bits are extracted each iteration. The attacker chooses a key candidate and creates a power consumption hypothesis for each input data, using knowledge of the implemented algorithm, and a leakage model (typically hamming-weight). The correlation between the hypothesis vector and the recorded power traces will be the highest when the correct key is used [48].

Secure integrated circuits must address power analysis attacks with appropriate countermeasures. Most countermeasures fit into two categories: masking and hiding. Masking uses random numbers to hide the value of intermediate computations [61]. Therefore, the power consumption is, in theory, uncorrelated to the actual data being processed. Hiding techniques, on the other hand, focus on making power consumption independent of the values being computed. Popular examples are differential input/output dynamic logic styles with precharge/evaluation phases [74]. Other countermeasures insert random delays during algorithm execution, causing an intentional misalignment in the recorded power traces which reduces the effectiveness of power analysis attacks.

Traditional power analysis mitigation techniques have limited scope of protection, and impose several restrictions on how sensitive data must be manipulated. In chapter 6, we demonstrate a bit-serial RISC-V microprocessor implementation with no plain-text data in the clear. All values are protected using Boolean masking and differential domino logic. Software can run with little to no countermeasures, reducing code size and performance overheads. Our methodology is fully automated and can be applied to designs of arbitrary size or complexity. We also provide details on other system components such as clock randomizer, memory protection, and random number generator. We report measurements from our 65 nm testchip for the quality of random numbers, and number of traces for key disclosure.

Chapter 2

Background

This chapter covers relevant background on PUFs, including the arbiter PUF, ring oscillator PUF, composite PUFs, challenge obfuscated PUFs, performance metrics, enrollment, authentication, and learning techniques. We also make a brief review of relevant attacks to integrated circuits, including probing, fault injection, and power analysis attacks.

2.1 Physical Unclonable Functions (PUFs)

Physical unclonable functions (PUFs) harvest process variability to produce a unique and unclonable identifier of an integrated circuit. Fabricating two identical PUFs is infeasible even for the original manufacturer, making it a promising weapon to fight counterfeiting. Unlike traditional identification alternatives, two authentic ICs using PUFs will produce distinct outputs, regardless of their programmed memory content. The ideal PUF design is lightweight, secure against modeling attacks, and produces high entropy responses that are chip unique, and stable over various environmental conditions.

PUFs generate chip unique responses based on internal parameters that are hidden from the user. The size of the challenge space determines if a PUF is classified as weak, or strong. Weak PUFs have a small challenge space and were originally conceived for key generation processes. Strong PUFs, however, have a large challenge space, such that exhaustive enumeration is impractical. Therefore, an (ideal) strong PUF may authenticate with an externally accessible, unencrypted, challenge-response protocol.

In this thesis, when clear from context, we may refer to strong PUFs simply as PUFs.

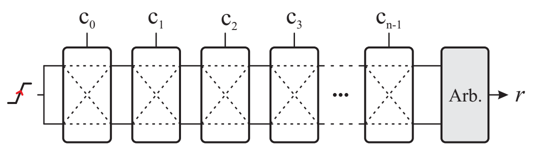


Figure 2.1: Arbiter PUF architecture.

2.1.1 Arbiter PUF (APUF)

The arbiter PUF (APUF) uses manufacture variability in gate delays as source of unclonable randomness [33]. The APUF architecture is shown in Fig. 2.1. The APUF has two signals racing through identically designed delay paths. At the end, an arbiter determines the response based on which signal arrived first. The delay stages are implemented with muxes controlled by the input challenge, which specifies if a particular stage will perform a direct, or twisted connection. Each n-bit challenge creates a unique set of connections that results in two nominally identical delay paths.

The APUF has one main advantage over key based challenge-response systems: the device's memory does not contain secret information. Probing attacks are not able to extract individual gate delays. Exhaustive enumeration attacks are impractical for challenge spaces of 64, or 128 bits. Moreover, since responses are taken from a differential measurement, they are mostly stable under different environmental conditions—reported APUF bit error rates are typically below 5% [87].

The internal parameters of PUFs define the challenge-response relationship. If adversaries can calculate the internal parameters of a PUF, its security is compromised. In the case of APUFs, researchers have demonstrated models trained using only 640 CRPs, that achieve 95% prediction accuracy for unseen challenges [76]. Consequently, the unprotected exposure of APUF challenge-response interface give attackers the resources needed to model its entropy source, and produce counterfeit copies of the target integrated circuit.

2.1.2 Ring Oscillator PUF (RO-PUF)

Similarly to the APUF, the ring oscillator PUF (RO-PUF) uses gate delays as source of unclonable randomness [97]. The RO-PUF architecture is shown in Fig. 2.2. The input challenge selects two oscillators from a pool of oscillators. The response is obtained by comparing their frequency. The number of possible pairings is $n(n-1)/2$, but due to correlated comparisons, the maximum number of responses that can be extracted is $\log(n!)$.

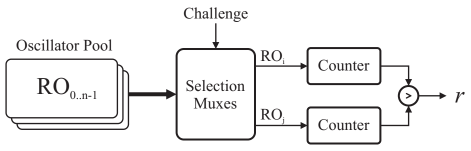


Figure 2.2: Ring oscillator PUF architecture.

Therefore, the RO-PUF as originally proposed in [97], is a weak PUF. Its challenge-response interface is internal, and used only for key generation.

Attempts to increase the challenge space of RO-PUFs were made in [58]. As an alternative to adding more oscillators to the pool, authors propose a low cost identity mapping function that expands the set of CRPs. Instead of pairwise frequency comparisons, the mapping function derives responses from a subset of oscillators. Nevertheless, later works were able to model the architecture using vulnerabilities in the identity mapping function [64].

Chapter 3 presents a new ring oscillator based strong PUF architecture using non-monotonic response quantization, which has an externally available challenge-response interface.

2.1.3 Composite PUF

A number of strong PUF architectures have used compositions to create a strong PUFs with enhanced security properties [52,66,78,80,116]. Similarly to the mathematical notion of a composite functions, PUF compositions use the output of certain instances as input to other instances. The resulting PUF has increased resilience to learning attacks, however, response bit error rate of the overall composition limits the achievable benefits.

Fig. 2.3 shows the PUF-on-PUF (POP) architecture, where the response of 64 APUFs in the first layer is used as challenge to a second layer APUF [116]. Compositions often use PUFs of various sizes to reduce area costs. In the case of Fig. 2.3, APUFs in the first layer have only two stages. Chapter 4 discusses how different choices in size can affect the security properties of the overall composition.

An important characteristic of composite PUFs is that the challenge-response interfaces is externally available. In the case of POP, the internal PUF parameters consist only in variability induced gate delays.

Other PUF architectures instantiate several strong PUFs that are not technically in a composite arrangement, such as the XOR-APUF introduced in [97], and shown in Fig. 2.4.

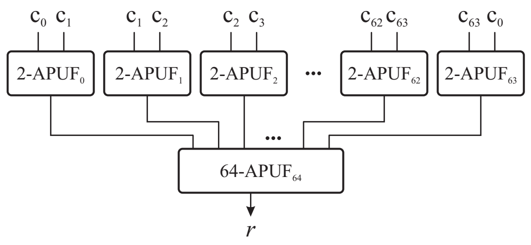


Figure 2.3: PUF-on-PUF composition with 2-stages APUF in the first layer.

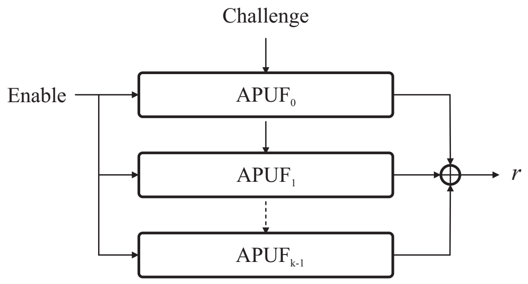


Figure 2.4: k-XOR-APUF composition with k arbiter PUFs.

In the XOR-APUF, the same challenge is evaluated by all instances, and their outputs is XORed to produce the final response. For simplicity, we also refer to such architectures as composite PUFs, or simply compositions.

2.1.4 Challenge Obfuscated PUF

To overcome the response stability problems of PUF compositions, challenge obfuscated architectures perform pre-processing of the input challenge with a secret key. The pre-processing logic is deterministic, and has no impact in the output response stability. Fig. 2.5 shows the multi PUF (MPUF) architecture, where weak PUFs are used to generate secret key bits that are XORed with the input challenge [56].

Challenge obfuscated PUFs seek a *lightweight* solution to make strong PUFs secure against learning attacks. Using an AES engine, for example, would achieve better security properties than the XOR performed by the MPUF, but at a much higher cost. Other proposals of challenge obfuscated PUFs were introduced in [45, 107, 110–112].

A commonly overlooked aspect of challenge obfuscation refers to power analysis attacks. Manipulating an external challenge with a secret key leaks sensitive information in the power consumption [48]. If the secret key is extracted, the strong PUF is exposed to attackers. Chapter 5 introduces a challenge obfuscated strong PUF using non-linear feedback shift registers, where we evaluate the cost with, and without countermeasures for power analysis attacks.

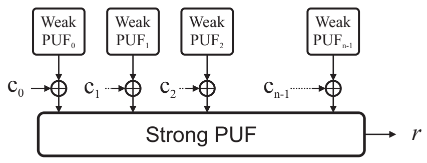


Figure 2.5: Multi PUF architecture.

2.1.5 Performance Metrics

Adequate assessment of strong PUFs requires metrics that evaluate uniformity, uniqueness, and stability of responses.

- *Uniformity:* estimates the ratio of zeros and ones in PUF responses. It is also known as normalized hamming weight. Ideal uniformity is 0.5, which indicates, on average, equal number of zeros and ones.
- *Uniqueness:* estimates the distance between responses from multiple instances. It is also known as normalized hamming distance. Ideal uniqueness is 0.5, which indicates that, for the same set of challenges, on average, half responses will differ.
- *Bit error rate (BER):* estimates reproducibility of responses under several environmental conditions. Bit error rate (BER) reports a ratio of bits (responses) that differ from their enrolled value. BER ideal value is 0%, which indicates no incorrect responses during measurement. Other literature may use the term reliability, which simply denotes (100% - BER).

2.1.6 Enrollment and Authentication

Authenticating a PUF enabled device requires an enrollment phase, which occurs in a secure environment. During enrollment, a randomly selected set of challenges is evaluated and responses are stored in a secure database. When verifying the authenticity of a device, the PUF is inquired with a subset of the enrolled CRPs. Some CRPs are less stable than others, therefore, to successfully authenticate, the number of correct responses must exceed a response threshold, otherwise the authentication fails. The response threshold is a system defined parameter which is set according to PUF response stability. If the threshold is expressed as a percentage of correct responses needed to authenticate, we can

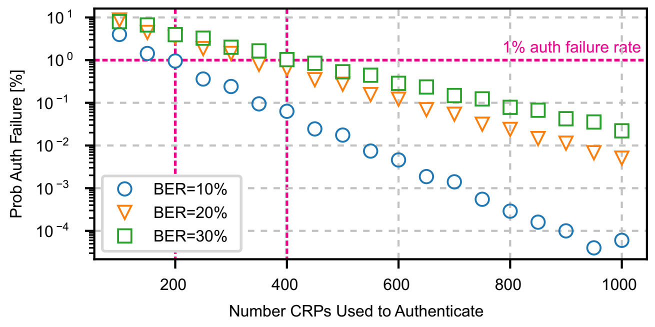


Figure 2.6: Probability of authentication failure simulated with 1M authentications. Uniformity of 50% is assumed. The minimum number of correct responses is 5% below (100% - BER).

argue that it must be less than (100% - BER), otherwise the PUF is unlikely to successfully authenticate.

Different PUF architectures have different bit error rates. From a system perspective, the minimum acceptable bit error rate depends on the number of CRPs used in each authentication, and on the failure rate required by the application. Fig. 2.6 simulates PUFs with different bit error rates. Threshold was set 5% below (100% - BER). For example, when simulating an authentication using 200 CRPs, with a PUF that has 10% BER, 170 correct responses are required to successfully authenticate. As shown in Fig. 2.6, the probability of authentication failure falls exponentially with the number of CRPs used to authenticate. For example, for a 1% failure rate, a PUF with 10% BER will require 200 CRPs, while if the PUF BER is increased to 20% or 30%, the required CRPs to achieve the same failure rate will increase to 350, and 400, respectively. Therefore, as it will be discussed in chapters 3, 4, and 5, applications can trade a moderate increase in BER for enhanced security, given that they can afford to use a larger number of CRPs during authentication.

2.1.7 Learning PUF Behavior

One of the security assumptions of PUFs is that an adversary is unable to predict responses based on past CRPs. Such assumption was demonstrated flawed in [55], and later consolidated in [76], where the authors report 95% prediction accuracy for an APUF, after seeing only 640 CRPs.

Researchers also demonstrated successful attack results by exploring and manipulating helper data for error correction. In particular, error correcting strong PUFs requires chal-

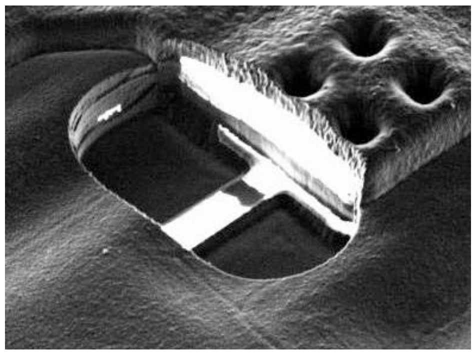


Figure 2.7: Blown polysilicon fuse exposed using focused ion beam (FIB). Source [49].

lenge specific helper data. The intentionally large challenge space makes it impractical to store all the helper data inside the chip, however, providing it externally is not a viable option either, since it was shown vulnerable to learning attacks in [26,95].

Attacks soon evolved to use side-channel stability information from CRPs. In [13], an evolution strategy (ES) technique is used to model an XOR-APUF. The key insight is that, in the APUF, CRPs with small delay difference are more susceptible to noise. Authors showed that increasing the number of APUFs was ineffective at improving learning resilience against their technique.

Recently, deep-neural networks (DNNs) have become a popular choice to model strong PUFs. DNNs require more CRPs and longer training time than typical machine learning techniques, but they are capable of learning complex PUF structures without a precise mathematical model of the target PUF [46].

2.2 Attacks to Integrated Circuits

Secure ICs must endure various types of attacks. This section briefly describes the main attack techniques used to extract secrets from ICs, including probing, fault injection, and power analysis attacks.

2.2.1 Probing Attacks

Probing attacks decapsulate the chip with chemical processes, and then use various techniques to directly access the values of internal wires. The most common probing techniques

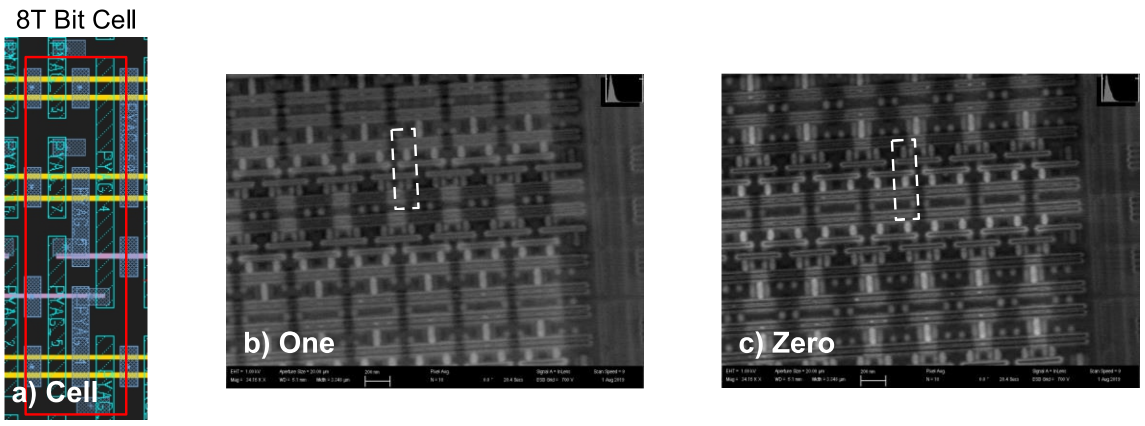


Figure 2.8: Inspecting SRAM content of a FinFet technology using scanning electron microscope (SEM). Source Intel [43].

are electrical, optical, and scanning electron probing.

Electrical probing uses microprobes that physically touch the target wire. Electrical probing is typically combined with a focused ion beam (FIB) equipment, which can remove and deposit material with nanometer precision. Fig. 2.7 shows a blown polysilicon fuse exposed after FIB material removal. The fuse can easily be reconnected using FIB deposition, or a conductive pad can be created for electrical contact with a microprobe. Electrical probing is typically performed through the front (top) of the chip [109].

Optical probing uses the back (bottom) of the chip to capture the photons emitted by transistors during switching. This technique passively detects the received photons to analyze the signal at a particular location. The wavelength of emitted photons is in the infrared, therefore, this technique has a resolution limit of about 900 nm, which limits its applicability [109].

Scanning electron probing uses a focused beam of electrons to measure voltages in metal wires. It is performed through the back side, and requires a reduction in sample thickness to about 10 μm [43]. Fig. 2.8 shows an SRAM array in Intel's FinFet technology. The metal wires for an 8T bit cell are shown in (a). The same 8T cell is highlighted in (b), and (c) with dashed lines, when storing a one and a zero. The arrays show clear difference in voltage contrast between the two states. The bright and dark contrasts correlate with low, and high voltage in the metal wire, respectively [43, 103].

The aforementioned method of reading SRAM memories is applicable to weak PUFs implemented with SRAMs. It must be observed, however, that this technique uses a very expensive and specialized equipment. Moreover, it requires the attacker to decapsulate and thin the IC without affecting its functionality, since the SRAM values are only readable while the circuit is operating.

The use of scanning electron probing is not limited to SRAMs and metal wires. Researchers have shown that with proper sample preparation, the same technique can be used to read the content of flash memories, where logic values are represented by the amount of trapped charges inside a floating gate [23, 121]. The sample preparation process requires exposing the floating gate tunnel oxide from the back side of the chip, therefore, it is a destructive attack. The technique requires a minimum concentration of trapped electrons in the floating gate, which restricts its applicability to older technology nodes. To the best of our knowledge, no published work has replicated the results in nodes smaller than 0.21 μm.

A common countermeasure for probing attacks is to use an active shield. Shields create a dense mesh of metal wires in the top-most metal layer of an IC. Each wire carries a signal generated from a pattern generator and is constantly monitored to detect possible tampering [21]. Chapter 6 discusses other techniques such as Boolean masking, clock randomization, and memory data encryption that also help mitigating the threat of probing attacks.

2.2.2 Fault Injection Attacks

Fault injection attacks actively manipulate a chip during operation to cause a transient error. The goal of injected faults is to temporally disable certain security checks or conditions. For example, if the number of bytes to read from memory is tampered, the device can produce a memory dump. Other possible exploitations include differential fault analysis, where erroneous outputs from cryptographic operations can reveal information about the secret key. A classical example is the RSA algorithm using Chinese remainder theorem. A single flipped internal register during computation causes the exposure of the private key [18]. Similar techniques were also developed for AES and DES algorithms [16, 44].

Typical approaches to inject faults use high intensity lasers, supply glitches, clock glitches, and electromagnetic pulses. Except for electromagnetic pulses, fault injection approaches typically require decapsulating the chip (assuming clock and supply pins are not externally available). Laser faults are often injected from the back of the chip, using wavelength of about 900 nm to benefit from silicon transparency properties [105]. Clock and supply glitches may be combined with FIB and microprobing for better chances of success [9]. For electromagnetic faults, it is possible to use a high voltage spark-gap burst above the surface of the chip [83].

Countermeasures for fault injection include the addition of sensors for voltage and clock glitches, as well as electromagnetic pulses [15,28]. Light sensors are not effective since they can be easily avoided by careful laser positioning [105]. Other techniques to detect fault injection attacks is the addition of redundancy—at all levels. Critical registers may be duplicated, memory can include error correction, data may be read multiple times and checked for consistency, among others [104]. Also, due to the non-deterministic nature of fault injection attacks, countermeasures that manipulate timing may lower attacker's chance of success. The clock randomization countermeasure is discussed in chapter 6.

Strong PUFs are mostly robust against fault injection attacks. Learning attacks that explore challenge reliability, however, may benefit from a higher bit error rates. In [14], authors manipulate the supply voltage to induce erroneous responses, which are then used to train the PUF model more efficiently. Environment temperature may also be subject to manipulation in an attempt to increase PUF bit error rate.

2.2.3 Power Analysis Attacks

It is well known that the power consumption of static CMOS logic gates is dependent on the value of its inputs, but it was a shock for security researchers when the first results on differential power analysis were published [48]. Authors demonstrated a non-invasive key extraction technique that uses nothing more than a set of power consumption traces. The impact of that work was tremendous, creating a large and very active field of research.

Power analysis attacks record power traces of the device manipulating the secret value with different input data. For example, recorded traces may include AES encryptions using the same key, but different plaintexts. The encryption algorithm must be known to the attacker, but implementation details are mostly irrelevant. The attack works by extracting a few key bits in each iteration. For example, the AES key is typically recovered one byte at a time, therefore, each key extraction iteration has 256 key candidates.

To find the correct key among all key candidates, the attacker locates an operation in the algorithm that manipulates both key and input data. A power consumption hypothesis vector is then created for that operation. In other words, for each input data, the attacker computes an approximate power consumption hypothesis for the target operation when using a particular key candidate. Very simple power models work well in most cases. A popular choice is the hamming-weight of the target operation output. The next step is to calculate the correlation between the recorded power traces, and the power consumption hypothesis vector, one key candidate at a time. The correct key candidate will have the highest correlation.

Countermeasures for power analysis attacks work by increasing the number of required power traces such that performing the attack becomes impractical. Most of them fall into two categories: masking and hiding. Masking uses random numbers to hide the value of intermediate computations. It can be applied at the algorithm level, or at the logic level [61]. In this thesis, we focus on Boolean masking, which is applied at the logic level [34]. Hiding techniques, on the other hand, focus on making power consumption independent of the values being computed. Popular examples are differential input/output dynamic logic styles with precharge/evaluation phases [74]. Chapters 5 and 6 discuss the implementation details of Boolean masking, dynamic logic styles, as well as clock randomization, which is another effective countermeasure for power analysis attacks.

An often overlooked property of strong PUFs is their robustness against power analysis attacks. The lack of a secret, in the traditional sense, makes it hard to find correlation between the input challenge, and the PUF entropy source. Previous publications demonstrated power analysis attacks against strong PUFs [1, 77], but as mentioned in [54], the applicability of such techniques is still very limited. One exception must be made to constructions that use weak PUFs, such as challenge obfuscated PUFs, where the obfuscation logic manipulates the input challenge with a secret key, leaking information in the power consumption. In chapter 5, we introduce a challenge obfuscation technique which has countermeasures for power analysis attacks.

Chapter 3

Strong PUFs with Non-monotonic Response Quantization

Many strong PUF architectures, at their core, encode a comparison result of two identical structures. For example, Fig. 3.1 (a) illustrates the popular arbiter PUF (APUF) with two identical delay lines [33]. Each n-bit challenge performs a unique selection of delay elements for the two paths. Depending on which path is faster, the arbiter makes a binary (quantized) decision. The APUF uses *typical response quantization*, where the decision remains the same regardless of the distance between the arriving signals. It's been shown that APUF, and many of its variants, can be modeled using learning algorithms [13,76,102,115]. In this chapter, we propose a technique that innovates by changing the core quantization principle, which is common to most previous strong PUF designs. We introduce the concept of *non-monotonic response quantization* (NMQ) to increase the security of strong PUF architectures. The quantized decision depends not only on which path is faster, but also on the distance between the arriving signals.

A non-monotonically quantized strong PUF (NMQ-PUF) can take various forms. We demonstrate the technique using a ring oscillator based architecture denoted as NMQ-RO, shown in Fig. 3.1 (b). It uses two challenge dependent oscillators. One oscillator is connected to a counter, and the other to a toggling bit. The control logic allows oscillators to run until the counter reaches a predefined value. The response is taken from the toggling bit. The final counter value is chosen such that responses, when plotted along the frequency difference axis, exhibit an alternating pattern of zeros and ones.

Our focus was to increase learning resilience, while keeping bit error rate at a reasonable level. In section 2.1.6, we showed that higher bit error rates can be tolerated using more

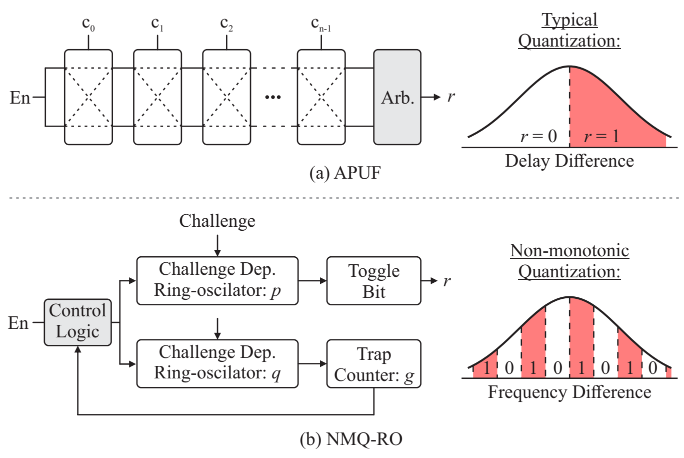


Figure 3.1: Arbiter PUF with typically quantized response in (a); and ring oscillator based PUF with non-monotonically quantized response in (b).

CRPs during authentication, but a fix for an insecure PUF is much harder.

3.1 Related Works

As described in chapter 2, the arbiter PUF (APUF) was the first silicon PUF [33]. PUF architectures have been continuously enhanced to improve resilience against learning attacks. The feed-forward PUF inserts extra arbiters in the delay path to reduce response linearity [52]. Similarly, XOR-APUF used XOR gates to combine multiple APUFs [97], while lightweight PUF added input challenge transformation and a parity based output function [59]. Double arbiter PUF combines two APUFs using a different wiring method and an XOR gate [57]. The interpose PUF uses a composite architecture of two XOR-APUFs to generate a response [66]. Recent advances in PUF architectures require larger CRP datasets, and longer training time, but none was shown secure against learning attacks [13, 46, 76, 102, 115]. Adding architectural, or wiring complexity that always encode the *fastest path* has limited benefits, and may ruin response stability. We innovate by changing the core quantization technique, which is common to all above mentioned PUF designs.

Authors in [33] discuss self-oscillating circuits and frequency comparison as a method for measuring on-chip delay, but no implementation details were presented. In [97], a strong PUF was implemented using a pool of ring oscillators. Responses were obtained by pairwise frequency comparisons, which resulted in a limited challenge space. The architecture was attacked in [76]. Our work does not use pairwise frequency comparisons. We generate responses using non-monotonic quantization instead of directly encoding oscillator performance information.

In [58], authors proposed an RO-PUF with an identity-mapping function to expand the set of CRPs. They also used error correction to reduce bit error rate. The architecture was attacked in [64] using vulnerabilities in the identity mapping function. Moreover, the use of error correction in strong PUFs was shown vulnerable in [26] due to helper data manipulation. We use challenge dependent delay paths as entropy source. Our proposed implementation of NMQ has challenge space of 2^n, where n is 64. Our architecture allows designers to find a compromise between learning resilience and response stability, therefore, we do not employ error correction techniques.

In [20], authors propose a strong PUF based on a bistable ring oscillator (BR-PUF). Such oscillators use an even number of challenge dependent delay stages that act as a large SRAM cell. After releasing the reset, values in each stage will eventually settle to either $01010\ldots01$, or $10101\ldots10$. As stated by the authors, once the ring first enters in a settled state, it does not leave the state again. However, a number of challenges will remain oscillating (unsettled) for a longer, or indefinite, period of time. The authors then define the settling time as an evaluation time limit, which is also used to discard unstable CRPs during enrollment.

3.2 Non-monotonic Quantization

3.2.1 Key Observations

PUFs compare the manufacturing variability of identically designed circuit components. The input challenge specifies the structures to be compared. Responses always encode the comparison result, leaking valuable information about the *best performing* circuit. For example, APUF responses immediately expose which delay path is faster for a certain challenge. Other APUF-based compositions introduce additional instances in an attempt to obfuscate the entropy source information from the final responses, but they have mostly failed to produce learning resistant strong PUFs, since the core quantization technique is still the same. Non-monotonic quantization (NMQ), however, produces outputs that cannot be individually translated into information about the entropy source.

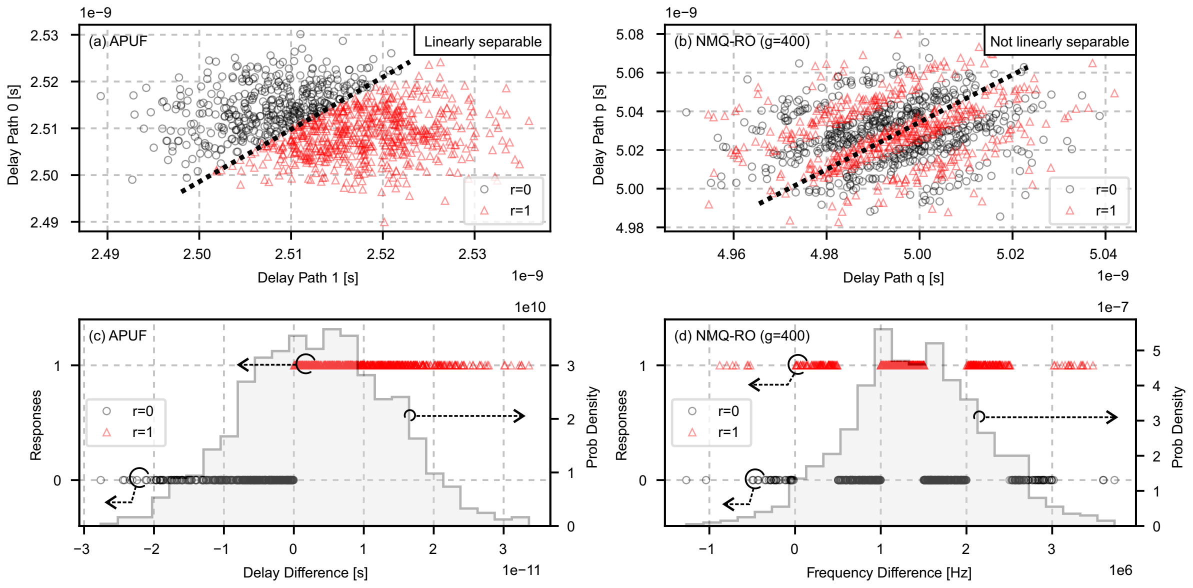


Figure 3.2: SPICE simulated responses for 1000 challenges; (a), and (b) plot APUF, and NMQ-RO (g=400) responses using accumulated path delay as basis, respectively; (c) and (d) plot APUF, and NMQ-RO (g=400) responses along the quantizer input axis, which is delay difference for APUF, and frequency difference for NMQ-RO; (c) and (d) also plot the delay and frequency difference histogram for APUF and NMQ-RO.

Observation 3.2.1 *Typically quantized strong PUFs, at their core, encode the best performing structure. Individual CRPs provide direct information about the entropy source.*

A geometrical perspective of non-monotonic quantization is shown in Fig. 3.2. In (a), APUF responses for 1000 challenges are plotted using accumulated path delay as basis. Similarly, (b) plots NMQ-RO responses. It is clear that the typically quantized responses in (a) are separable by a line. The NMQ responses in (b) however, cannot be linearly separable on the chosen basis.

Observation 3.2.2 *Non-monotonically quantized strong PUF responses are not linearly separable on the basis of accumulated path delay.*

Based on Obs. 3.2.2, we may argue that linear models, such as logistic regression (LR), are not able to model NMQ-PUFs. While APUFs require composition to avoid LR attacks, our NMQ-PUF implementation is shown resistant against LR, even when responses are taken from a single instance. Detailed experimental data is presented in section 3.6.

Fig. 3.2 (c) and (d) plot APUF and NMQ-RO responses along the delay, and frequency difference axis, respectively. While typically quantized responses from the APUF encode the delay difference sign, NMQ-RO responses show an alternating pattern of zeros and ones. The delay and frequency difference histograms are plotted above responses in (c) and (d), for APUF, and NMQ-RO. Unlike APUF, the NMQ-RO histogram is not necessarily centered at zero. NMQ-RO responses are defined by the distance between the two frequencies, not the sign of their difference.

3.3 NMQ-PUF Using Ring Oscillators

A non-monotonically quantized strong PUF (NMQ-PUF) can take various forms. We demonstrate a ring oscillator based strong PUF, denoted NMQ-RO, which is shown in Fig. 3.1. It uses two identical, challenge dependent ring oscillator structures. One ring oscillator is connected to a counter named *trap counter*, and the other to a *toggling bit*. The toggling bit is complemented, and the trap counter is incremented, at every rising edge of their associated oscillating signal. The control logic disables both ring oscillators when the trap counter reaches a predetermined value. The final response is taken from the toggling bit.

The response, r, may be written in terms of circuit variables as

$$r = \text{LSB}\left(\left\lfloor g\frac{D_p(\mathbf{c})}{D_q(\mathbf{c})}\right\rfloor\right), \tag{3.1}$$

where $\text{D}_{\{p,q\}}(\mathbf{c})$ are the challenge dependent propagation delays of ring oscillator p, and q. The term g denotes the pre-determined trap counter final value. The function LSB() returns the least significant bit.

According to Eq. 3.1, NMQ-RO responses encode the performance ratio of two ring oscillators into a single bit value that can only represent $\{0, 1\}$. Non-monotonic quantization arises due to information lost during the encoding of $D_p(\mathbf{c})/D_q(\mathbf{c})$ into the single bit response r. For example, if

$$g\frac{D_p(\mathbf{c})}{D_q(\mathbf{c})} \geq 2,$$

the performance ratio information, for that particular challenge, is no longer directly encoded in the response, since only the least significant bit of the above product is captured. From a design perspective, we can promote information loss by increasing the trap counter

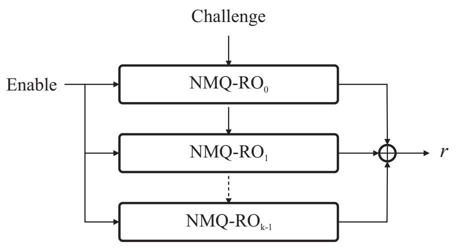


Figure 3.3: XOR-NMQ-RO composition, the same challenge is applied to all instances.

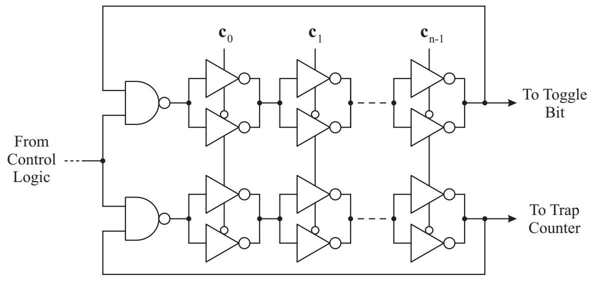


Figure 3.4: CMOS implementation of the NMQ-RO challenge dependent ring oscillators.

final value, g. The parameter g must be selected such that information loss occurs for a significant number of challenges.

Large g values, however, increase bit error rate. Our measurement results and security assessment, sections 3.5 and 3.6, indicate that large g enhances learning resilience, but impacts response stability. Designers should explore the trade-off between security and response stability, choosing the parameter g accordingly.

3.4 Compositions of NMQ-PUFs

The security of NMQ-RO is primarily controlled by the chosen g value. However, our measurement data reveals that g sets a trade off between learning attack resilience and response stability. As shown in section 3.6, improving security solely by increasing g can have a significant impact on bit error rate.

In order to find a compromise between security and response stability, we explore PUF compositions. Researchers have demonstrated that compositions enhance APUF resilience against learning attacks [76]. Therefore, we apply similar methods to NMQ-RO, but instead of using typically quantized PUFs, we use NMQ-RO with a moderate final trap counter value (g=200), such that bit error rate for the overall composition is less than 20%.

We demonstrate a composite strong PUF using three NMQ-ROs. The k-XOR-NMQ-RO is shown in Fig. 3.3. Term k refers to the number of NMQ-RO instances used. Similarly to a k-XOR-APUF, all instances evaluate the same challenge. Outputs are XORed to produce the final response.

Other PUF compositions could also increase the resilience to learning attacks, while maintaining response stability at an acceptable level. We chose the XOR-APUF style of

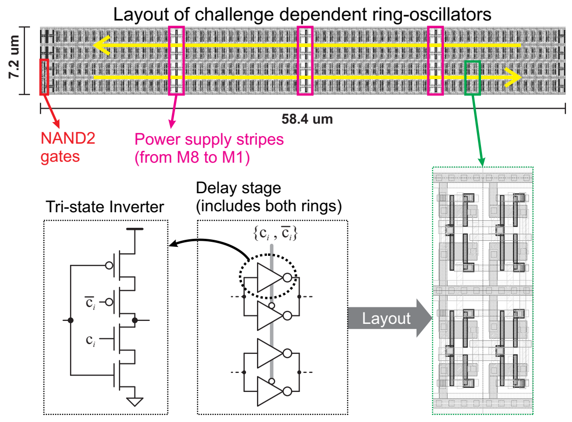

Layout of challenge dependent ring-oscillators
7.2 um
58.4 um
NAND2 gates
Power supply stripes (from M8 to M1)
Tri-state Inverter
Delay stage (includes both rings)
$\{c_i, \overline{c}_i\}$
$\overline{c}_i$
c_i
Layout

Figure 3.5: Layout and schematic of challenge dependent ring oscillator.

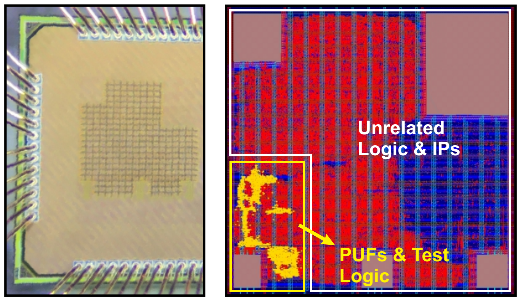


Figure 3.6: Die photo of fabricated chip in 65 nm CMOS process.

composition because it is well understood, with mature attack techniques.

3.5 Testchip Design and Measurement Results

We designed a testchip to assess strong PUF performance metrics, such as uniformity, uniqueness, and bit error rate (BER). Our objective is to evaluate the impact of g in response stability, which will provide essential data to explore the security/stability trade-off in section 3.6.

3.5.1 Testchip Design

We designed a testchip in 65 nm CMOS technology. Our testchip includes 10 instances of NMQ-RO, with $n = 64$. The XOR-NMQ-RO composition is realized through post processing responses from these instances.

The CMOS implementation of NMQ-RO uses custom designed ring oscillators with an even number of challenge enabled tri-state inverters, and a NAND gate for control, as shown in Fig. 3.4. Layout of ring oscillators, and the schematic for tri-state inverters are shown in Fig. 3.5. No special circuit technique was used to compensate for temperature or noise.

We used automated synthesis, placement, and routing tools to design the trap counter and surrounding test logic, which are shown in Fig. 3.6, highlighted in yellow. An addi-

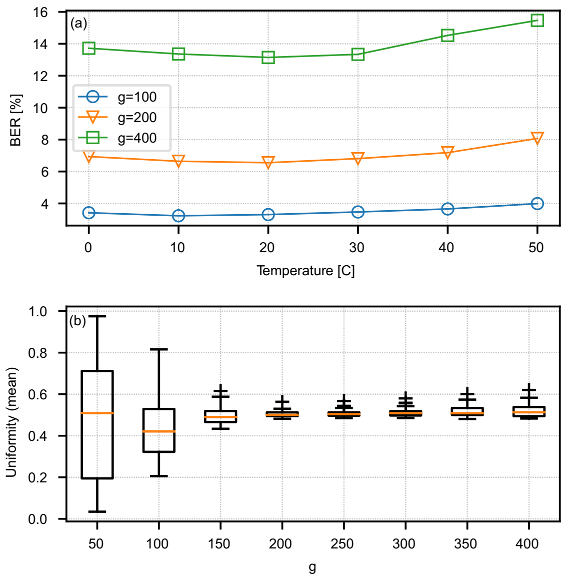


Figure 3.7: NMQ-RO (a) average bit error rate, and (b) distribution of instance uniformity, for multiple values of g, using 10 k CRPs.

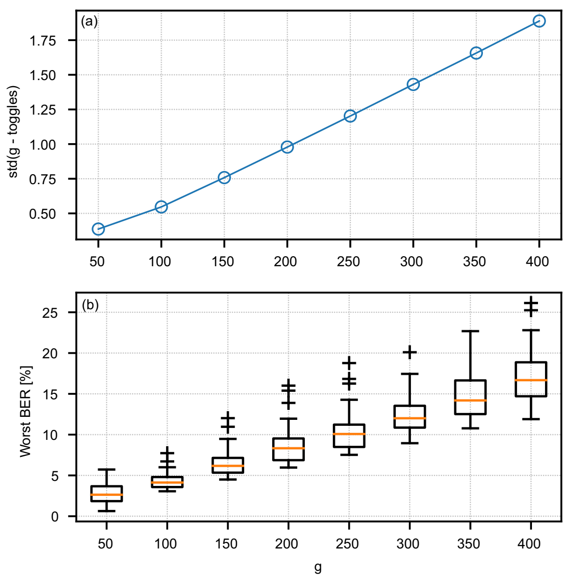


Figure 3.8: NMQ-RO (a) standard deviation of trap counter final value minus the number of toggles; and (b) worst bit error rate.

tional test counter was created to keep track of the total number of toggles, which is only used for experimental reasons.

3.5.2 Measurement Results

Measurements were performed on a total of 60 NMQ-RO instances, spread over 6 dies. Each die also included one APUF instance. We performed enrollment at 20 °C with a single evaluation (no temporal majority voting). To calculate bit error rate, CRPs are evaluated 100 times at each temperature from 0 °C to 50 °C. The reported metrics were calculated over all 60 instances of NMQ-RO.

Bit error rate measures the stability of PUF responses. In the case of NMQ-RO, it depends on the parameter g. Fig. 3.7 (a) shows bit error rate for different values of g, measured using 10 k CRPs. At 20 °C, we measured BER of 3.3%, 6.5%, and 13.1%, for g=100, g=200, and g=400, respectively. Other temperatures in the 0 °C – 50 °C range show a small BER variation compared to their respective enrollment temperature.

Fig. 3.7 (b) depicts the uniformity distribution (mean values) for all instances with respect to g values. Uniformity was calculated using 10 k CRPs. The larger uniformity

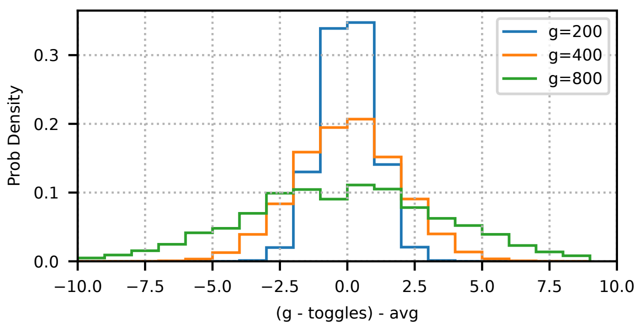


Figure 3.9: NMQ-RO histogram of trap counter final value minus number of toggles for various g, using 100 k challenges. The average value was subtracted to center histograms at zero.

spread observed when g is less than 200 is caused by performance mismatches between the two ring oscillators. Therefore, when NMQ-RO runs with small g values, a significant number of challenges will not accumulate a sufficiently large delay difference to move $\lfloor gD_p/D_q \rfloor$ away from its *average* value, leading to biased responses. The use of twisted ring oscillators could solve this issue, unlocking configurations where g is less than 200. Designers must be careful, however, with possible frequency locking between the two ring oscillators due to increased capacitive coupling.

The standard deviation of trap counter final value minus number of toggles is shown in Fig. 3.8 (a), for a range of g values, using 10 k CRPs. As g increases, more cycles become available to accumulate the delay difference between the two ROs, which leads to larger standard deviation values. Fig. 3.8 (b) plots the worst BER distribution for all instances, at the respective g, also using 10 k CRPs. It shows that bit error rate also increases for larger values of g. Another perspective on the results is shown in Fig. 3.9, where the same data is plotted in a histogram format. The average value of each distribution was subtracted to center them at zero.

The uniformity and uniqueness histograms were calculated using 1M CRPs per instance, and exhibits near ideal behavior. The uniformity histogram, for all instances combined, is shown in Fig. 3.10 (a). Uniformity mean is 0.505 with 0.093 standard deviation, which represents that, on average, 50.5% of responses are 0, and 49.5% are 1. The uniqueness was calculated with multi-bit responses of 32 bits. The uniqueness histogram is shown in Fig. 3.10 (b). Uniqueness mean is 0.500 with 0.003 standard deviation, which indicates that, on average, 50.0% of response bits were different in every group of 32 responses, for the same challenges, across multiple instances.

Our results report data for a total of 18 instances of 3-XOR-NMQ-RO, and 30 instances

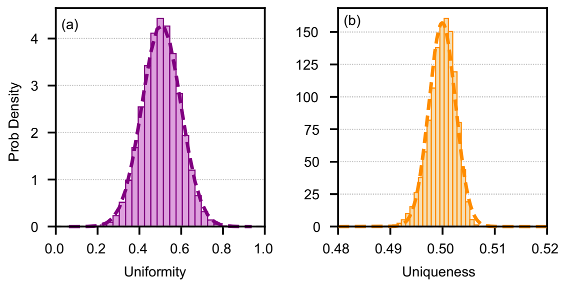


Figure 3.10: NMQ-RO (a) uniformity for 1 M CRPs with all instances combined, and (b) uniqueness for 1 M CRPs across all instances (g=200).

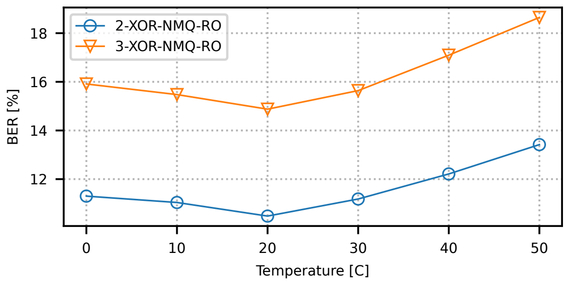


Figure 3.11: Bit error rate of 2-XOR-NMQ-RO, and 3-XOR-NMQ-RO compositions, using 5 k CRPs.

of 2-XOR-NMQ-RO. Fig. 3.11 shows mean BER calculated using 5 k challenges, from 0 °C to 50 °C. The lowest BER occurs at the enrollment temperature of 20 °C, while the peak bit error rate is found at 50 °C. The worst BER for 2-XOR-NMQ-RO, and 3-XOR-NMQ-RO is 13.4%, and 18.6%, respectively.

Better response stability is possible via the utilization of specialized circuit techniques, or different RO designs [2, 6, 8]. They have potential to improve stability of NMQ-RO responses.

3.6 Security Assessment

The goal of learning attacks is to find a representation of the PUF entropy source, such that it can be used to accurately predict responses to unseen challenges. In this section, we first develop the concept of uniqueness sensitivity to entropy source, showing how uniqueness sensitivity can provide rapid insight on PUF resilience to learning attacks. Next, we report extensive experimental results with various learning attack techniques.

3.6.1 Uniqueness Sensitivity to Entropy Source

We investigate a technique to gain rapid insight on learning resilience of strong PUFs by exploring the uniqueness metric, and its interaction with PUF entropy source. Experiments using this technique take less than one hour to complete, without any specialized hardware accelerators. This technique does not replace comprehensive attack experiments,

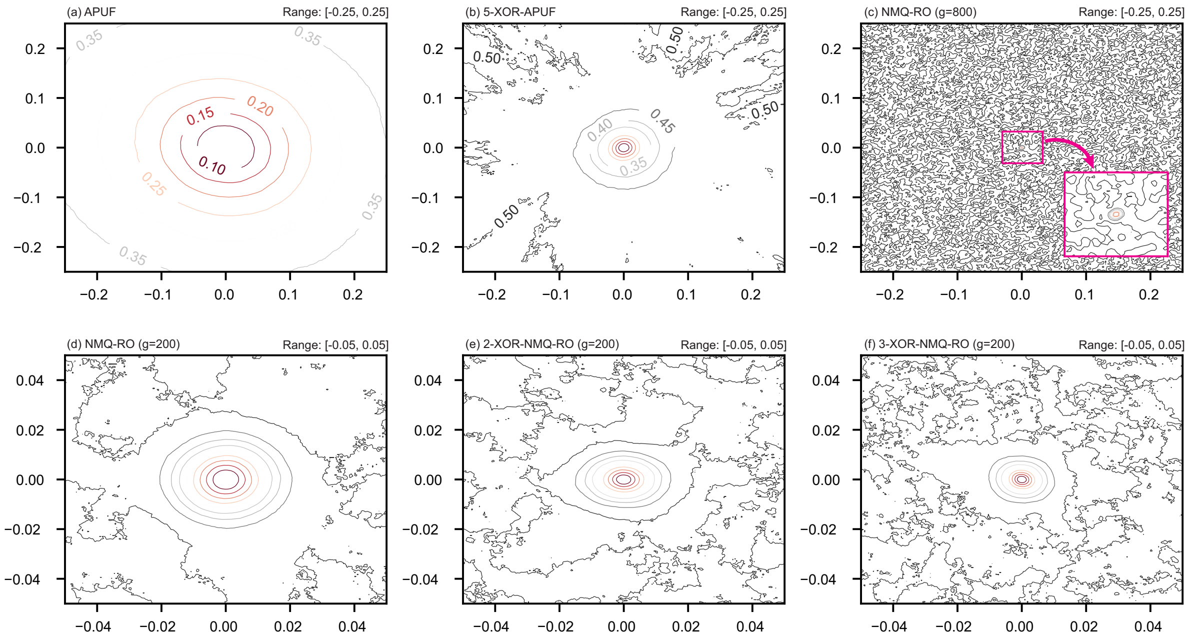


Figure 3.12: Contour plots of uniqueness surface exploration over entropy source space; (a) APUF, (b) 5-XOR-APUF; (c) NMQ-RO (g=800); (d) NMQ-RO (g=200); (e) 2-XOR-NMQ-RO (g=200); (f) 3-XOR-NMQ-RO (g=200).

but instead, offers valuable insight during design stage, allowing fast iteration between different PUF architectures, and design parameters. The code for computing uniqueness sensitivity is publicly available [92].

The entropy source of PUFs originates at random deviations in device parameters that occur during IC manufacturing. Small deviations in the entropy source of a strong PUF should produce unique responses. If $\boldsymbol{\theta}$ denotes entropy source parameters, the uniqueness of two different PUF instances is given by $U(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1)$.

Definition 3.6.1 *Uniqueness sensitivity is the rate of change in uniqueness with respect to changes in the entropy source.*

The high-dimensional space of PUF parameters makes practical use of uniqueness sensitivity non-trivial. To visualize the impact of small entropy source changes in uniqueness, we use a curvature exploration method, described in [53]. We pick two random direction vectors $\boldsymbol{\delta}$ and $\boldsymbol{\eta}$, of same dimension as $\boldsymbol{\theta}$, and evaluate

$$f(\alpha, \beta) = U(\boldsymbol{\theta}_0, \boldsymbol{\theta}_0 + \alpha\boldsymbol{\delta} + \beta\boldsymbol{\eta}). \tag{3.2}$$

Parameters (α, β) are scalars, which are swept as input coordinates to calculate uniqueness over a set of challenges. When $(\alpha = 0, \beta = 0)$, uniqueness will be evaluated for two identical instances (same entropy source), resulting in zero. As (α, β) move away from the origin, the distance to the original entropy source, $\boldsymbol{\theta}_0$, will increase, hence the uniqueness between the original instance, and the new instance parameterized by $\boldsymbol{\theta}_0 + \alpha\boldsymbol{\delta} + \beta\boldsymbol{\eta}$, will diverge from 0 and center around 0.5.

Fig. 3.12 shows contour plots for uniqueness in different PUF architectures. In (a), (b), and (c) the input coordinate range to Eq. 3.2 is $\alpha \in [-0.25, 0.25]$, and $\beta \in [-0.25, 0.25]$. In (a), the APUF uniqueness is zero at $(0, 0)$, and gradually increases as α and β move away from the origin. Similar behavior is observed in (b), where the 5-XOR-APUF uniqueness is zero at $(0, 0)$, and increases as we move away from the origin. However, the 5-XOR-APUF shows a much steeper uniqueness increase, with the ideal contours of 0.5 appearing closer to the origin.

The uniqueness contour plots for NMQ-RO and XOR-NMQ-RO are plotted in Fig. 3.12 (c), (d), (e), and (f). In (c), the NMQ-RO (g=800) is plotted with the same range of input coordinates as (a), and (b). Uniqueness sensitivity is such that contours with values smaller than 0.5 are barely visible, requiring a zoomed graph inset.

For plots in Fig. 3.12 (d), (e), and (f), the input coordinate range is 5x narrower, $\alpha \in [-0.05, 0.05]$, and $\beta \in [-0.05, 0.05]$. The contour plots for NMQ-RO (g=200) is shown in (d). Extra instances of NMQ-RO (g=200) are XORed to form 2-XOR-NMQ-RO in (e) and 3-XOR-NMQ-RO in (f). We observed that XORing additional NMQ-RO instances caused a small increase in uniqueness sensitivity, but removed noticeable patterns in the contour lines near the origin.

Hypothesis 3.6.1 *High uniqueness sensitivity to entropy source deviations is an indicator of strong PUF robustness to learning attacks.*

Typically, learning attacks use differentiable neural network architectures trained by stochastic, gradient descent algorithms. Those optimization algorithms search for weights and biases which *minimize* a loss function between the collected CRP database, and network predictions. If the learned network parameters are a representation of the strong PUF entropy source, we can argue that the contour plots of uniqueness over the entropy source space, shown in Fig. 3.12, capture the impact of a particular PUF architecture on the loss function used for training. Therefore, our results suggest that strong PUFs with high uniqueness sensitivity will likely produce harder to optimize loss functions.

3.6.2 Learning Attack Results

We performed experiments to assess the security of NMQ-PUFs. We used CRP databases generated from a high-level model, and from our silicon implementation. The high-level model, the attack code, and the CRP database collected from silicon are publicly available [92, 94]. Our findings are summarized in Table 3.1. The *Quant.* column defines the quantization method used, either non-monotonic, or typical. The k column is the number of XORed instances. The *BER* represents the worst case measured BER over the temperature range from 0 °C to 50 °C. The *Database Type* column specifies the origin of CRPs used in the attack—*Model* refers to the high-level model, while *Silicon* refers to CRPs from our testchip. If both database types are specified, it indicates the experiment was run twice, once with each database, obtaining very similar results for both runs.

The learning resilience of NMQ-PUFs was first assessed using LR. We used 1 M CRPs to train an LR model with NMQ-RO responses using g=200. As shown in Table 3.1, the modeling accuracy obtained was 50.1%, which supports Obs. 3.2.2, suggesting that NMQ-PUF responses cannot be learned with linear models.

Next, we tried the attack in [13], where the authors implemented a covariance matrix adaptation (CMA-ES) algorithm that uses response stability as side-channel information. The key insight is that, in typically quantized PUFs, CRPs with small delay difference are more susceptible to noise. But CMA-ES assumptions on noisy CRPs do not hold for NMQ-PUF, since small performance differences are not correlated to response stability. This is confirmed by the low CMA-ES accuracy against NMQ-RO (g=200), of 49.8% (shown in Table 3.1).

Attack techniques that rely on a limited challenge space are unlikely to model NMQ-RO responses. As mentioned before, NMQ-RO uses challenge dependent ROs, which may be seen as a pool of 2^{64} different ROs. For example, in [30], the polynomial-size decision list cannot represent our architecture. Another similar attack using Fourier analysis was introduced in [31], and as shown in Table 3.1, it was not able to obtain generalized learning of NMQ-RO.

Deep-neural networks (DNNs) are emerging as an efficient attack technique capable of learning complex PUF structures. Our DNN attack makes no assumption on the underlying architecture, and its code is publicly available [92]. We use a 12-layer DNN architecture

Table 3.1: Learning attack results for non-monotonically quantized PUFs.

	Quant.	**k**	**BER**	**CRPs**	**Database Type**	**Attack**	**Accur.**	**Time**
NMQ-RO (g=200)	NMQ	1	8%	1 M	Model & Silicon	LR	50.1%	10 s
				1 M	Model & Silicon	CMA-ES	49.8%	10 h
				1 M	Model & Silicon	Fourier	50.0%	1 h
				1 M	Model & Silicon	DNN	95.1%	12 h
NMQ-RO (g=400)	NMQ	1	15%	1 M	Model & Silicon	DNN	90.0%	12 h
				5 M	Model & Silicon	DNN	91.4%	2.5 days
NMQ-RO (g=800)	NMQ	1	29.4%	1 M	Model & Silicon	DNN	75.0%	12 h
				5 M	Model & Silicon	DNN	86.5%	2.5 days
NMQ-RO (g=5000)	NMQ	1	*	50 M	Model	DNN	50.7%	4 days
XOR-NMQ-RO (g=200)	NMQ	2	13.4%	20 M	Model	DNN	50.3%	7 days
				10 M	Silicon	DNN	49.6%	3.5 days
		3	18.6%	20 M	Model	DNN	49.9%	7 days
				10 M	Silicon	DNN	50.2%	3.5 days

Notes: BER is the worst bit error rate from 0 °C to 50 °C, with challenges enrolled at 20 °C.

proposed in [46] for all our DNN attacks. The input and output layers have 64, and 2 units, respectively. Hidden layers have 2000 units. Table 3.1 shows that DNN attacks are capable of learning NMQ-ROs using g=200, g=400, and g=800. The trade-off between final trap counter value and stability is present, where larger g values increase learning resilience, but decrease response stability. An exploratory DNN attack was performed with g=5000. The DNN model was not able to model NMQ-RO using g=5000, which shows that increasing g will, eventually, make the PUF resistant to this attack—aside from the significant reduction in response stability that makes such configuration impractical.

To find a compromise between learning resilience and response stability, we explored compositions that use NMQ-RO. Since compositions tend to decrease response stability, we use NMQ-RO at a moderate g value (g=200). Results in Table 3.1 show the DNN attack was not able to obtain generalized learning of XOR-NMQ-RO. Moreover, both implementations, with 2, and 3 instances, show acceptable BER of 13.4% and 18.6%, while preserving the PUF entropy source parameters from DNN attacks using up to 20 M CRPs.

3.6.3 Comparison with Prior Works

A comparison of 2-XOR-NMQ-RO with other strong PUF designs is shown in Table 3.2. In addition to the traditional metrics of area, bit rate, power, and energy, we highlight two very important criteria, bit error rate, and security testing.

As discussed in section 2.1.6, the requirements for bit error rate are application dependent. We make one important remark, however, on the discarding of unstable CRPs during enrollment. It is a popular technique to selectively enroll only stable CRPs, but designers must be conscious that as a side effect, the verifier will have to transmit all CRPs, bit by bit, during the authentication process. This is unlike solutions without selective enrollment, where CRPs can be locally unrolled using a linear feedback shift register (LFSR), from a single input challenge.

3.7 Conclusion

We introduced a non-monotonic quantization technique to increase security of strong PUF architectures. We implemented a non-monotonically quantized RO-based strong PUF in 65 nm CMOS technology. Our experiments show that the resulting PUF has increased security against learning attacks. Measurement results also show the proposed PUF has less than 13.4% BER over a temperature range of 0 °C to 50 °C. Moreover, we introduced the concept of uniqueness sensitivity to entropy source, showing how it can provide rapid insight on PUF resilience to learning attacks.

Table 3.2: Comparison with recent strong PUF prototypes.

	This work (2-XOR-NMQ-RO)	**TCAS I'22** [122]	**TCAS I'22** [50]	**TCAS I'20** [124]	**TCAS II'20** [108]	**ISSCC'15** [119]
Technology	65 nm	65 nm	28 nm	130 nm	65 nm	40 nm
Type	RO-based	Amp. chain	Amp. chain	Cur. array	Volt. array	Delay
Chal. space	1.8×10^{19}	1.2×10^{21}	3.7×10^{19}	3.7×10^{19}	1.15×10^{18}	5.5×10^{28}
# BER test	30	Unclear	Unclear	Unclear	Unclear	Unclear
BER (native)	13.4%	5.7%	10.4%	9%	10.9%	9.1%
BER (discard)	–	–	–	0.4%	–	0%
Discarded CRPs	–	–	–	42%	–	34%
Uniqueness	50.0%	49.9%	50.9%	49.9%	50.26%	50.1%
Temp range	0, 50 °C	-30, 125 °C	-40, 100 °C	-20, 80 °C	0, 50 °C	-25, 125 °C
Area (μm^2)	1178	6845	188	44700	18700	845
Bit rate (Mb/s)	1.4	20	2	0.006	12.7	1.6
Power (μW)	237	375	0.441	0.068	3.8	28.4
Energy (pJ/bit)	165	18.75	0.22	11	0.3	17.75
Security test	DNN/others	LR/ES/MLP	LR/SVM/MLP	LR/SVM/MLP	LR/SVM/MLP	–
# CRPs used	20 M	1 M	20 k	10 k	8 k	–
Model Avail.	Yes [92]	No	No	No	No	No
CRPs Avail.	Yes [94]	No	No	No	No	No

Chapter 4

Design Exploration of Composite Strong PUF Implementations

In this chapter, we study the effect of composite architectures in the learning resilience and response stability of PUFs. In particular, we designed and implemented a testchip in 65 nm technology with several variations of the PUF-on-PUF (POP) architecture, introduced by Wu *et al* [116]. POP uses a two layers construction, shown in Fig. 4.1, where responses from first layer serve as input to the second layer. Our implementation also adds new features to POP, including support for multiple first round evaluations, and temporal majority voting (TMV) for noise removal. We use APUFs as basic building block. Our testchip includes different implementations of the first layer using using APUF of 2, 4, 6, 8, 12, and 24 stages.

To assess the security of our POP implementations, we performed extensive learning attacks varying depth (rounds) and size (number of stages in first layer APUFs). We report experiments where attacks using deep neural networks (DNNs) achieve low prediction scores when the first layer uses APUFs of 6, 8, 12, and 24 stages. We also found counter-intuitive results for learning resilience. POP implementations using 2, and 4 stage APUFs in the first layer are shown vulnerable to DNN attacks. Moreover, increasing the number of rounds brought no improvements against DNN attacks, in fact, it made small sized POP implementations more vulnerable. We reflect on such results, extending previous techniques of influential bit analysis to assess stage bias in APUF instances [29]. To shed light on why depth increase is ineffective to thwart learning attacks, we show that the hamming-distance of first layer responses decreases as the number of rounds increases. Therefore, small APUFs in the first layer limit the challenge space of the second layer

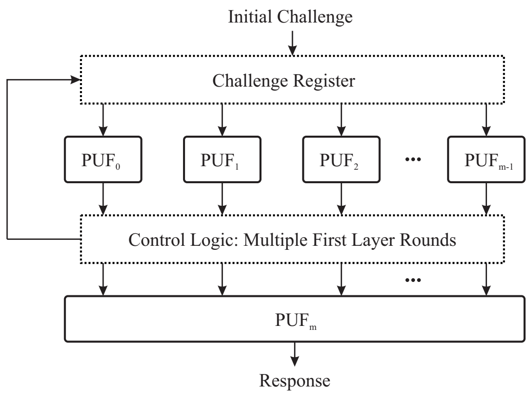


Figure 4.1: Composite strong PUF architecture supporting multiple first round evaluations. The design extends the PUF-on-PUF (POP) concept, proposed in [116].

APUF, showing that compositions not always preserve security properties of PUFs—the size of composing PUFs plays a crucial role.

4.1 Related Works

PUF architectures have been continuously enhanced to improve resilience against learning attacks [52, 57, 59, 78, 97]. Other works have tried to design PUFs with non-linear challenge-response relationship by operating in subthreshold regime [50, 124], or using amplifier chains [122]. Such solutions were effective in improving resilience to classical machine learning approaches [76], but none was shown resistant to recent attacks using deep neural networks [46, 82].

Composite architectures use the output of PUFs as input of other PUFs. They were initially introduced in [80]. Later, researchers also proposed combining weak and strong PUFs [56]. The concept of composite architectures was used in the interpose PUF, where multiple XOR-APUF instances form a composition with improved resilience to learning attacks [66]. Composite constructions require larger CRP datasets, and longer training time, but still, security against DNN attacks remains an open problem [46,82,102,115,116].

The work by Wu *et al* [116] highlights the vulnerabilities of prior composite PUFs against cryptanalysis attacks [79], and introduces a new architecture which is resistant against such attacks, denoted as PUF-on-PUF (POP). In this chapter, we explore the design space of POP, implementing a testchip in 65 nm CMOS process, and performing an extensive security assessment on various POP implementations.

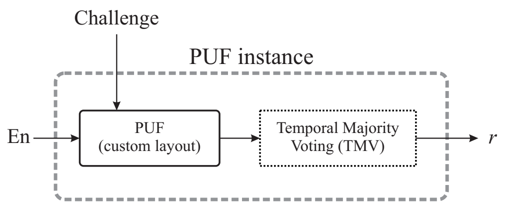


Figure 4.2: Temporal majority voting (TMV) is implemented in each individual PUF instance. It evaluates the PUF multiple times, returning the most frequent response.

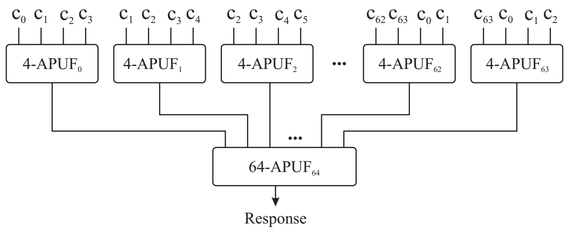


Figure 4.3: Input challenge wiring in first layer for APUFs of size 4. Other sizes follow similar pattern, where challenge bits are applied to multiple PUF instances.

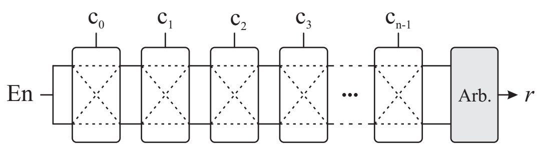
4.2 Notation

An arbiter PUF (APUF) with an n-bit challenge is denoted as an APUF of size n, or n-APUF. An n-APUF-POP refers to a POP implementation where all APUFs in first layer have n stages. Vectors are written in bold text, and are indexed from zero, for example, $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$. The hamming weight and hamming distance functions are denoted as HW(), and HD(), respectively. The *narrow*, and *wide* temperature sets refer to {0 °C, 20 °C, 60 °C}, and {-30 °C, 0 °C, 20 °C, 60 °C, 80 °C}.

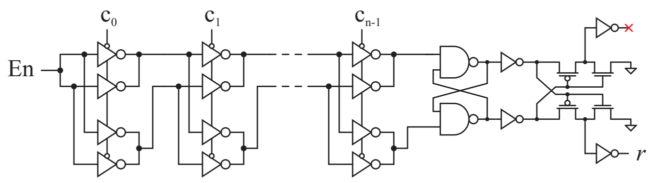
4.3 The PUF-on-PUF Architecture

The PUF-on-PUF (POP) architecture was proposed by Wu *et al* [116]. It uses composition as an alternative to increase strong PUF resilience to learning attacks. Our implementation of POP is shown in Fig. 4.1. It uses a two layers construction, where responses from the first layer serve as input to the second layer. Our implementation adds support for multiple first round evaluations as a low-cost alternative to increasing the number of layers. In other words, first layer responses can be reused as input challenge for additional evaluation rounds, prior to the final second layer evaluation.

Our implementation of POP uses APUFs as building block, for its simplicity, stability, and well understood security characteristics. The input challenge has 64 bits, and the number of APUF instances in the first layer matches the number of challenge bits. The first layer can be implemented with APUFs of any size, while the second layer APUF must match the number of stages with the number of APUF instances in the previous layer. In this chapter, we restrict ourselves to first layer implementations where all APUFs have the



(a) APUF block diagram



(b) APUF schematic

Figure 4.4: Arbiter PUF block diagram in (a), and implemented schematic using tri-state inverters and NAND-based arbiter in (b).

same size.

Each APUF instance uses temporal majority voting (TMV) to filter out noise, as shown in Fig. 4.2. TMV performs a predetermined number of repeated evaluations, returning the most frequent response. It is important to notice that TMV is not applied to the overall composition, but to each individual APUF instance.

As described by Wu *et al*, careful wiring of challenge bits in the first layer is required to avoid cryptanalysis attacks [116]. The wiring pattern must use each challenge bit in more than a single PUF instance. The POP wiring for a first layer implementation using 4-APUF is shown in Fig. 4.3. Each 4-APUF$_i$ is connected to the input challenge at offset i. If the sum of offset and APUF size is greater than 63, the challenge bits simply wrap around. When performing evaluations with multiple rounds, the challenge register is re-loaded with responses from APUFs in the first layer. Responses of each APUF$_i$ are used as challenge bit i for first layer re-evaluation. Similarly, second layer evaluations are performed by wiring responses of each APUF$_i$ in the first layer to the stage i of the second layer APUF.

4.4 Testchip Design

High-level models offer a convenient alternative to test the security of new PUF architectures. The delay of each APUF stage follows a well understood normal distribution. When noise is not considered, PUF responses from high-level models tend to be indistin-

Table 4.1: Area cost of each APUF size.

	# Stages	**Height (μm)**	**Width (μm)**	**Area (μm^2)**	**Norm. (ND2)**
2-APUF	2	1.8	11.8	21.2	14.8
4-APUF	4	1.8	18.2	32.8	22.8
6-APUF	6	1.8	24.6	44.3	30.8
8-APUF	8	1.8	31	55.8	38.8
12-APUF	12	1.8	43.8	78.8	54.8
24-APUF	24	1.8	82.2	148.0	102.8
64-APUF	64	1.8	210.2	378.4	262.8

guishable from responses obtained from silicon. This allows designers to perform an early assessment of uniformity, uniqueness, and resilience to learning attacks. However, response stability is a key performance metric which cannot be accurately estimated without silicon implementation. For this reason, we designed and implemented a testchip in 65 nm CMOS technology. Our testchip is used to evaluate the response uniformity, uniqueness, and bit error rate of our POP implementation.

We use APUF as building block. Our APUF instances are designed with tri-state inverters and a NAND-based arbiter, as shown in Fig. 4.4. Layout of APUFs is custom-made to ensure identical routing of both delay paths. The layout of an APUF with 2 stages is shown in Fig. 4.5. We designed APUFs with 2, 4, 6, 8, 12, 24, and 64 stages. Area information for each APUF is shown in Table 4.1. Height and width dimensions are listed in μm, while area is provided in μm^2, and normalized by the NAND2 area. Our APUF cells have the same height as logic gates from the commercial standard-cell library, which allows automatic placement and routing by EDA tools. This methodology significantly reduces design effort, without loosing the performance of a custom approach.

Die photo and layout are shown in Fig. 4.6. The cells highlighted in yellow implement a JTAG interface and test logic. APUF instances used in the first layer are highlighted in blue, they account for 64 instances of each APUF cell size, including 2, 4, 6, 8, 12, and 24 stages. In total, 384 APUFs are instantiated to construct 6 different first layer implementations. The PUF with 64 stages, used in the second layer, is highlighted in red. Cells highlighted in green implement the round control logic and TMV counters. The TMV logic is largely oversized for exploratory reasons, using a total of 65 counters, each with 24-bits. Results reported in section 4.5, show that more than 15 TMV evaluations bring diminishing returns in response stability. Therefore, the size of TMV counters may be significantly reduced. Further area optimization is possible if the first layer PUFs do not evaluate simultaneously, allowing operation with fewer TMV counters. This impacts

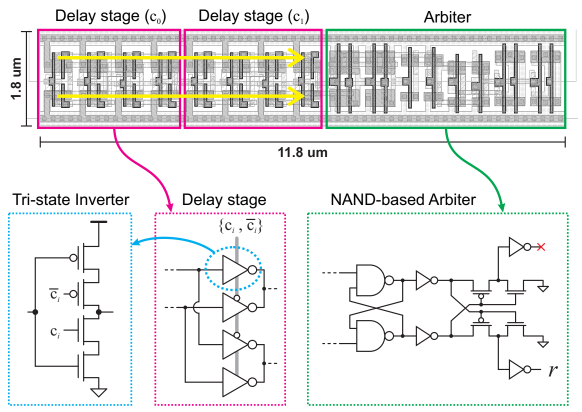


Figure 4.5: Custom-made layout of a 2 stages APUF. Same height as standard-cell logic for integration with EDA tools.

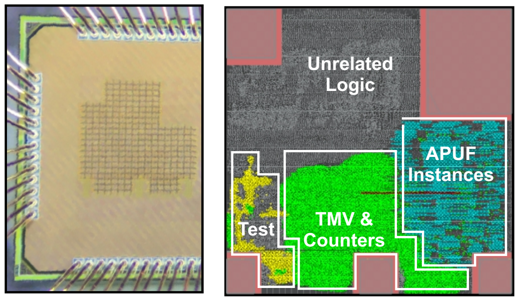


Figure 4.6: Die photo and layout view of implemented design. TMV logic is oversized since it uses larger counters than necessary.

throughput, but significantly reduces the TMV hardware size.

In summary, each testchip includes 6 different first layer implementations. Each implementation uses a different APUF size, including 2, 4, 6, 8, 12, and 24 stages. The first layer does not mix APUFs of different sizes. There is a single 64-APUF instance, therefore, the testchip includes only one second layer implementation.

4.5 Measurement Results

We measured a total of 10 dies to accurately assess uniformity, uniqueness, and response stability. Each die contains 6 different first layer implementations, and a single 64-APUF which implements the second layer. We performed enrollment at 20 °C. To calculate bit error rate, CRPs are evaluated 100 times at the *narrow* and *wide* temperature sets, which denote {0 °C, 20 °C, 60 °C}, and {-30 °C, 0 °C, 20 °C, 60 °C, 80 °C}, respectively. Temporal majority voting (TMV) is used in all evaluations. TMV with a single (one) repeated evaluation is equivalent to an ordinary evaluation without temporal majority voting. Boxplots show the distribution of *mean* values for different chips—they include 10 different mean values of the respective performance metric, one from each tested chip.

Bit error rate measures the stability of PUF responses. Fig. 4.7 (a), and (b) plot bit error rate for POP implementations with various APUF sizes in the first layer, for the narrow and wide temperature sets, respectively. A single first layer evaluation was

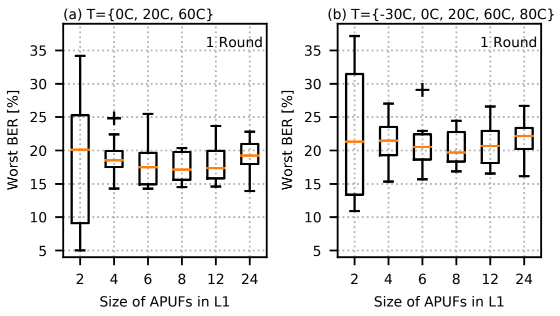


Figure 4.7: Worst BER for POP implementations with various APUF sizes in the first layer; in (a) for the narrow temperature set; and (b) for the wide temperature set. Measured with 5 k CRPs, single first layer evaluation (one round), and 15 TMV.

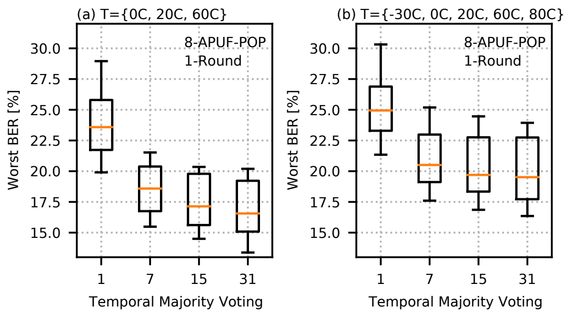


Figure 4.8: Worst BER for different repeated evaluation (TMV) values; (a) assess bit error rate over the narrow temperature set; and (b) over the wide temperature set. Measured with 5 k CRPs, and a single first layer evaluation (one round).

performed (one round). A total of 15 repeated evaluations were used for TMV. The median BER for 24-APUF-POP is 19.2%, and 22.1% in the narrow, and wide temperature sets, respectively. Minimum bit-error rate is found with 8-APUF-POP, where we measured median BER of 17.1%, and 19.7% for the narrow, and wide temperature sets. Small APUFs, with 2, 4, and 6 stages, showed a steep increase in the spread of bit error rate across different chips. This is likely related to stage bias, which is discussed in section 4.6.3.

Temporal majority voting (TMV) performs multiple evaluations of a PUF instance to remove noise from responses. We assessed the impact of TMV in POP response stability. Results plotted in Fig. 4.8 show bit error rate for different TMV settings. In (a), the median BER for 8-APUF-POP, in the narrow temperature set, is 23.5%, 18.5%, 17.1%, and 16.5%, for 1, 7, 15, and 31 repeated TMV evaluations. Increasing the number of repeated evaluations quickly reaches diminishing returns. A similar trend was observed for other sizes of POP, in both temperature sets. Therefore, 15 TMV offers a reasonable compromise between throughput and bit error rate for composite evaluations.

Our POP implementations adds support to multiple first layer evaluation rounds. We assessed the response stability for 1, 2, 3, and 4 evaluation rounds, using 1 k CRPs. Results are shown in Fig. 4.9 (a), (b), (c), and (d) for 2, 4, 8, and 24 stages in first layer APUFs, respectively. All implementations showed median BER below 20% for single round evaluations. Larger APUFs in the first layer lead to lower response stability when using additional rounds. Similarly to Fig. 4.7, small APUFs in the first layer show a wide spread

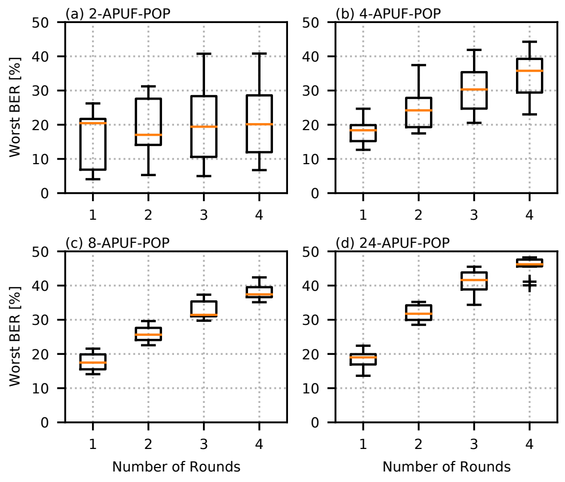


Figure 4.9: Bit error rate for various evaluation rounds using 2-APUF-POP in (a), 4-APUF-POP in (b), 8-APUF-POP in (c), and 24-APUF-POP in (d). Measured in the narrow temperature set, with 1 k CRPs, and 15 TMV.

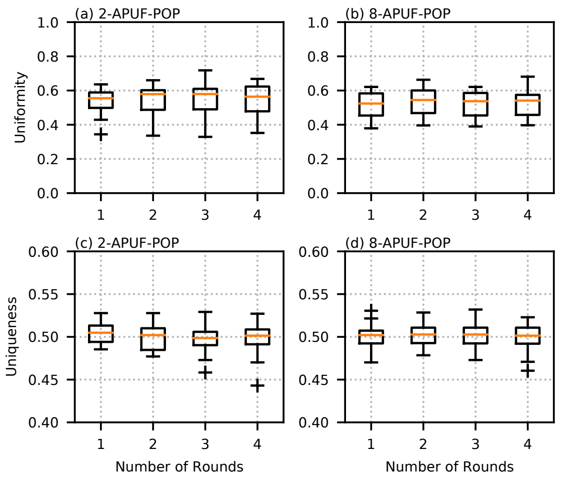


Figure 4.10: Uniformity and uniqueness for various evaluation rounds for 2-APUF-POP in (a, c), and 8-APUF-POP in (b, d). Measurements were performed with 1 k CRPs at 20 °C, with 15 TMV.

of BER measurement across multiple chips. Therefore, median BER for 2-APUF-POP, and 4-APUF-POP, may be a misleading performance metric, if not accompanied by the corresponding standard deviation.

We also measured uniformity and uniqueness of POP instances in our testchip. The former, captures the balance of zeros and ones in the final response, while the later, measures the normalized hamming distance of different instances. Measured uniformity and uniqueness for various rounds are plotted in Fig. 4.10, for 2-APUF-POP in (a, c), and 8-APUF-POP in (b, d). Both uniformity, and uniqueness showed a small variation in performance values across different sizes of POP. The median uniformity for 2-APUF-POP, and 8-APUF-POP, with one evaluation round, is 0.55, and 0.52, respectively. The corresponding median uniqueness is 0.50 for both 2-APUF-POP and 8-APUF-POP. Other POP implementations with different sizes presented similar results.

In section 4.6.4 we show that small APUFs in the first layer lead to poor hamming distance in the intermediate responses. Such result is not noticeable when evaluating the final output of POP, but it is likely the cause of security vulnerabilities studied in the next section.

4.6 Security Assessment

The primary motivation for composite strong PUFs is to increase security against learning attacks. We start this section discussing the applicability of logistic regression, cryptanalysis, and reliability attacks to POP. Next we describe how deep neural networks (DNNs) are used to model strong PUFs. We assess the learning resilience of our POP implementations against (DNNs) using up to 10 M CRPs. Next, we discuss influential bits, and hamming distance of intermediate responses, showing that compositions do not necessarily preserve the security properties of PUFs—the size of composing PUFs plays a crucial role.

4.6.1 Learning Attack Results

Logistic regression and cryptanalysis: logistic regression (LR), and cryptanalysis attacks are described in [76] and [79]. LR uses gradient descent to find coefficients of a linear model that minimize the prediction error. Modeling POP with LR is non-trivial, since careful mathematical manipulation is required for an adequate linear fitting [116]. The cryptanalysis attack exploits the mapping of challenge bits to different APUFs in the first layer. It was shown in [116], that such attacks are ineffective against POP due to the wiring scheme used in the first layer—every challenge bit is fed to more than a single PUF instance.

Reliability based attacks: reliability based attacks were initially introduced in [13]. The key insight is that CRPs with small delay difference are more susceptible to noise. Authors demonstrated the attack efficacy against XOR-APUFs using response stability as side-channel information. The POP architecture, however, uses a construction where noisy CRPs in the first layer affect the final output with different probabilities. Such characteristic is described in detail in section 4.6.2. Therefore, analogously to the formal proof provided for the interpose PUF in [66], reliability-based attacks are unlikely to obtain better prediction accuracy than other learning attacks that do not exploit response stability information.

Deep neural networks: deep-neural networks (DNNs) are emerging as an efficient attack technique capable of learning complex PUF structures. DNNs do not require a mathematical model of the PUF being modelled. We use a 12-layer DNN architecture proposed in [46] for our DNN attacks. The input and output layers have 64, and 2 units, respectively. Hidden layers have 2000 units. Performance metrics in section 4.5 were calculated using CRPs generated from our testchip, but our attack experiments use CRPs from a high-level model to avoid noise as confounding factor for prediction accuracy. Table 4.2

Table 4.2: Learning attack results.

Implementation	**Rounds**	**BER**	**Area (ND2)**	**Accuracy**
2-APUF-POP	1	20.0%	1.2 k	81.8%
4-APUF-POP	1	18.5%	1.7 k	74.7%
6-APUF-POP	1	17.4%	2.2 k	48.0%
8-APUF-POP	1	17.1%	2.7 k	51.5%
12-APUF-POP	1	17.3%	5.4 k	49.6%
24-APUF-POP	1	19.2%	6.8 k	54.4%
2-APUF-POP	1	20.0%	1.2 k	81.8%
2-APUF-POP	2	17.0%	1.2 k	91.3%
2-APUF-POP	4	20.1%	1.2 k	96.1%
2-APUF-POP	8	–	1.2 k	99.5%
24-APUF-POP	1	19.0%	6.8 k	54.4%
24-APUF-POP	2	31.7%	6.8 k	55.0%
24-APUF-POP	4	41.6%	6.8 k	56.4%
24-APUF-POP	8	–	6.8 k	56.2%

Notes: worst BER is the worst bit error rate (median) from {0 °C, 20 °C, 60 °C}, using 15 repeated evaluations for TMV. The area includes only APUF instances (control and TMV logic not included). All attacks were performed using deep neural networks (DNNs), with 10 M CRPs and 72h training time.

summarizes DNN attack results using 10 M CRPs. The *Implementation* column specifies POP parameters used, for example, the 2-APUF-POP uses 64 instances of 2-APUF in the first layer. All our POP implementations use a 64-APUF instance in the second layer. The *Rounds* column refers to the number of first layer evaluations used, prior to the second layer evaluation. The *BER* column reports the median of worst bit error rate among the temperature points of {0 °C, 20 °C, 60 °C}. The *Area* column reports the silicon area (normalized by the NAND2 area) for all APUF instances, excluding control, and TMV logic. The *Accuracy* column reports the accuracy obtained after 72 h of training using a Quadro P4000 GPU.

Results reported in Table 4.2 show that the DNN model achieved accuracy of 81.8% for 2-APUF-POP using 1 round. This implementation has median BER of 20.0%, and uses an area of 1.2 k ND2. Increasing the size of APUFs in the first layer to 4, 6, 8, 12, and 24 stages reduces prediction accuracy to 74.7%, 48.0%, 51.5%, 49.6%, and 54.4%, respectively. Therefore, our results suggest that increasing the size of APUFs in the first layer strengthens the POP composition, at an area and response stability cost.

We explored multiple first layer evaluation rounds as a cost-effective approach for strengthening POP against learning attacks. Table 4.2 reports bit error rate and prediction accuracy result for 2-APUF-POP and 24-APUF-POP, using 1, 2, 4, and 8 rounds. Response stability falls as more evaluation rounds are used. We measured median BER of 17.7% and 31.7% for 2-APUF-POP and 24-APUF-POP when using 2 evaluation rounds, respectively. Prediction accuracy for 24-APUF-POP did not shown significant change when varying number of rounds, however, prediction accuracy for 2-APUF-POP increased when more rounds are used. This result was unexpected, and counter-intuitive. We carefully reflect on possible explanations for such outcome in the next sections.

4.6.2 Probability of Output Change

The DNN prediction accuracy when using small APUFs in the first layer, reported in section 4.6.1, deserves additional investigation. In this section, we use the concept of strict avalanche criterion (SAC) to look into the differences between APUFs of various sizes.

As defined in [113], if a cryptographic function is to satisfy the strict avalanche criterion (SAC), then, each output bit should change with a probability of one half, whenever a single input bit is complemented. In [60, 65] this concept was extended to strong PUFs, where authors measure the probability of output response change given a single bit change in the input challenge. Furthermore, it was demonstrated that the probability of output change for the APUF, depends on the distance between toggled bit and the arbiter. Challenge bits applied to stages near the arbiter are more likely to cause a change in the response. This result is reproduced in Fig. 4.11 (a). Using simulation, we estimate the probability of output change for 64-APUF when evaluating a random challenge, before, and after it is XORed with a mismatch pattern $\mathbf{e}$. When $\text{HW}(\mathbf{e}) = 1$, a single challenge bit will toggle between evaluations. In the plots of Fig. 4.11, the position of the toggled bit is shifted towards the arbiter, and denoted as mismatch pattern shift. When the pattern shift is zero, probability of output change is 5.5%, but as the toggled bit nears the arbiter, probability of output change increases, reaching 90.5% at the last APUF stage. This result can be intuitively explained by the conditional wire twist present in every stage of the APUF, and the cumulative nature of the delay path.

We also estimate the probability of output change when two consecutive challenge bits are toggled. This is plotted in Fig. 4.11 (a) as $\text{HW}(\mathbf{e}) = 2$, showing that, for 64-APUF, the probability of output change remains nearly constant, at 8%. This result represents a more realistic view on the SAC criterion for APUFs, where minimal input change requires toggling two adjacent stages, instead of a single one. The same technique was applied to APUFs of various sizes in Fig. 4.11 (b). The estimated probability of output change for

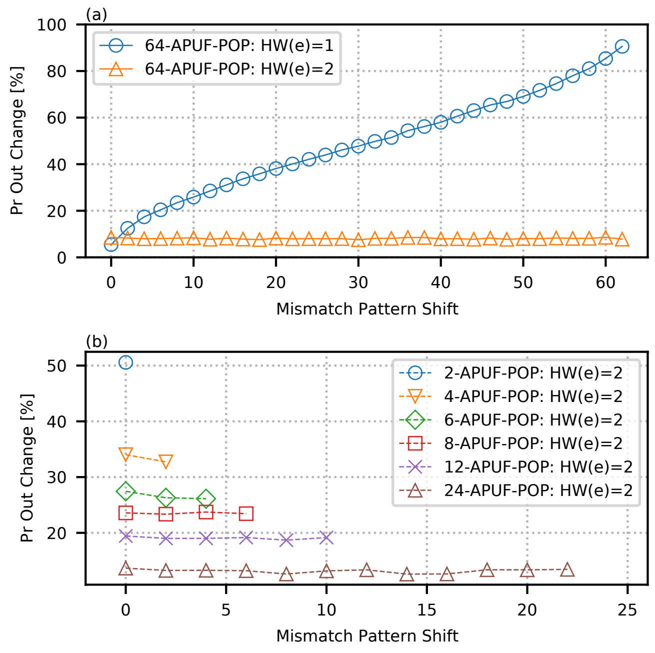


Figure 4.11: Simulated probability of output change when evaluating a random challenge, before, and after it is XORed with a mismatch pattern. Results in (a) for 64-APUF, and in (b) for smaller APUF sizes. The HW(**e**)=2 only includes mismatch patterns where nonzero bits adjacent.

APUFs with 24, 12, 8, 6, 4, and 2 was 13.2%, 19%, 23.6%, 26.5%, 33.5%, and 50.5%, respectively. The increase in probability of output change for smaller sizes of APUFs gives an important insight to understand the results found in section 4.6.1: the influence of individual stages on the output increases, as APUF size decreases.

4.6.3 Influential Bits and Stage Bias

Influential bits were previously studied in [29, 84, 117]. Authors showed how distinct challenge bits have different influence on the output of a bistable ring PUF (BR-PUF). Based on the value of a few influential challenge bits, it is possible to predict responses with high accuracy [29]. To avoid confusion with previous work nomenclature, we denote the *influence* of each challenge bit as *stage bias*. This section performs an assessment of stage bias in APUFs of various sizes, leading to conclusions that help explain learning attack results obtained in section 4.6.1.

Algorithm 1 Stage bias assessment for an APUF instance.

```
1: let $NC$ be the number of challenges
2: let $CW$ be the challenge width in bits
3: let $y$ and $n$ be zero initialized matrices of size $(2, CW)$
4: for $i = 0$ to $NC - 1$ do
5:     $c = \text{RandomizeChallenge}()$
6:     $r = \text{EvaluateResponse}(c)$
7:     $p = 0$
8:     for $j = 0$ to $CW - 1$ do
9:         $p = p \oplus c[j]$
10:    end for
11:    for $j = 0$ to $CW - 1$ do
12:        $t = c[j]$
13:        $p = p \oplus t$
14:        $y[t, j] \mathrel{+}= (r \oplus p)$
15:        $n[t, j] \mathrel{+}= 1$
16:    end for
17: end for
18: $y = y/n$
```

To the best of our knowledge, no previous literature reports the measurement of stage bias in APUFs. We introduce Algorithm 1 for measuring stage bias in APUFs. The main idea is to evaluate a set of randomized challenges, keeping track of responses statistics per stage, and per challenge bit value. The key insight of Algorithm 1 is on line 14. When summing the response, r, for challenge bit value t, at stage j, the response is conditionally inverted (XORed) with p, where p is a *parity bit* (reduced XOR operation) over the challenge bits from position $j + 1$, onwards. If the number of twisted stages after position j is odd, the value of p will be 1, which then inverts the response r. The final stage bias is stored in the matrix y, indexed by challenge bit value, and by stage position. Notice that the division operation in line 18 is element-wise.

Using CRPs from our testchip, we calculated stage bias for different APUF sizes. The results are shown in Fig. 4.12. A distinction was made for stage bias when c_i is zero (first row), and one (second row), since challenge bits select between two pairs of inverters in each stage, and each pair exerts different influence on the response. The plots (a, e), (b, f), (c, g), and (d, h), refer to APUFs with 2, 4, 8, and 24 stages, respectively. The results refer to a single APUF instance of each size (not the overall POP composition).

Results plotted in Fig. 4.12 express the probability of response r being equal to $(1 \oplus p_i(\mathbf{c}))$, given challenge bit c_i is zero, or one. The term $p_i(\mathbf{c})$ is denoted as *parity*, and will have a value of one when the challenge imposes an odd number of wire twists between the stage under analysis i, and the arbiter. Parity is calculated as

$$p_i(\mathbf{c}) = \bigoplus_{j=i+1}^{n-1} c_j. \tag{4.1}$$

For example, based on data from Fig. 4.12 (f) for the 4-APUF, the probability of $r = 1$, given $c_3 = 1$, is 0.38, or equivalently, probability of $r = 0$, given $c_3 = 1$, is 0.62. In this case, the parity calculated by Eq. 4.1 is zero, since there are no stages that could twist the wires between position 3 and the arbiter. As an additional example, the stage bias reported in Fig. 4.12 (c) for the 8-APUF shows that, the probability of $r = (1 \oplus p_i(\mathbf{c}))$, given $c_4 = 0$, is 0.76. Therefore, all challenges which have $c_4 = 0$, and an even number of ones in (c_5, c_6, c_7), have 0.76 probability of evaluating to 1. Moreover, challenges that have $c_4 = 0$, but an odd number of ones in (c_5, c_6, c_7), have a 0.76 probability of evaluating to 0.

Large stage bias deviations from 0.5 are undesirable, since they grant certain challenge bits an unfair influence over the response. It was also shown that large stage bias can be exploited by attackers [29]. Section 4.6.4 presents data suggesting that large stage bias is correlated with poor uniqueness performance. To understand how stage bias varies across APUFs of different sizes, we simulated 100 APUF instances using 3 k CRPs, and plotted the stage bias distribution in Fig. 4.13. The stage bias mean is 0.5 for all APUF sizes, but the standard deviation increases significantly for smaller APUFs. Therefore, we may conclude what is also apparent in the measurements presented in Fig. 4.12, fewer APUF stages increase the likelihood of large stage bias deviations.

4.6.4 Hamming Distance of Intermediate Responses

Previous section assessed the effects of smaller APUFs on stage bias, showing that reducing the size of APUFs creates challenge bits that hold large influence over the final response. Such result motivates an investigation of the hamming distance between responses produced by the first layer of the POP architecture, over multiple rounds (same challenge), and across multiple challenges. Although mathematically similar, we avoid using the term uniqueness for such experiment, since it is not applied to the final POP response.

The normalized hamming distance (HD) measures the distance between two numbers, divided by their length. The normalized HD is herein denoted as distance, for short.

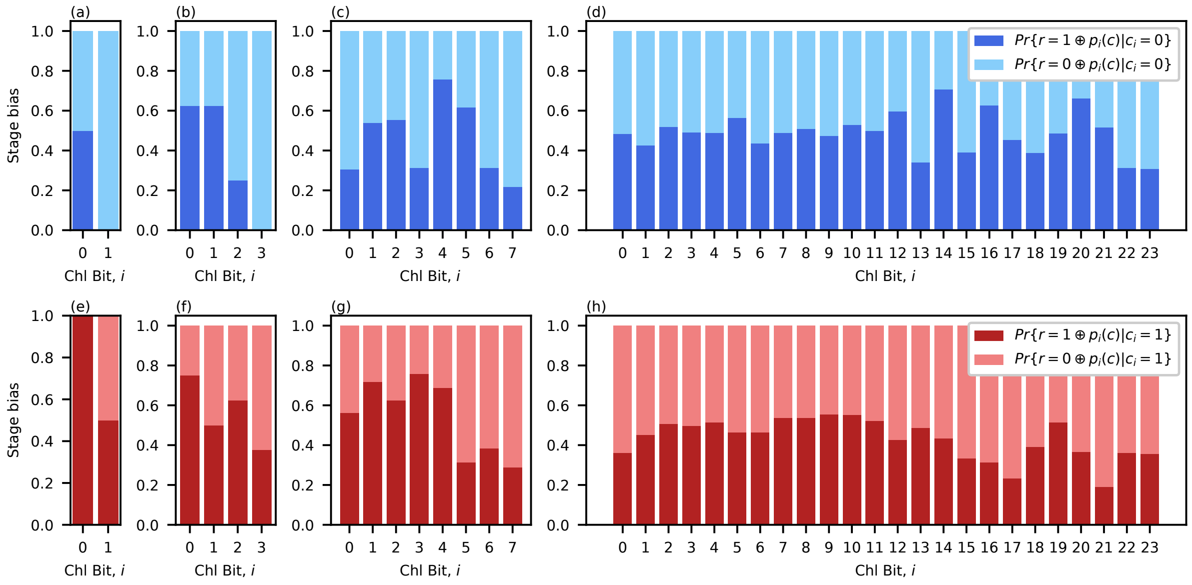


Figure 4.12: Stage bias calculated using CRPs from our testchip. The first row shows biases when $c_i = 0$, while the second row shows biases when $c_i = 1$. The plots (a, e), (b, f), (c, g), and (d, h), refer to APUFs with 2, 4, 8, and 24 stages, respectively. We used 100 k CRPs to calculate stage bias for the 24-APUF, while APUFs of 2, 4, and 8 stages were enumerated (all CRPs were collected).

For example, if the distance between two numbers is 0.5, it implies that half the bits of their binary representation differ. Fig. 4.14 (a) plots the average distance between the responses produced by the POP first layer, across multiple rounds, for the same challenge, for various APUF sizes. For instance, the data denoted as *(1,2)-round* reports the average distance between responses produced from first to the second evaluation round. While implementations with 24-APUF show nearly ideal distance of 0.5 across all evaluation rounds, reducing the size of APUFs gradually degrades the distance between responses. In implementations with 2-APUF, the average distance between responses from first to the second round is 0.36, which implies that 64% of response bits from the first evaluation remained unchanged after the second evaluation. As the number of rounds increases, average distance for 2-APUF continues to fall, reaching 0.24 for responses between fourth, and fifth rounds.

We also examine the average distance of first layer responses, across multiple challenges, after 1, 2, 4, and 8 evaluation rounds. Results are plotted in Fig. 4.14 (b). Implementations using 24-APUF show nearly ideal distance of 0.5 across multiple challenges, but reducing

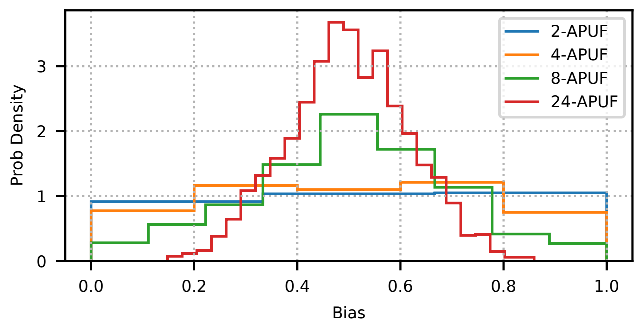


Figure 4.13: Distribution of stage bias for obtained from simulation with 100 APUF instances and 3 k CRPs.

the size of APUFs, gradually degrades the distance between responses—this time, across multiple challenges. For example, 2-APUF implementations show average distance of 0.36, 0.30, 0.25, and 0.22 when evaluated with 1, 2, 4, and 8 rounds. Moreover, the data suggests that even with a single evaluation round, small APUFs fail to produce responses with distance near 0.5. Another perspective to such result, is to consider that small APUFs limit the challenge space of the second layer APUF, seriously impacting to the learning resilience of the overall composition.

The poor hamming distance performance of smaller APUFs, likely caused by larger stage bias, is a plausible cause for the learning results observed in section 4.6.1. The bad performance seen for APUFs with 2 and 4 stages in Fig. 4.14 agrees with our attack experiments, where DNNs showed reduced prediction accuracy starting from 6-APUF-POP implementations and above (see section 4.6.1). It is also important to notice that our results do not assess the benefits of multiple round evaluations for larger APUFs in the first layer. In terms of prediction accuracy, those implementations were already resilient to DNN attacks with a single round. Our analysis shows, however, that there is no apparent reduction in challenge space when multiple evaluations rounds are used in first layer implementations with 12, and 24 stages.

4.7 Conclusion

We explored the design space of the POP architecture using APUFs of various sizes. We performed extensive DNN attacks to assess the security of POP. Our results suggest an increase in learning resilience when using APUFs with 6, or more, stages in the first layer.

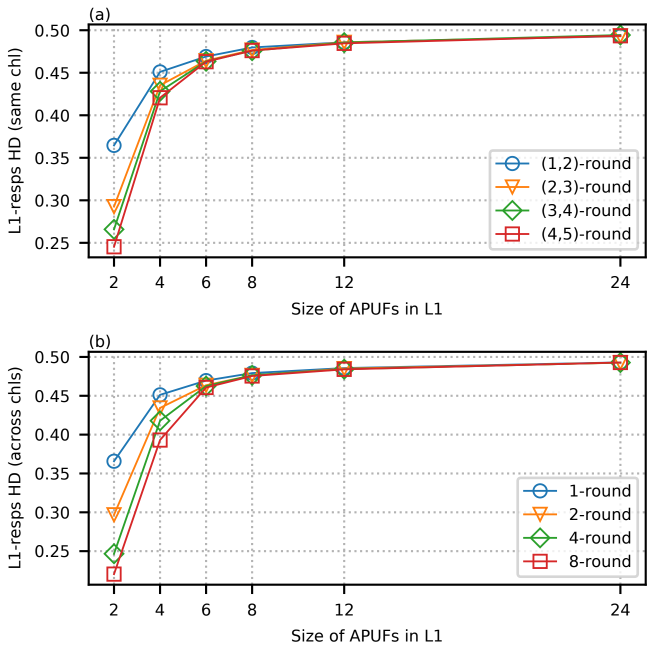


Figure 4.14: Simulation of the average normalized hamming distance (HD) between responses for various sizes of APUFs in first layer (L1). Across multiple rounds, for the same challenge in (a); and across multiple challenges, after 1, 2, 4, and 8 evaluation rounds in (b).

Compositions using APUFs with 2, and 4 stages are shown vulnerable to DNN attacks. Moreover, POP implementations with 2 stage APUFs in the first layer show a trend of higher prediction accuracy as the number of evaluation rounds increases. To study such result, we extended previous techniques of influential bits to assess stage bias in APUF instances. Our data suggests that small APUFs in the first layer limit the challenge space of the second layer APUF, showing that compositions do not always preserve security properties of PUFs. Measurements from our testchip show that minimum bit error rate is obtained when using APUFs with 8 stages, while fewer APUF stages lead to a large spread in bit error rate across different chips.

Chapter 5

Key Based Challenge Obfuscation for Strong PUFs

In this chapter we propose a lightweight key based challenge obfuscation for strong PUFs. Our architecture is designed to be resistant against learning attacks. A high-level view is presented in Fig. 5.1. First, we XOR the external challenge with a secret key. The result is loaded into a non-linear feedback shift register (NLFSR), which is run for a number of cycles before the first evaluation (warm-up). The NLFSR state is then used as (obfuscated) challenge to evaluate the strong PUF. Responses are directly provided to the user—no error correction or post-processing is required. The secret key may be implemented with a one time programmable (OTP) flash, or weak PUF.

The mitigation of key extraction attacks is crucial for any challenge obfuscation architecture that uses a secret key. A brief review of common attack techniques is presented in chapter 2. In this chapter, we focus on power analysis attacks, but the solutions presented here help mitigate other types of attacks as well, such as probing and fault injection.

Protecting hardware implementations against power analysis attacks is costly, but effective [34]. We made careful design choices in our obfuscation architecture, such that side-channel mitigation techniques have minimum impact in cost. Our implemented NLFSR uses only two instances of non-linear logic gates, reducing the area overhead and complexity of Boolean masked implementations. We also discuss the implementation of a clock randomization countermeasure against power analysis attacks.

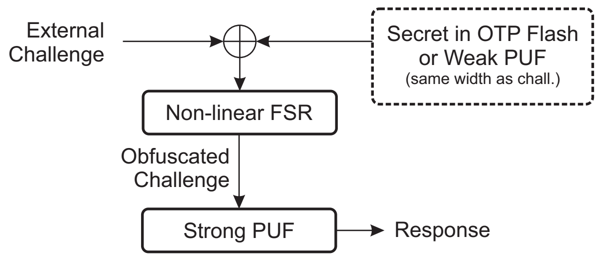


Figure 5.1: High-level view of our proposed key based challenge obfuscation architecture for strong PUFs.

5.1 Related Works

Previous works used composition of strong PUFs to obfuscate the internal challenge, but resilience to learning attacks is ultimately limited by the stability of responses [80,90,116]. Error corrected strong PUFs require external helper data to cope with its large challenge space, but such alternative was shown to be insecure [26]. Recent works employ weak PUFs to generate an error corrected chip-unique secret, which is then used to obfuscate the external challenge [45, 56, 107, 110–112]. The obfuscation algorithm must be lightweight, and secure against learning attacks. Moreover, manipulating external data (challenge) with sensitive information (secret key), requires countermeasures against power analysis attacks [48].

5.2 Challenge Obfuscation

This section describes our proposed architecture for challenge obfuscation, and its associated evaluation algorithm. We also briefly discuss some important details of two secret storage options, OTP flash, and weak PUF.

5.2.1 Non-linear Feedback Shift Register (NLFSR)

Non-linearity is a fundamental property for obfuscation algorithms. In the case of block ciphers, non-linear transformations are performed by substitution boxes (SBOXes). The implementation cost of SBOXes, however, is significant [98]. To achieve lightweight non-linear challenge transformation, we use NLFSRs. NLFSRs are deterministic digital circuits capable of non-linear state transitions. Unlike their linear counterpart, NLFSRs lack a solid

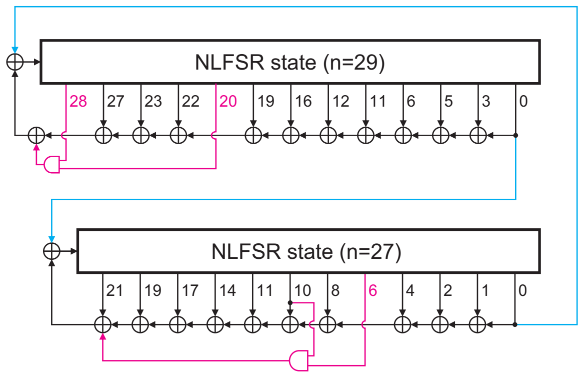


Figure 5.2: NLFSR arrangement used in challenge obfuscation architecture.

mathematical representation. The sequence length is found using brute force methods, therefore, maximum length NLFSRs hardly exceed 2^{31} [24].

Our NLFSR design is shown in Fig. 5.2. It uses a composition of two smaller NLFSRs, with 27 and 29 bits. Our feedback expressions have maximum length and were taken from [24]. A 56 bit challenge is obtained by concatenating the state of both NLFSRs. To make both states dependent of one another, we XOR the lowest significant bit of each NLFSR with the next state logic of the other—a similar technique was used in the Trivium cipher [75]. In the context of strong PUFs, registers are required to hold the challenge even if obfuscation is not used. Therefore, in addition to the control logic, the only overhead present in Fig. 5.2 are the logic gates that compute the next state.

Unless otherwise specified, we may refer to the 56 bit concatenated state simply as NLFSR state. Moreover, when clear from context, the term *state* might be omitted.

5.2.2 Evaluation Algorithm (no countermeasures)

The evaluation procedure for our proposed challenge obfuscation is listed in Algorithm 2. This algorithm is not yet protected against side-channel attacks. It assumes the secret key was read from OTP flash or weak PUF. First, the external challenge is XORed with secret key, and loaded in the NLFSR, which is run for a total of 112 *warm-up* cycles. Before each strong PUF evaluation, we *flush* the previous NLFSR state, running it for 56 cycles.

More warm-up cycles enhances security, but increases latency. The number of warm-up cycles (112) was determined empirically to satisfy the avalanche criterion (see section 5.5.1). Other NLFSR expressions may require shorter, or longer warm-up periods to satisfy the avalanche criterion.

Algorithm 2 Evaluate external challenge using an implementation without side-channel attack countermeasures.

Assumptions: secret key has been read from OTP flash, or weak PUF. Both secret key and external challenge are 56 bits.

```
1. XOR key with external challenge, and load result to NLFSR
2. Run NLFSR for 112 cycles (for warm-up)
3. Run NLFSR for 56 cycles (flush state)
4. Evaluate strong PUF with NLFSR state as challenge
5. Output the 1 bit response (no post-processing)
6. If number of response bits is enough: *return success*
7. Otherwise: *goto* step 3
```

5.2.3 Secret Key Storage

The secret key storage may be implemented with one time programmable (OTP) flash, or weak PUF. Since uniqueness and unclonability properties are already provided by the strong PUF, using a weak PUF for secret key storage is possible, but adds extra complexity. One may argue that if a secret key is stored in flash, there is no need for a strong PUF. Such statement is inaccurate. For example, if the strong PUF is removed, over-produced chips (with blank OTP flash) can be programmed to behave alike any other device. That is not feasible when a strong PUFs is included in the design.

Other relevant considerations for both secret key storage options include protection against fault injection attacks, such as voltage and clock glitches. Storing the secret key with error correction data, and performing multiple reads from memory for consistency checking was shown very effective in mitigating fault injections [104].

5.3 Power Analysis Attack Mitigation

Manipulating external data (challenge) with sensitive information (secret key) is vulnerable to power analysis attacks [48]. This section discusses countermeasures to mitigate such risks.

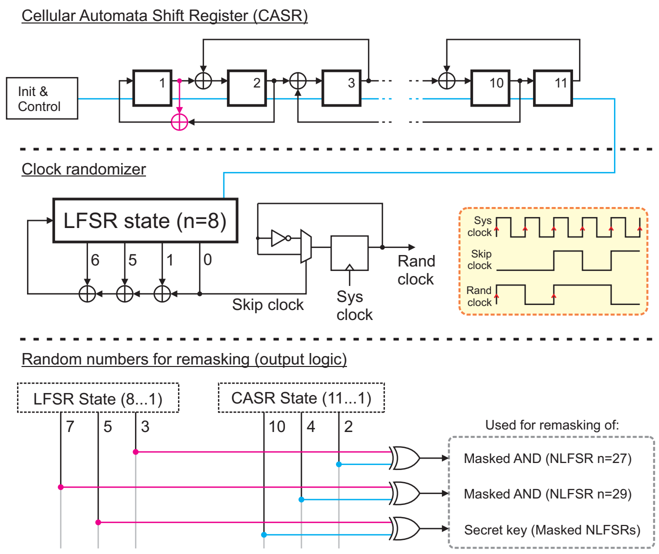


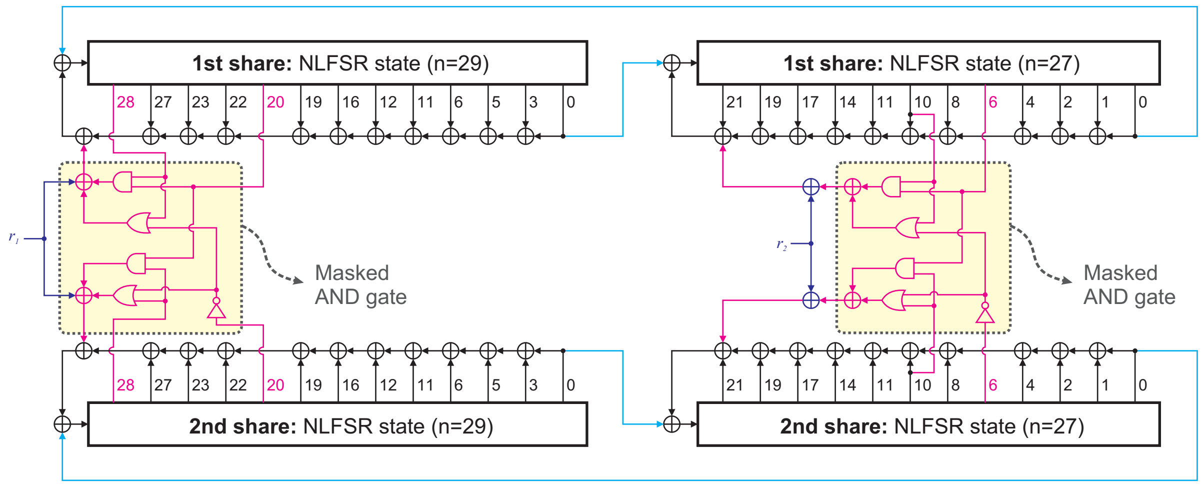
Figure 5.3: Clock randomization logic and pseudo random number generator (PRNG).

5.3.1 Pseudo Random Number Generator (PRNG)

Random numbers are necessary to implement the countermeasures described in this section. Our pseudo random number generator (PRNG) is based on [88], and is shown in Fig. 5.3. It uses a linear feedback shift register (LFSR) and a cellular automata shift register (CASR), with outputs derived from their combined (XORed) state. The PRNG has three outputs, each of which generates a new random number every clock cycle. LFSR and CASR have maximum sequence length, with expressions taken from [85], and [19]. Because their cycle length is relatively prime, the total cycle length of the output is close to 2^{19}. The seeding of LFSR/CASR states is performed using the available strong PUF. For that, we identify a challenge with unstable responses during enrollment, and store it in non-volatile memory. This challenge is repeatedly evaluated to generate random bits that initialize the LFSR/CASR states.

5.3.2 Clock Randomization

Power analysis attacks extract the key over a large number of recorded power traces. The effectiveness of power attacks is higher when traces are aligned in time. The clock randomizer circuit is shown in Fig. 5.3, it produces an irregular clock waveform, which will randomly skip clock edges. Our obfuscation architecture uses the randomized clock



1st share: NLFSR state (n=29)
1st share: NLFSR state (n=27)
2nd share: NLFSR state (n=29)
2nd share: NLFSR state (n=27)
28 27 23 22 20 19 16 12 11 6 5 3 0
21 19 17 14 11 10 8 6 4 2 1 0
r_1
r_2
Masked AND gate
Masked AND gate

Figure 5.4: Boolean masked implementation of the NLFSR used for challenge obfuscation. Random numbers r_1 and r_2 are outputs from the PRNG.

output. The *skip clock* signal is derived from the 8 bit LFSR, therefore, the randomized clock waveform pattern repeats every 255 cycles, allowing authentication with predictable performance.

5.3.3 Boolean Masking of the NLFSRs

Boolean masking uses random numbers to split each value into two shares that are (ideally) uncorrelated to the original value. For example, the shared representation of x is (x_1, x_2), which are computed as $x_1 = x \oplus r$, and $x_2 = r$, where r is a random number. The original value is recovered by XORing the shares, therefore, $x = x_1 \oplus x_2$. Probing of both shares is necessary to inspect the original data. When operations are performed with shared data, the power consumption is, in theory, uncorrelated to the data being processed. Moreover, if the secret key is stored in two shares, its plain-text value is never exposed.

Converting conventional single share circuits to operate with multiple shares of data differs for linear and non-linear operations. Linear operations, like XOR, shifts, and permutations, are simply applied to both shares without any changes. Non-linear operations, however, require a replacement circuit that will compute the shared outputs without disclosing the original value. We redesigned the NLFSRs described in section 5.2.1 to run using multiple shares. The resulting circuit is shown in Fig. 5.4. The number of state registers doubles to accommodate both shares of all values. Since XORs and shifts are linear operations, the NLFSR design remains mostly unchanged, except for the AND gates, which are replaced by their masked counterpart. The expression for the masked AND was taken from [17]. In fact, during the design of our challenge obfuscation architecture, we intentionally selected NLFSR expressions with a small number of non-linear gates to reduce the cost and complexity of masked implementations. For example, glitches are a well known source of information leakage in masked non-linear logic [62], but our original design has both AND inputs driven by registers—the best practice was already implemented.

Other design details include remasking the AND gate at every cycle, which is done by XORing fresh random numbers, r_1 and r_2, with both shares of the masked AND output. Remasking helps remove possible correlations between the original data and the shared values.

When the NLFSR warm-up cycles are completed, their shared state is XORed to obtain the unmasked, obfuscated challenge, necessary for evaluation. It is very important that this XOR operation is only performed after completion of all warm-up cycles. In other words, the XOR inputs must be gated during warm-up to avoid information leakage from the toggling XOR outputs.

5.3.4 Evaluation Algorithm (with countermeasures)

The evaluation procedure for the masked implementation is listed in Algorithm 3. Similarly to Algorithm 2, it assumes that the secret key was read from OTP flash, or weak PUF. However, the key is expected in two shares of 56 bits each. First, the PRNG is seeded and the clock randomizer is enabled. The NLFSR is then seeded with random numbers from the PRNG. Both shares must be loaded with the same random number. The secret key is then XORed with the NLFSR state, which essentially performs a remasking operation of the key shares. Next, the NLFSR is run for 128 cycles to allow random delay insertion by the clock randomizer. The external challenge is then XORed with current NLFSR state. The second share of the external challenge is considered to be zero. This will not leak information since the NLFSR content is already masked with fresh random numbers. The remaining steps are analogous to Algorithm 2, with the extra requirement of XORing the NLFSR shares to unmask the obfuscated challenge before each strong PUF evaluation.

Algorithm 3 Evaluate external challenge using an implementation with clock randomization, and Boolean masking.

Assumptions: secret key has been read from OTP flash, or weak PUF in two shares of 56 bits each. External challenge is 56 bits, with second share being all zeros.

```
1. Using the challenge with unstable responses, (serially) seed the PRNG with random numbers ($8 + 11 = 19$ bits)
2. Enable the clock randomizer
3. Using the PRNG output, (serially) seed the NLFSR with the same 56 bit random number in both shares
4. XOR the NLFSR content with the key, and load result back to NLFSR (for key remasking)
5. Run the NLFSR for 128 cycles (for time misalignment)
6. XOR external challenge with NLFSR, and load result back to NLFSR
7. Run the NLFSR for 112 cycles (for warm-up)
8. Run NLFSR for 56 cycles (flush state)
9. XOR the NLFSR shares and evaluate strong PUF with the unmasked state (obfuscated challenge)
10. Output the 1 bit response (no post-processing needed)
11. If number of response bits is enough: *return success*
12. Otherwise: *goto* step 8
```

5.4 Testchip Implementation

To accurately assess the effectiveness of our countermeasures against power analysis attacks, we designed a 65 nm testchip. Our design was submitted for fabrication, and the RTL code is publicly available [89]. Table 5.1 shows the post-layout area results for the three implementations included in the testchip. The number of instances, and area are listed for each implementation, and its components. Area for test logic is not reported. All implementations use 7 repeated evaluations for enhanced response stability. The 56-APUF is a custom designed arbiter PUF with 56 delay stages, see chapter 4 for APUF circuit details. The clock randomizer and CASR were split in two different design blocks. The placement of standard-cells and layout is shown in Fig. 5.5.

The 56-LFSR-APUF uses an APUF with challenge stored/unrolled by an LFSR. It represents the smallest viable authentication system that can be built using an arbiter PUF. It uses an area of 1583 ND2, which we define as our comparison baseline. The 56-NLFSR-

Table 5.1: Testchip area results in 65 nm CMOS (post-layout).

	# Inst.	**Area (μm^2)**	**Area (ND2)**
56-LFSR-APUF	**490**	**2279**	**1583**
LFSR	152	754	524
56-APUF	1	332	231
Control logic & buffers	337	1193	829
56-NLFSR-APUF	**717**	**2904**	**2017**
NLFSR	186	854	593
Clock randomizer	24	148	103
56-APUF	1	332	231
Control logic & buffers	506	1570	1090
56-NLFSR-APUF (Masked)	**1448**	**5016**	**3483**
Masked NLFSR	494	2057	1428
Clock randomizer	25	148	103
CASR	65	237	165
56-APUF	1	332	231
Control logic & buffers	863	2243	1557

Notes: area associated to the secret key storage/read is not included. Area is reported in μm^2 and normalized by NAND2.

APUF implements our challenge obfuscation architecture, without Boolean masking. This implementation is not completely unprotected against power analysis attacks. It includes a clock randomizer instance, which represents 5% of the used area. The overall area of the 56-NLFSR-APUF implementation is 27% (1.27x) larger than the baseline. Finally, the 56-NLFSR-APUF (Masked) denotes the fully masked implementation, with clock randomization. The area for this implementation is 120% (2.2x) larger than the baseline. Results reported in chapter 6 suggest that clock randomization alone delivers 2260x increase in resilience against power analysis attacks. An even greater increase, of 17330x, is achieved when clock randomization is combined with Boolean masking.

5.5 Security Assessment

This section evaluates the security of our challenge obfuscation architecture using strict avalanche criterion, and deep-neural networks.

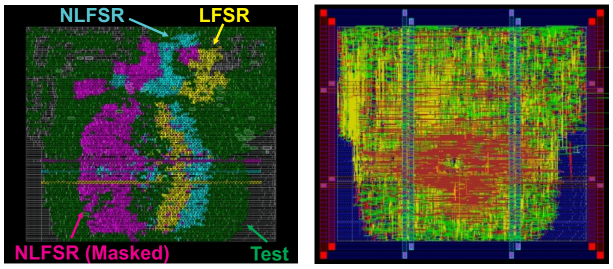


Figure 5.5: Testchip placement of standard-cells and layout in 65 nm CMOS.

5.5.1 Avalanche Criterion

As defined in [113], if a cryptographic function is to satisfy the strict avalanche criterion (SAC), then, each output bit should change with a probability of one half, whenever a single input bit is complemented. We assess the avalanche criterion from the perspective of our obfuscation algorithm—input is the external challenge, and output is the NLFSR state after (112 + 56) cycles. Our experiment used 10000 unique external challenges, each of them was run twice, with a single toggled bit between runs. The secret key is randomized at the beginning and remained unchanged during the experiment.

Fig. 5.6 (a), (b), and (c) show that toggling a single input bit causes a widespread (avalanche) effect in the NLFSR state, where each obfuscated challenge bit has an estimated 50% probability of changing. Same behavior was observed when other bit positions were toggled. For comparison, we replaced the NLFSR by a 56 bit linear feedback shift register (LFSR) of maximum sequence length (same as our baseline implementation discussed in section 5.4). The LFSR experiment also evaluates using a secret key, with output taken after (112 + 56) cycles. Results for the LFSR are reported in Fig. 5.6 (d), (e), and (f), showing that the effects of a single toggled input bit on the final LFSR state are deterministic, that is, either 0%, or 100%. Therefore, LFSR based challenge obfuscation fails to meet the avalanche criterion.

5.5.2 DNN Attacks

Deep-neural networks (DNNs) are capable of learning complex PUF structures, without a mathematical model of the PUF being modelled. We performed experiments using a challenge obfuscated arbiter PUF, and a 12-layer DNN architecture similar to [46]. The input and output layers have 56, and 2 units, with 2000 units in hidden layers. The DNN was trained for 72 hours using 10 million CRPs, with a resulting accuracy equivalent to

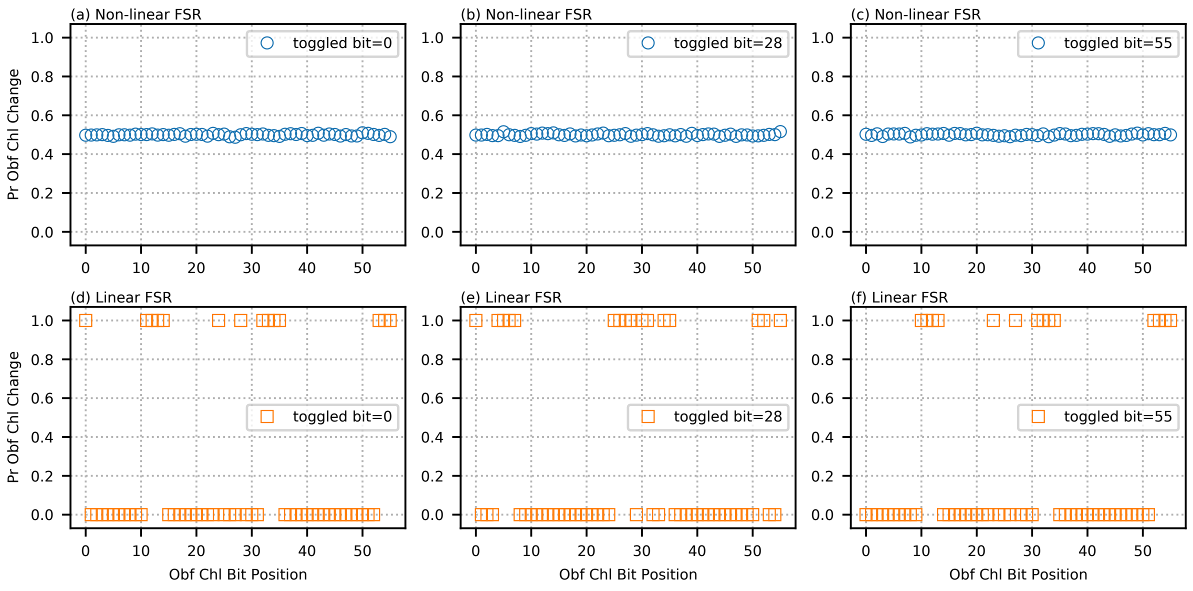


Figure 5.6: Probability of change in NLFSR/LFSR state when a single bit of the external challenge changes.

the PUF uniformity bias, which was 56% in our experiment. Therefore, the DNN model failed to obtain generalized learning on the challenge obfuscated arbiter PUF.

5.6 Conclusion

We demonstrated the design of a lightweight key based challenge obfuscation for strong PUFs. We addressed security for both learning, and power analysis attacks. Future work shall use the fabricated testchip to assess the effectiveness of our countermeasures against power analysis attacks.

Chapter 6

Design and Implementation of a Secure Microprocessor

Previous chapters discussed PUFs, and their applications for counterfeit prevention. PUFs establish trust between the user and the hardware. However, to accommodate modern internet protocols, devices should also provide mechanisms for message integrity, confidentiality, and (often) non-repudiation [114]. The National Institute of Standards and Technology (NIST), has standardized a number of algorithms that offer the aforementioned properties, but solutions require on-chip storage of a secret key. Quoting Ron Rivest, at CRYPTO 2011, “Merely calling a bit string a secret key does not make it secret, but rather identifies it as an interesting target for the adversary”.

Similarly to chapter 5, this chapter focus on power analysis attacks, but the described solutions help mitigate probing and fault injection attacks as well.

Traditional countermeasures only target a few most vulnerable elements, such as system bus, memory, and cryptographic accelerators. In this chapter, we demonstrate a bit-serial RISC-V microprocessor implementation with no plain-text data in the clear. All values are protected using Boolean masking (BM) and differential domino logic (DDL). Our architecture sets no constraints on how sensitive data must be manipulated. Software implementations can run with little to no countermeasures, reducing code size and performance overheads.

Unlike previous literature, our methodology is fully integrated to the ASIC digital design flow, requiring no changes to the input RTL. We used a bit-serial microprocessor due to area constraints, but our techniques can be applied to digital designs of any size or complexity. Original circuit functionality and latency are not affected.

6.1 Related Works

As discussed in chapter 2, secret keys can be extracted from the power consumption or electromagnetic emanations of unprotected devices [48,73]. Boolean masking was proposed as a possible countermeasure in [34]. It splits each value into two or more shares that are (ideally) uncorrelated to the original value. Most of Boolean masking literature is centered at protecting non-linear operations in cryptographic algorithms, but interest in larger scopes of protection has risen in recent years. In [36], a microprocessor ALU is masked using TI. Other works applied masking to a binarized neural network [27], and to RISC-V microprocessors [25,37]. In this chapter, we explore low cost masking expressions which are easy to integrate into already existing designs [17].

Differential logic styles with return to zero encoding make power consumption less data-dependent. For example, in [99] a sense amplifier based logic (SABL) is used to reduce asymmetries of the dynamic logic. In [100], wave dynamic differential logic (WDDL) uses static CMOS gates to replicate the behavior of dynamic logic, but glitches still leak sensitive information. Our work uses a well known differential domino logic style [74], but we innovate in the applied methodology. Instead of "semi-custom" techniques, our scripts are integrated with EDA tools and perform automatic replacement of relevant cells by dynamic cells, without affecting place and route or timing analysis.

High-throughput random number generators (RNGs) are essential for effective masking of digital circuits. Large literature exists for harvesting random numbers from different entropy sources [22, 118, 123]. But perhaps the most well understood entropy source is still phase jitter [11]. While cryptographic algorithms need high-quality random numbers with forward/backward secrecy [47], masking implementations tolerate random numbers generated by lightweight RNG designs that focus on throughput—and that is where our RNG is positioned.

6.2 Hardware Architecture

6.2.1 Top Level Design

Our design has three RISC-V microprocessors (see Fig. 6.1). Each microprocessor is designed to test a different set of security countermeasures. They share two SRAM memories for code, data, and register file (RF). The test logic selects which one of the three microprocessors will be operational, as well as the desired configuration for clock edge randomization

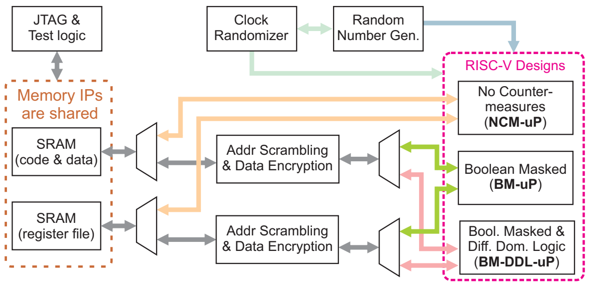


Figure 6.1: Top level block diagram of our testchip.

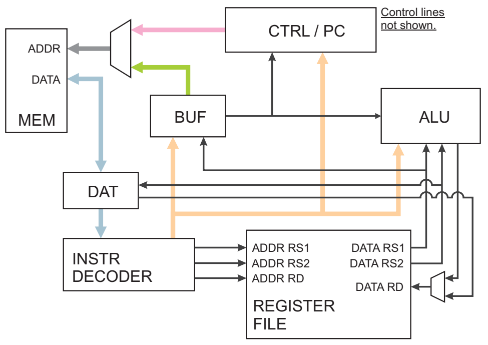


Figure 6.2: Bit-serial RISC-V datapath. Thin lines have width of a single bit.

and random numbers. The RISC-V implementation with no countermeasures (NCM-uP) has direct access to the SRAM memories, while others go through a memory protection unit. Netlist manipulation techniques described in this section are only applied to the BM-uP and BM-DDL-uP microprocessors. Other blocks are synthesized with the typical digital design flow, using a commercial library of standard cells.

6.2.2 Bit-serial RISCV Microprocessor

We used a bit-serial CPU, show in Fig. 6.2. Although not shown in the diagram, memory and register file are shared among the three microprocessors. The CPU is based on a publicly available design[1], which was chosen for its small area footprint. The internal datapath is one bit wide. Techniques described in this work are equally applicable to larger microprocessors, with wider datapaths.

The main memory is single port and uses 32 bit words to store both code and data. The register file (RF) is dual port. One port for writes, another for reads with 2 bit words to avoid an additional read port. Arithmetic logic unit (ALU) operates on a single bit of data per cycle. The fetch-execute-writeback process requires 36 cycles. Instructions need one, or two phases to complete. Two-phase instructions such as loads, stores, and branches can use up to 70 cycles. Only three 32 bit registers exist. BUF is a 32 bit register that holds data between phases, which can be addresses for branches, load/store, or data to be shifted. DAT is another 32 bit register, used for memory load and store operations, where data is shifted in from RS2 during store operations, and shifted out to RD during load operations.

[1]https://github.com/olofk/serv

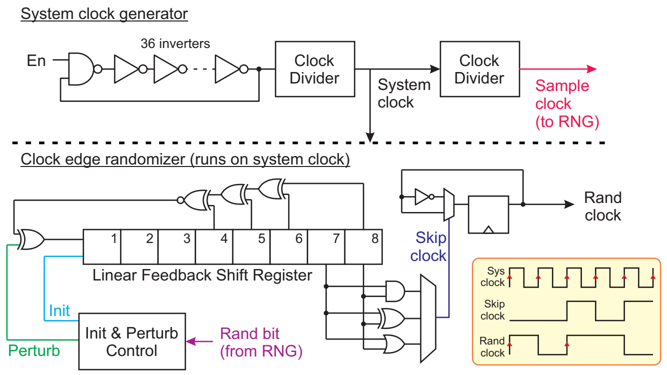


Figure 6.3: Clock generation and randomization logic. Glitch suppression, test, and forbidden state prevention logic are not shown.

The last 32 bit register holds the program counter (PC). If the instruction is not a branch, the next address (PC + 4) is calculated one bit at a time, in parallel with ALU execution. All three 32 bit registers work similarly to a shift register during execution. For memory operations, parallel output/capture functionality is used. The instruction decoder (ID) parses instructions from DAT. It outputs immediate fields to BUF, CTRL/PC, and ALU, one bit at a time, to calculate jumps addresses and arithmetic operations.

Control and status registers (CSRs), interrupts, multiplication, and divide instructions were not implemented. Using 65 MHz clock, an AES encryption of 128 bits, without any software countermeasures, takes nearly 20 ms—substitution boxes (SBOXes) not stored as lookup tables, but computed during execution.

6.2.3 Clock Generation and Randomization

Power analysis attacks require a large number of power traces. As discussed in section 6.3.5, the effectiveness of power attacks is significantly higher if traces are aligned in time. Therefore, the insertion of random delays to purposely misalign power traces may be used as countermeasure. To avoid changing software or existing hardware architectures, we insert random delays by manipulating the clock signal.

Our clock generation and randomization logic is shown in Fig. 6.3. A ring oscillator generates the clock using 36 inverters and a NAND gate. Layout of the ring oscillator was done manually. To generate a random signal for clock edge randomization we used an 8 bit LSFR of maximum sequence length [3]. The frequency on which clock cycles are skipped

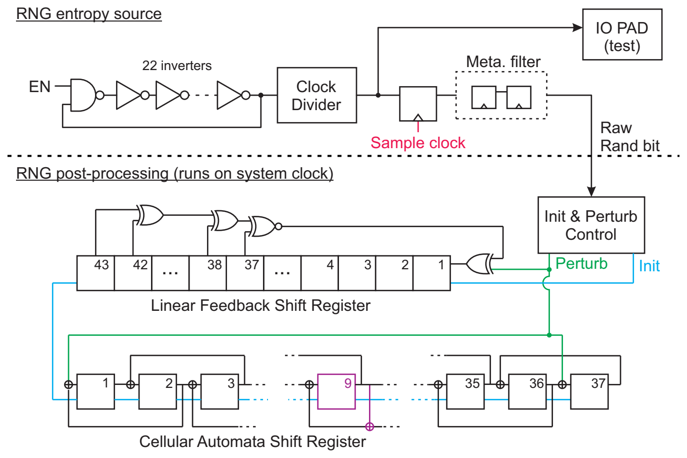


Figure 6.4: Random number generator architecture. The sampling clock comes from the clock generation circuit.

is selected using a multiplexer. The values of 25%, 50%, or 75%, correspond to outputs from AND, XOR, and OR, respectively.

Within a 255 cycles window, the randomized clock will have a constant number of edges. This enables performance predictability with resolution of 255 system clock cycles. Nevertheless, one may argue that a repeating clock randomization pattern introduces potential vulnerabilities. For this reason, we allow the 8 bit LFSR to be perturbed, at a user adjustable rate. We suggest a moderate perturbation rate, for example, when perturbed every 32 cycles, the clock randomizer pattern will follow the 8 bit LFSR natural counting sequence for 31 cycles, until it is perturbed again.

6.2.4 Random Number Generator (RNG)

Random numbers are a critical component of secure integrated circuits. Both clock randomization and Boolean masking require random numbers to work effectively. In particular, Boolean masking needs a large quantity of fresh (new) random numbers every clock cycle for remasking operations. The entropy requirement for Boolean masking however, is not as high as typical cryptographic uses such as key generation. Therefore, we designed a random number generator (RNG) focused on high throughput to attend the demand of our implemented countermeasures.

Our RNG design is shown in Fig. 6.4, it uses thermal noise as entropy source. The ring oscillator has 22 inverters and a NAND gate, it accumulates thermal noise over the sampling period. The number of stages is relatively prime with the clock oscillator to avoid frequency locking, since sampling clock is derived from system clock. The entropy of sampled values is proportional to the sampling period used, which depends on target entropy and technology noise characteristics.

The sampling clock is typically much slower than the system clock. To increase throughput and remove possible entropy source biases, we added a post-processing block. Our solution is based on [101]. We use an 43 bit LFSR [3], and a one-dimensional linear hybrid cellular automata shift register (CASR) of 37 cells [19]. The LFSR/CASR states are initialized with raw random numbers during power on. Both LFSR and CASR update at every system clock cycle, and are perturbed when a new raw random bit is available, which depends on the sample frequency.

Output is derived by XORing LFSR and CASR states. We use an XOR network, shown in Fig. 6.5, that derives 243 outputs from LFSR/CASR state. Up to 1591 outputs are possible using two input XOR gates. More outputs require the XORing more than two state bits. The connection pairs are unique, randomly assigned, and defined at design time (they are static).

6.2.5 Boolean Masking

Boolean masking (BM) splits each value into two or more shares that are (ideally) uncorrelated to the original value. Computation using multiple shares requires careful logic manipulation, where each operation is replaced by its masked counterpart. We applied Boolean masking to the microprocessors designated as BM-uP and BM-DDL-uP, with masking expressions from [17]. Although the masking expressions themselves are secure, further linear combinations can make them insecure [39]. Nevertheless, the used expressions have key characteristics that are very attractive for the construction of *fully-masked* large-scale designs: i) they are compact, and ii) do not require fresh randomness at every operation.

Fig. 6.6 shows the masked implementation of basic non-linear logic gates (AND, OR) using expressions from [17]. Linear operations such as XOR, shifts, and permutations do not require any modification, they are applied to both shares equally. Fresh random numbers are required when converting from single share to two shares. Conversions between 2 shares and single share are only performed in the memory protection module, while data is encrypted by session keys. Therefore, plain-text values are never exposed.

We developed a script that converts arbitrary circuit netlists into their masked counterparts. The script is integrated with EDA tools and is shown in Fig. 6.7. It imposes

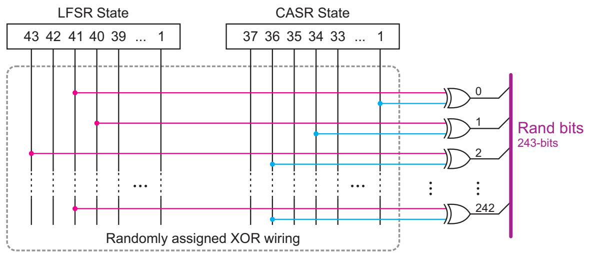


Figure 6.5: Output XOR network for random numbers. XOR wiring is defined randomly at design phase.

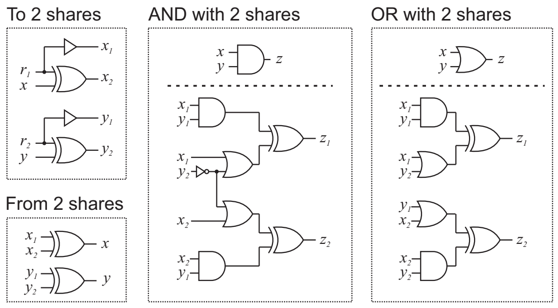


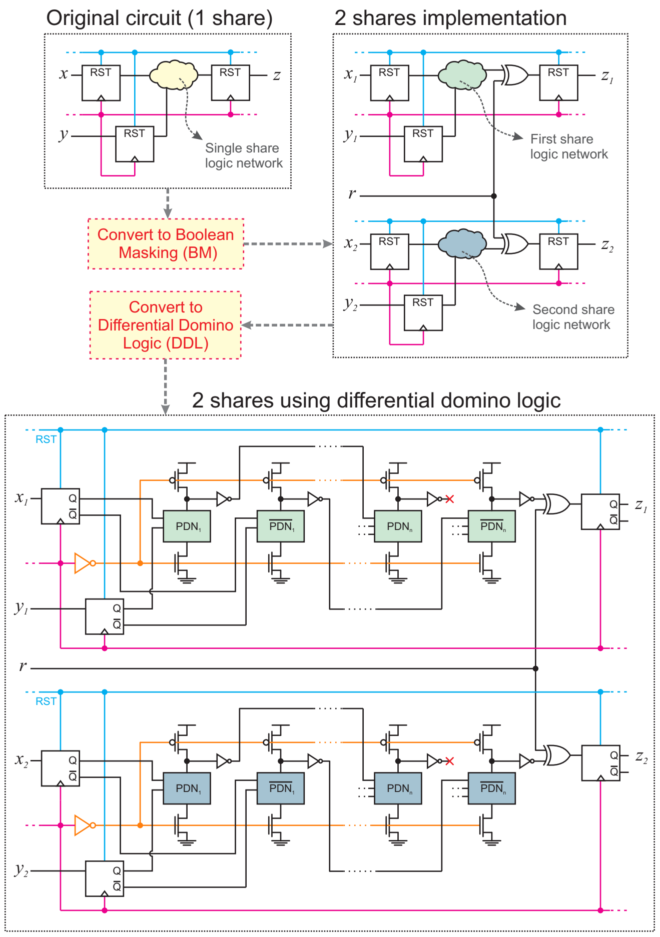
Figure 6.6: Boolean masking expressions to perform non-linear operations on shared values.

only two restrictions: i) the input netlist cannot use clock and reset as data; and ii) reset must be asynchronous. First, the script splits every signal, except reset and clock, in two shares, replacing all basic operations by their masked counterparts. Circuit functionality and latency are not affected. Then, all registers are duplicated and wired to accommodate the additional share of each state. An XOR gate is added before every register to refresh the masking with a new random number every cycle. New top level ports are automatically created to accommodate the additional share of all previously existing input/output ports. A new top level input port bus for fresh random numbers is also created, it is driven by the RNG output network, shown in Fig. 6.5.

6.2.6 Differential Domino Logic

Glitches in the masking of non-linear logic may momentarily expose protected values, reducing the effectiveness of Boolean masking [62]. Traditional solutions use registers to stop glitch propagation, which modifies circuit latency and requires significant design effort to protect preexisting designs. We reduce glitches by implementing our logic gates using differential domino logic (DDL), a differential input/output dynamic logic style with precharge/evaluation phases [74]. During precharge, both outputs are driven to the same value, but only one of them toggles in the evaluation phase. Logic gate inputs are restricted to a single 0 to 1 transition. Fig. 6.8 shows the transistor level schematic of AND, OR, and XOR using DDL, with associated layout. Inverters are implemented by swapping the output wires.

Our netlist conversion script was adapted to use our DDL logic gates instead of the commercial library cells. The output netlist has each wire split in two shares for Boolean

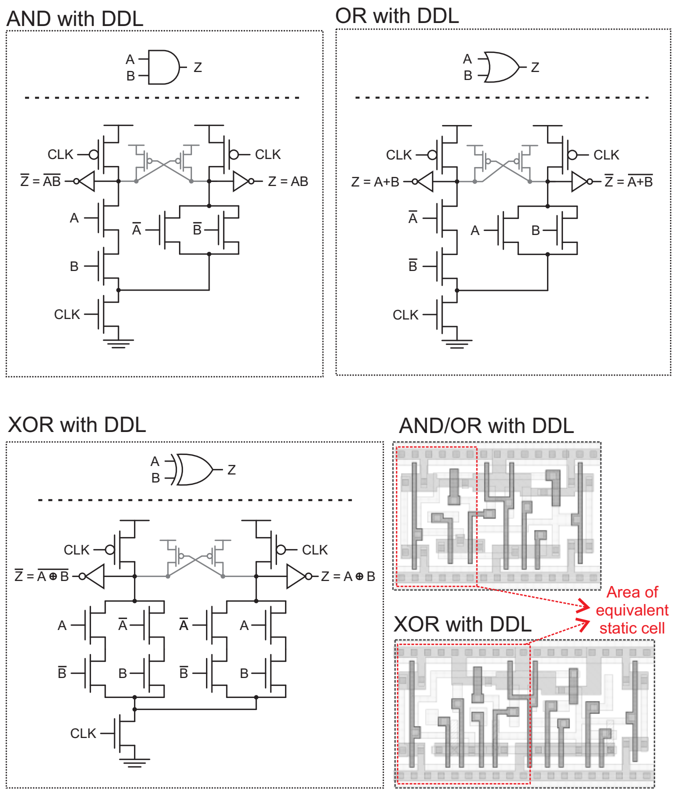


Original circuit (1 share)
2 shares implementation
Single share logic network
First share logic network
Second share logic network
Convert to Boolean Masking (BM)
Convert to Differential Domino Logic (DDL)
2 shares using differential domino logic
RST

Figure 6.7: Netlist conversion from original circuit, to shared, and then dynamic logic.

masking. Moreover, each share is again split in two complementary signals for DDL. This technique was applied to BM-DDL-uP, in Fig. 6.1. The layout of our DDL cells has same height of the commercial cells, so that we can conveniently mix DDL gates with gates from the commercial library. Better area results are possible with taller DDL cells, but interoperability with commercial library cells would be extremely hard.

Our output netlist uses sequential elements from the commercial library. The remasking XOR gate preceding every sequential element also uses a cell from the commercial library—its output has no connection to dynamic cells, and glitches in that gate are unlikely to cause significant information leakage since one of the inputs is a random number. Inverted inputs to DDL gates are wired to registers inverted output pins. Complementary signals were



AND with DDL
OR with DDL
XOR with DDL
AND/OR with DDL
CLK
$\overline{Z} = \overline{AB}$
$Z = AB$
$Z = A+B$
$\overline{Z} = \overline{A+B}$
$\overline{Z} = \overline{A \oplus B}$
$Z = A \oplus B$
Area of equivalent static cell

Figure 6.8: Dynamic domino gates and associated layout.

not granted their own sequential element—the number of registers do not quadruple with respect to the original design. Inverted outputs from the last DDL gate in a combinational path (prior to the remasking XOR) are left unconnected. This is a source of load imbalance that will leak information, but it avoids significant area cost.

We characterized multiple strengths of the DDL cells using a commercial tool. Our custom DDL library has two strengths of the AND/OR gate, and three strengths of the XOR gate. Dynamic logic is not supported by the characterization tool, so we manually specified the relevant input/output timing arcs for delay characterization. We also wrote Verilog models for functional simulation, supporting delay annotation. The Liberty and Verilog files were used for timing analysis and design sign-off.

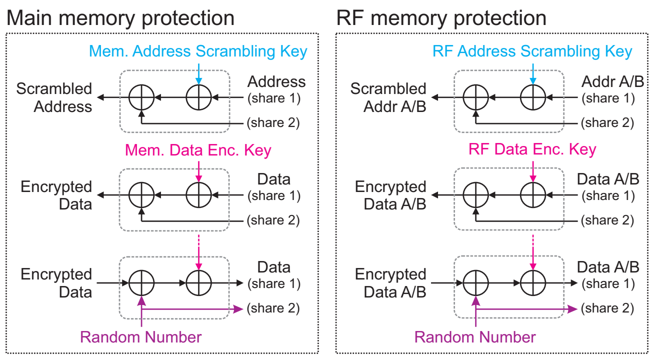


Figure 6.9: Memory protection scheme.

6.2.7 Memory Protection

A trivial solution for memory protection is to duplicate the memory size and store both shares of the Boolean masked data. To avoid a twofold increase in the SRAM size, we perform encryption using a session based key, which is XORed with the data to alter its hamming weight. Moreover, we also scramble the address bus using another session based key that is XORed with the address, changing the memory layout. Session keys have different values at every session. The definition of a session is application dependent, but in our case, it corresponds to one execution of the target program in one of the microprocessors. Session keys remain stable for the duration of a session. Our session keys are provided via test logic. Fig. 6.9 shows the memory protection blocks. The order in which the XOR operations are performed is extremely important to avoid exposing plain-text values.

Single-port memory protection can be improved further. The encrypted data can be XORed with the address before it is written, making data encryption address dependent, but this requires a non-linear expansion of the 10 bit address into a 32 bit word, which has extra area costs. It is also important to notice that our memory protection unit does not include firewall capabilities to detect unauthorized accesses. If present, such logic must be protected with Boolean masking and DDL.

Table 6.1: Area utilization in 65 nm CMOS.

Unit name	**# of Instances**	**Area** (μm^2)
Clock generation & randomization	203	922
Random number generation	674	2735
Microprocessor NCM-uP	731	3509
Microprocessor BM-uP	8232	22967
Microprocessor BM-DDL-uP	7633	44426
Main memory (code & data)	13	39505
Register file	19	10720
Memory protection and muxes	768	2653

Notes: XOR output network uses 242 XOR instances and account for 871-μm^2 of the area reported for the RNG.

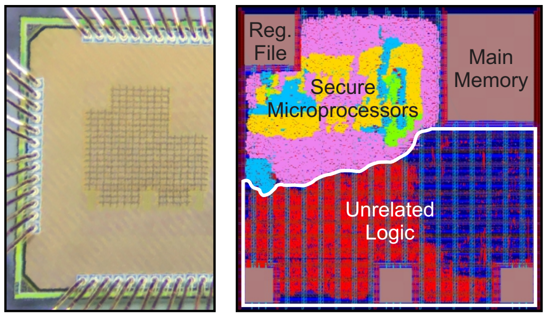


Figure 6.10: Die photo of the implemented chip in 65 nm CMOS technology and its layout. The layout highlights standard cells for NCM-uP (green), BM-up (yellow), and BM-DDL-uP (pink). Test logic, RNG, clock generation/randomization are shown in light blue.

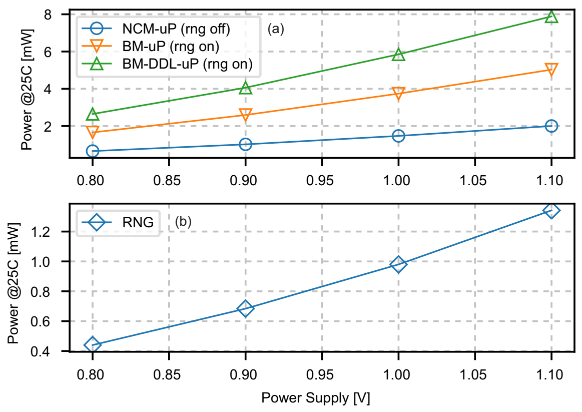


Figure 6.11: Power consumption of (a) RISC-V microprocessors, (b) RNG. Static components are not included.

6.3 Measurement Results

6.3.1 Chip Area Utilization

We fabricated a testchip in a 65 nm CMOS process, shown in Fig. 6.10. Table 6.1 shows the number of instances and area utilization of each block after design sign-off. The clock generation includes a 57-μm^2 RO and all clock edge randomization logic. The RNG includes a 37-μm^2 RO and all post-processing logic (LFSR, CASR, and XOR network). The XOR network uses 242 XOR instances and accounts for 871-μm^2 of the reported area for RNG. The main memory uses a single-port 32 kbit SRAM (1024x32). RF uses a dual-port 1 kbit SRAM (512x2). The BM-uP implementation had an area increase of 6.5x compared to baseline (NCM-uP), while the number of instances increased by 11.3x. The BM-DDL-uP had an area increase of 1.9x compared to BM-uP, but the number of instances showed small reduction due to the absence of inverters in differential logic styles.

6.3.2 Power Consumption

Dynamic power consumption was measured from 0.8V up to 1.1V, for the NCM-uP, BM-uP, BM-DDL-uP, and RNG. Nominal voltage is 1.0 V (see Fig. 6.11 (a) and (b)). We share the same power supply pin with other non-related digital circuits which had their clock disabled during experiments. Moreover, static power was measured and excluded from all reported values. The RNG was not active during measurements of NCM-uP. The power reported for the RNG includes system clock oscillator for sampling, RNG oscillator, LFSR/CASR post-processing, XOR output network, and remasking XOR gates.

6.3.3 Quality of Random Numbers

The National Institute of Standards and Technology (NIST) published tests to assess the quality of random numbers [10]. Table 6.2 reports results for all tests. A total of 100 bitstreams with 1 M bits each was used. All tests meet the suggested passing criterion of 80%. Fig. 6.3 plots the autocorrelation of 1 M (a) raw random bits, and (b) after post-processing by LFSR/CASR logic. We used a sampling frequency of 3.4 MHz. Dashed lines represent the interval that contains 95% of the data. Fig. 6.3 (c) plots the Shannon entropy for 1 M bits at various sampling frequencies. Entropy was calculated using intervals of 5 bits.

Table 6.2: NIST tests on RNG output.

Test Name	ρ	**Proportion**
Frequency	0.554420	98/100
Block Frequency	0.042808	98/100
Cumulative Sums	0.964295	97/100
Runs	0.013569	99/100
Longest Run	0.289667	98/100
Rank	0.249284	98/100
FFT	0.474986	99/100
Non Overlap	0.867692	96/100
Overlap	0.115387	100/100
Universal	0.334538	97/100
Approx Entropy	0.181557	99/100
Rand Exc	0.941144	67/69
Rand Exc Var	0.619772	66/69
Serial	0.224821	98/100
Linear Complexity	0.017912	100/100

Notes: Cumulative Sums, Non Overlapping, Random Excursions, Random Excursions Var, and Serial run multiple times in the NIST SP 800-22 Rev1a. Values in table refers to the worst pass proportion among all runs.

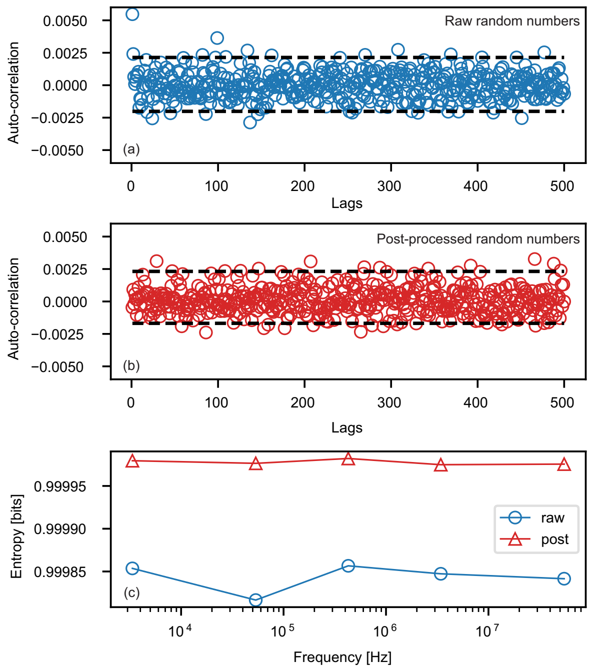


Table 6.3: Auto-correlation test results on (a) raw, and (b) post-processed random numbers. Shannon entropy tests in (c).

Table 6.4: Comparison of RNG results.

	This work	JSSC'16 [63]	JSSC'16 [7]	JSSC'22 [123]
Technology	65 nm	14 nm	65 nm	130 nm
Entropy	Jitter	Meta	Meta+Jitter	Meta
Bit rate (Gb/s)	12.8	0.225	3	0.002
Area (μm^2)	2735	1088	1609	5561
Power (mW)	0.992	1.5	5	-
Energy (fJ/bit)	0.078	6.67	1.67	-
Post-processing	LFSR/CASR	AES	None	VN8W

Notes: Power and energy reported for our work do not include the static component.

Our RNG produces 243 random bits each clock cycle, therefore its total throughput at 56 MHz is 12.8 Gb/s. Further details such as area, power, and energy efficiency are provided in Table 6.4. This lightweight, throughput focused RNG design meets our requirement for Boolean masking operations, but arguably, other sensitive contexts such as key generation, padding, and nonces will require minimum entropy guarantees, slower sampling frequencies, and forward/backward secrecy in the post-processing—which increases the hardware size significantly. We recommend the interested reader to see AIS 31 standard for details [47].

6.3.4 Experimental Setup for Side-channel Analysis

Fig. 6.12 shows our experiment setup for side-channel analysis. We added a 75 Ohms resistor in series with the testchip power supply. The signal is conditioned by a high-speed amplifier (OPA858, 5.5 GHz GBW) in a non-inverting configuration with gain of 7. A 0.84 V reference was connected to the gain resistor to avoid clamping of the output signal. The external chip power supply was set to 1.2 V instead of the nominal 1.0 V, to account for the voltage drop in the series resistor. Our oscilloscope has input bandwidth of 1 GHz. The acquisition system is orchestrated by a desktop computer which controls the oscilloscope using GPIB and communicates to the testchip using an FPGA. The FPGA writes the encrypted software using JTAG. All power traces use unique session keys for memory protection. The sampling rate is 10 GS/s. We used an open source Julia toolbox[2] to perform the correlation power analysis (CPA) with, and without dynamic time warping (DTW) pre-processing. We attack only the first key byte during an AES encryption operation, targeting the last instruction of the SBOX computation routine—an XOR between a register and a constant, where the result is stored into another register. We used a hamming weight leakage model. Oscilloscope trigger is obtained from a testchip output set by the microprocessor software 36 cycles before the sensitive instruction. The number of samples logged in each power trace is adjusted depending on clock edge randomization settings to ensure all relevant clock cycles are captured.

6.3.5 Correlation Power Analysis

Correlation power analysis (CPA) uses a set of power traces to extract secret information from a device. Traces record the device manipulating the secret value with different input data. For example, recorded power traces may include many AES encryptions using the same key but different plaintexts. CPA attacks extract a few key bits in each iteration.

[2]https://github.com/Riscure/Jlsca

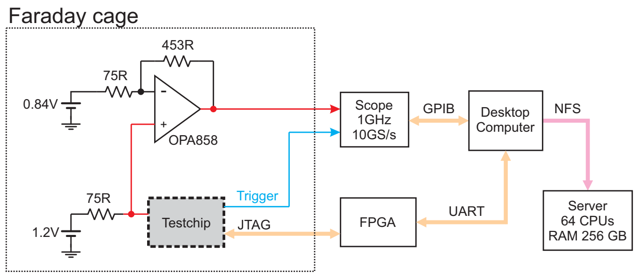


Figure 6.12: Side-channel analysis acquisition setup.

The attacker chooses a key candidate and creates a power consumption hypothesis for each input data, using knowledge of the implemented algorithm, and a leakage model (typically hamming-weight). The correlation between the hypothesis vector and the recorded power traces will be the highest when the correct key is used.

Our side-channel analysis does not include test vector leakage assessment (TVLA) [12]. Such tests do not replace conventional key extraction attacks, but provide a quick alternative to detect potential side-channel problems. Nevertheless, they require saving a long power trace that includes a series of encryptions, which would use an enormous amount of oscilloscope memory due to the low throughput of our hardware architecture. Future work shall use a faster architecture so that we can perform these tests.

Fig. 6.13 shows CPA results, plotting the correlation versus number of traces used in the attack. Each plot has 256 lines, one for each key candidate. The correct key (red) is exposed when its correlation is the highest. In Fig. 6.13 (a), the baseline implementation with no countermeasures (NCM-uP) had its key exposed with 375 power traces. Results with Boolean masking are shown in Fig. 6.13 (b), where the key is exposed with 6200 power traces. If Boolean masking is combined with the dynamic logic implementation, Fig. 6.13 (c), it takes 375 k power traces, 1000x compared to the baseline, to expose the key.

Next, we investigated effectiveness of the clock randomization on the baseline microprocessor (NCM-uP). We tested three types of clock edge randomization. In Fig. 6.13 (d) 25% of clock edges were skipped, and it took 850 k power traces to expose the key. The required number of traces to expose the key increased dramatically to 6.2 M, and 9.1 M, as we skipped 50% and 75% of the clock edges, as respectively shown in Fig. 6.13 (e) and (f).

Pre-processing techniques such as dynamic time warping (DTW) may be used to align power traces before a CPA attack [81]. The DTW algorithm originated from speech recognition systems to match spoken words to a database containing prerecorded words with

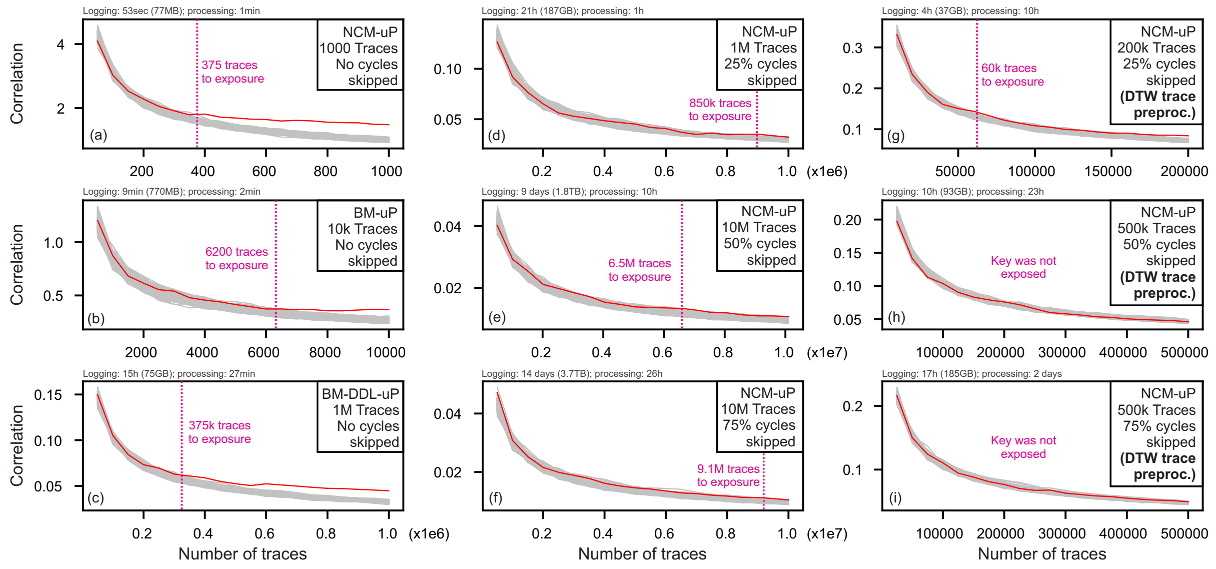


Figure 6.13: CPA coefficient versus number of traces for different sets of countermeasures. Results without clock edge randomization for (a) NCM-uP; (b) BM-uP, and (c) BM-DDL-uP; results for original netlist (NCM-uP) using clock edge randomization of (d) 25%, (e) 50%, and (f) 75%; and results using DTW pre-processing for trace alignment on NCM-uP with clock edge randomization of (g) 25%, (h) 50%, and (i) 75%.

different timing. We use a FastDTW variant which has complexity $O(Tk)$, where T is number of traces, and k is trace length [106]. In our experiments with DTW pre-processing, the radius parameter was set to 90. Larger radius enhance representation accuracy of the original DTW algorithm, but significantly increase computing time and memory usage.

Fig. 6.13 (g), (h), and (i) show the CPA attack results with DTW pre-processing on the baseline microprocessor (NCM-uP), with clock randomization enabled. In Fig. 6.13 (g), with 25% of clock cycles skipped, the key was exposed with only 60 k traces, which is 14.2x fewer traces compared to CPA without DTW pre-processing, as shown in Fig. 6.13 (d). However, it takes nearly 10x longer computation time. Similarly, Fig. 6.13 (h), and (i), with 50% and 75% of clock cycles skipped, show that the key was not exposed after a 500 k traces CPA attack using DTW pre-processing, as opposed to the 6.5 M and 9.1 M traces required to expose the key without DTW.

We also assess the information leakage of all countermeasures combined. For that, we skip 50% of clock cycles at the BM-DDL-uP microprocessor, which implements both Boolean masking and DDL. Fig. 6.14 shows that CPA attacks using (a) 20 M traces without pre-processing, and (b) 2 M traces with DTW, were not enough to expose the key.

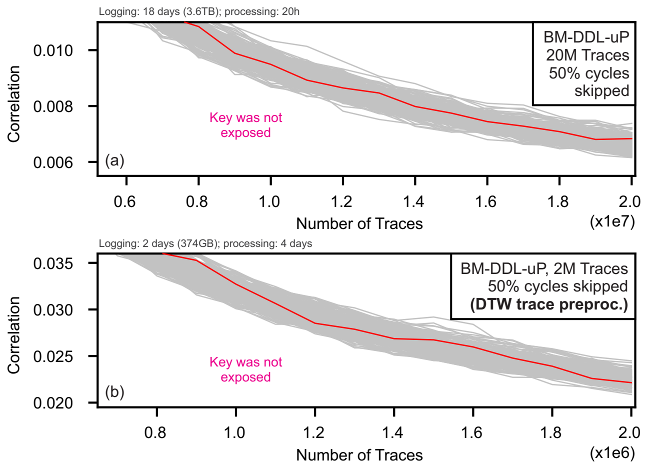


Figure 6.14: CPA coefficient versus number of traces for BM-DDL-uP with clock edge randomization of 50%. Results (a) without DTW trace pre-processing and (b) with DTW trace pre-processing.

6.3.6 Comparison with Prior Work

There are several implementations of Boolean masked microprocessors in the literature [25, 37,71]. Table 6.5 provides a comparison of our work with previous publications with respect to several key security features listed in the first column. It is clear from Table 6.5 that our work covers a broad range of techniques and components necessary to implement a secure microprocessor.

Unlike other works, we used a CPU implementation with datapath size of 1 bit. Such decision was made due to area constraints in our testchip, but our methodology can be applied to any digital design, of any size. In fact, our RTL and implementation scripts are publicly available to interested researchers [86]. However, comparing the effectiveness of countermeasures from different publications is not trivial. Changes in the acquisition system, and the presence of combined countermeasures make it hard to draw any definitive conclusions. It is also important to mention that the effectiveness of our countermeasures will likely increase when applied to larger designs. In addition to higher switching noise, the sensitive signals of larger designs will be relatively weaker, compared to the total power consumption, requiring attackers to collect more power traces.

Table 6.5: Comparison with other secure microprocessors.

	This work			**CHES'07** [71]	**CARDIS'16** [37]	**DAC'19** [25]
Technology	65 nm	65 nm	65 nm	130 nm	FPGA	FPGA
Architecture	RISC-V	RISC-V	RISC-V	8051	RISC-V	RISC-V
Datapath	1 bit	1 bit	1 bit	8 bit	32 bit	32 bit
Clock rand	No	No	Yes (50%)	No	No	No
Entropy source	Jitter	Jitter	Jitter	No	No	No
PRNG	Yes	Yes	Yes	Not avail	Not avail	Not avail
Rand bits/cycle	243	243	243	1	–	–
Mem addr Scr	Yes	Yes	Yes	No	No	No
Mem data enc	Yes	Yes	Yes	No	No	Yes
Masking	[17]	[17]	[17]	MDPL [72]	DOM [38]	TI [67]*
Fully masked	Yes	Yes	Yes	Yes*	No	Yes*
Pre-charge logic	No	DDL	DDL	MDPL [72]	No	No
Methodology	Auto.	Auto.	Auto.	Not avail	Manual	Not avail
Traces to discl.	375	375	375	5 k	–	15 k
Max Traces	6.2 k	375 k	20 M	300 k	100 M	3 M
Attack types	CPA	CPA	CPA/DTW	DPA	TVLA	TVLA
Open-source	Yes [86]	Yes [86]	Yes [86]	No	No	No

Notes: (*) few details were provided in the publication.

6.4 Conclusion

We demonstrated a bit-serial RISC-V microprocessor implementation with no plain-text data in the clear. Our design uses Boolean masking at the logic level, and dynamic domino logic at the transistor level. We selected a set of countermeasures that require no changes to the input RTL code. Unlike previous literature, our methodology is fully integrated with EDA tools, and can be applied to digital designs of any size or complexity. We also provided details on other key components of secure ICs, such as clock randomizer, memory protection, and random number generator. The random numbers generated with our RNG pass on all NIST tests. Side-channel analysis on the baseline implementation extracted the AES key using only 375 traces, while our secure microprocessor was able to withstand attacks using 20 M traces.

Chapter 7

Conclusion

PUFs harvest process variation to create a unique and unclonable identification of an IC, but learning attacks are able to impersonate PUFs after seeing a subset of CRPs. We studied circuits and architectures to make PUFs more resilient to learning attacks. Our approaches included low voltage operation, non-monotonic quantization, composite architectures, and challenge obfuscation.

Our first initiative was to explore low voltage operation of APUFs, under the hypothesis that PUFs could benefit from increased sensitivity to process variations, and produce more stable responses. The results of this attempt were not included in this thesis, but are published in [87]. We found that running APUFs at lower supply voltages offers modest improvements in reliability, with reasons possibly related to larger noise components. We also learned that SPICE simulations fail to predict the stability of responses. We recommend the interested reader to see [87] for details.

Next, we pursued the development of an architecture where responses not always encode the best performing circuit structure. Our hypothesis was that APUF responses immediately expose which delay path is faster, giving learning algorithms valuable information about the PUF entropy source. Therefore, we introduced the concept of non-monotonic response quantization, and demonstrated a ring-oscillator based strong PUF that implements the idea. Our silicon measurements showed a significant improvement in learning resistance, with bit error rate smaller than 13.4% for the temperature range from 0 °C to 50 °C. The reported results motivate continued development of the architecture, possibly with circuit techniques that could yield better temperature stability. Other contributions of this work include a technique to quickly estimate, and visualize the learnability of strong PUFs.

Looking for alternatives to design learning resistant strong PUFs, we investigated composite PUFs. In particular, we explored the design space of the POP architecture to better understand the trade-off between security and response stability. We designed a testchip with POP implementations of various sizes. We also added temporal majority voting for better response stability, and multiple evaluation rounds to investigate the possible security improvements of an increased architectural depth. Our security assessment was done with DNNs, and it produced unintuitive results. While our DNN experiments report increased security when using APUFs with 6, or more stages in the first layer, smaller APUFs sizes were shown vulnerable to DNN attacks. Moreover, they showed a trend of higher prediction accuracy as the number of rounds increased. To carefully examine such results, we extended previous techniques of influential bits to assess stage bias in APUF instances. Our conclusion is that small APUFs in the first layer limit the challenge space of the second layer APUF, therefore, compositions not always preserve security properties of PUFs.

The trade-off between response stability and security dictates what is feasible in the design of strong PUFs. Following a recent trend of combining weak and strong PUFs in a single design to overcome aforementioned problems, we investigated key based challenge obfuscation. Our proposed architecture uses NLFSRs to manipulate the external challenge prior to evaluation. Responses are directly provided to the user, without error correction or extra post-processing steps. Security was assessed using strict avalanche criterion, and DNN attacks. We highlighted the fact that secret key storage can be achieved with a weak PUF, or an OTP flash, since unclonability properties are already provided by the underlying strong PUF. We also noted that key based challenge obfuscation mechanisms are susceptible to key extraction attacks that use power consumption information. We detailed countermeasures to protect our architecture against such attacks, including Boolean masking and clock randomization. Compared to the baseline APUF implementation, the cost increase of our proposed architecture is 1.27x, and 2.2x when using clock randomization, and when clock randomization is combined with Boolean masking, respectively.

PUFs establish trust between the user and the hardware, but internet connected applications demand mechanisms for message integrity, confidentiality, and (often) non-repudiation—which require on-chip storage of a secret key. Even if the key is produced by a PUF, it will be subject to key extraction attacks that use power consumption information. Secure integrated circuits must address power analysis attacks with appropriate countermeasures. Traditional countermeasures have limited scope of protection, and impose several restrictions on how sensitive data must be manipulated. We demonstrated a bit-serial RISC-V microprocessor implementation with no plain-text data. All values are protected using Boolean masking and differential domino logic. Software can run with little to no countermeasures, reducing code size and performance overheads.

Unlike previous literature, our methodology is fully automated and can be applied to designs of arbitrary size or complexity. We also provided details on other system components such as clock randomizer, memory protection, and random number generator. Using measurements from a testchip fabricated in 65 nm technology, we confirmed the quality of random numbers using NIST tests, and also performed extensive power analysis attacks with a total of more than 40 M traces. The baseline implementation had its key extracted using only 375 traces, while our secure microprocessor was able to withstand attacks using 20 M traces.

Future work may address the aging effects in CMOS circuits and how they affect PUF bit error rate. Other possibilities include the study of challenge selection aided by the metadata produced by temporal majority voting circuitry in PUF-on-PUF implementations. With respect to the secure RISC-V microprocessor, there is space for further investigation on the trade-off between clock randomizer perturbation and performance predictability.

References

[1] A Aghaie and A Moradi. TI-PUF: Toward side-channel resistant physical unclonable functions. *TIFS*, 15:3470–3481, 2020.

[2] J Agustin and ML Lopez-Vallejo. A temperature-independent PUF with a configurable duty cycle of CMOS ring oscillators. In *Intl. Symp. on Circuits and Systems*, pages 2471–2474. IEEE, 2016.

[3] P Alfke. Efficient shift registers, LFSR counters, and long pseudo random sequence generators, 1996.

[4] R Anderson and M Kuhn. Tamper resistance-a cautionary note. In *USENIX Ws. on Electronic Commerce*, volume 2, pages 1–11, 1996.

[5] Semiconductor Industry Association. Detecting and removing counterfeit semiconductors in the U.S. supply chain. https://www.semiconductors.org/wp-content/uploads/2018/06/ACTF-Whitepaper-Counterfeit-One-Pager-Final.pdf, 2011. Accessed on July 2022.

[6] MJ Azhar, F Amsaad, and S Köse. Duty-cycle-based controlled physical unclonable function. *TVLSI*, 26(9):1647–1658, 2018.

[7] S-G Bae, Y Kim, Y Park, and C Kim. 3-Gb/s high-speed true random number generator using common-mode operating comparator and sampling uncertainty of d flip-flop. *JSSC*, 52(2):605–610, 2016.

[8] C Bai, X Zou, and K Dai. A novel thyristor-based silicon physical unclonable function. *TVLSI*, 24(1):290–300, 2015.

[9] A Barenghi, L Breveglieri, I Koren, and D Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, 2012.

[10] LE Bassham III, AL Rukhin, J Soto, JR Nechvatal, ME Smid, EB Barker, SD Leigh, M Levenson, M Vangel, DL Banks, and Others. SP 800-22 Rev. 1a. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010.

[11] M Baudet, D Lubicz, J Micolod, and A Tassiaux. On the security of oscillator-based random number generators. *Journal of Cryptology*, 24(2):398–425, 2011.

[12] G Becker, J Cooper, E DeMulder, G Goodwill, J Jaffe, G Kenworthy, T Kouzminov, A Leiserson, M Marson, P Rohatgiand, and S Saab. Test vector leakage assessment (TVLA) methodology in practice, 2013.

[13] GT Becker. The gap between promise and reality: On the insecurity of XOR arbiter PUFs. In *CHES*, pages 535–555. IACR, 2015.

[14] GT Becker. On the pitfalls of using arbiter-pufs as building blocks. *TCAD*, 34(8):1295–1307, 2015.

[15] N Beringuier-Boher, K Gomina, D Hely, J-B Rigaud, V Beroulle, A Tria, J Damiens, P Gendrier, and P Candelier. Voltage glitch attacks on mixed-signal systems. In *Euromicro Conf. on Digital System Design*, pages 379–386. IEEE, 2014.

[16] E Biham and A Shamir. *Differential cryptanalysis of the data encryption standard*. Springer Science & Business Media, 2012.

[17] A Biryukov, D Dinu, YL Corre, and A Udovenko. Optimal first-order boolean masking for embedded IoT devices. In *CARDIS*, pages 22–41. Springer, 2017.

[18] D Boneh, RA DeMillo, and RJ Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of cryptology*, 14(2):101–119, 2001.

[19] K Cattell and S Zhang. Minimal cost one-dimensional linear hybrid cellular automata of degree through 500. *JET*, 6(2):255–258, 1995.

[20] Q Chen, G Csaba, P Lugli, U Schlichtmann, and U Rührmair. The bistable ring PUF: A new architecture for strong physical unclonable functions. In *HOST*, pages 134–141. IEEE, 2011.

[21] J-M Cioranesco, J-L Danger, T Graba, S Guilley, Y Mathieu, D Naccache, and XT Ngo. Cryptographically secure shields. In *HOST*, pages 25–31. IEEE, 2014.

[22] LT Clark, SB Medapuram, and DK Kadiyala. SRAM circuits for true random number generation using intrinsic bit instability. *TVLSI*, 26(10):2027–2037, 2018.

[23] F Courbon, S Skorobogatov, and C Woods. Reverse engineering flash EEPROM memories using scanning electron microscopy. In *CARDIS*, pages 57–72. Springer, 2016.

[24] P Dabrowski, G Labuzek, T Rachwalik, and J Szmidt. Searching for nonlinear feedback shift registers with parallel computing. *Information Processing Letters*, 114(5):268–272, 2014.

[25] E De Mulder, S Gummalla, and M Hutter. Protecting RISC-V against side-channel attacks. In *DAC*, pages 1–4. IEEE, 2019.

[26] J Delvaux and I Verbauwhede. Key-recovery attacks on various RO PUF constructions via helper data manipulation. In *DATE*, pages 1–6. IEEE, 2014.

[27] A Dubey, R Cammarota, and A Aysu. BoMaNet: Boolean masking of an entire neural network. In *Intl. Conf. On Computer Aided Design*, pages 1–9. IEEE, 2020.

[28] D El-Baze, J-B Rigaud, and P Maurine. A fully-digital EM pulse detector. In *DATE*, pages 439–444. IEEE, 2016.

[29] F Ganji, S Tajik, F Fäßler, and J-P Seifert. Strong machine learning attack against pufs with no mathematical model. In *CHES*, pages 391–411. Springer, 2016.

[30] F Ganji, S Tajik, and J-P Seifert. Let me prove it to you: Ro pufs are provably learnable. In *Information Security and Cryptology*, pages 345–358. Springer, 2015.

[31] F Ganji, S Tajik, and J-P Seifert. A fourier analysis based attack against physically unclonable functions. In *Financial Cryptography and Data Security*, pages 310–328. Springer, 2018.

[32] B Gassend, D Clarke, M Van Dijk, and S Devadas. Controlled physical random functions. In *Computer Security Applications Conference*, pages 149–160. IEEE, 2002.

[33] B Gassend, D Clarke, M Van Dijk, and S Devadas. Silicon physical random functions. In *Conf. on Computer and Communications Security*, pages 148–160. ACM, 2002.

[34] L Goubin and J Patarin. DES and differential power analysis the “duplication” method. In *CHES*, pages 158–172. Springer, 1999.

[35] SN Graybeal and PB McFate. Getting out of the STARTing block. *Scientific American*, 261(6):61–67, 1989.

[36] H Gross. Sharing is caring—on the protection of arithmetic logic units against passive physical attacks. In *Intl. Ws. on Radio Frequency Identification: Security and Privacy Issues*, pages 68–84. Springer, 2015.

[37] H Gross, M Jelinek, S Mangard, T Unterluggauer, and M Werner. Concealing secrets in embedded processors designs. In *CARDIS*, pages 89–104. Springer, 2016.

[38] H Groß, S Mangard, and T Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. *Cryptology ePrint Archive*, 2016.

[39] H Gross, K Stoffelen, L De Meyer, M Krenn, and S Mangard. First-order masking with only two random bits. In *Ws. on Theory of Implementation Security*, pages 10–23. ACM, 2019.

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

[41] DE Holcomb, WP Burleson, and K Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *TC*, 58(9):1198–1210, 2008.

[42] DE Holcomb, WP Burleson, K Fu, et al. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Conf. on RFID Security*, volume 7-2, 2007.

[43] J Huening, P Joshi, S Zhao, W-H Chuang, T Tong, and Z Ma. E-beam probing: A high-resolution technique to read volatile logic and memory arrays on advanced technology nodes. In *PAINE*, pages 1–6. IEEE, 2021.

[44] M Karpovsky, KJ Kulikowski, and A Taubin. Differential fault analysis attack resistant architectures for the advanced encryption standard. In *CARDIS*, pages 177–192. Springer, 2004.

[45] M Khalafalla, MA Elmohr, and C Gebotys. Going deep: Using deep learning techniques with simplified mathematical models against XOR BR and TBR PUFs (attacks and countermeasures). In *HOST*, pages 80–90. IEEE, 2020.

[46] M Khalafalla and C Gebotys. PUFs deep attacks: Enhanced modeling attacks using deep learning techniques to break the security of double arbiter PUFs. In *DATE*, pages 204–209, 2019.

[47] W Killmann and W Schindler. A proposal for: Functionality classes for random number generators, Version 2.0, 2011.

[48] P Kocher, J Jaffe, and B Jun. Differential power analysis. In *Annual Intl. Cryptology Conf.*, pages 388–397. Springer, 1999.

[49] O Kömmerling and MG Kuhn. Design principles for tamper-resistant smartcard processors. *USENIX Ws. on Smartcard Technology*, 99:9–20, 1999.

[50] Y-C Lai, C-Y Yao, S-H Yang, Y-W Wu, and T-T Liu. A robust area-efficient physically unclonable function with high machine learning attack resilience in 28-nm CMOS. *TCAS*, 2021.

[51] PA Layman, S Chaudhry, JG Norman, and JR Thomson. Electronic fingerprinting of semiconductor integrated circuits, 2004. US Patent 6,738,294.

[52] JW Lee, D Lim, B Gassend, GE Suh, M Van Dijk, and S Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *VLSI Symp.*, pages 176–179. IEEE, 2004.

[53] H Li, Z Xu, G Taylor, C Studer, and T Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[54] Y Li, J Shen, W Liu, and W Zou. A survey on side-channel attacks of strong PUF. In *ICAIS*, pages 74–85. Springer, 2020.

[55] D Lim. Extracting secret keys from integrated circuits. In *Thesis*. MIT, 2004.

[56] Q Ma, C Gu, N Hanley, C Wang, W Liu, and M O'Neill. A machine learning attack resistant multi-PUF design on FPGA. In *ASP-DAC*, pages 97–104. IEEE, 2018.

[57] T Machida, D Yamamoto, M Iwamoto, and K Sakiyama. A new mode of operation for arbiter puf to improve uniqueness on FPGA. In *Federated Conference on Computer Science and Information Systems*, pages 871–878. IEEE, 2014.

[58] A Maiti, I Kim, and P Schaumont. A robust physical unclonable function with enhanced challenge-response set. *TIFS*, 7(1):333–345, 2011.

[59] M Majzoobi, F Koushanfar, and M Potkonjak. Lightweight secure PUFs. In *ICCAD*, pages 670–673. IEEE, 2008.

[60] M Majzoobi, F Koushanfar, and M Potkonjak. Testing techniques for hardware security. In *Intl. Test Conf.*, pages 1–10, 2008.

[61] S Mangard, E Oswald, and T Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer, 2008.

[62] S Mangard, T Popp, and BM Gammel. Side-channel leakage of masked CMOS gates. In *Cryptographers Track at the RSA Conf.*, pages 351–365. Springer, 2005.

[63] SK Mathew, D Johnston, S Satpathy, V Suresh, P Newman, MA Anders, H Kaul, A Agarwal, SK Hsu, G Chen, et al. μRNG: a 300–950 mV, 323 Gbps/W all-digital full-entropy true random number generator in 14 nm FinFET CMOS. *JSSC*, 51(7):1695–1704, 2016.

[64] PH Nguyen, DP Sahoo, RS Chakraborty, and D Mukhopadhyay. Efficient attacks on robust ring oscillator PUF with enhanced challenge-response set. In *DATE*, pages 641–646, 2015.

[65] PH Nguyen, DP Sahoo, RS Chakraborty, and D Mukhopadhyay. Security analysis of arbiter PUF and its lightweight compositions under predictability test [Revised Version]. *TODAES*, 22(2):1–28, 2016.

[66] PH Nguyen, DP Sahoo, C Jin, K Mahmood, U Rührmair, and M van Dijk. The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks. *TCHES*, pages 243–290, 2019.

[67] S Nikova, C Rechberger, and V Rijmen. Threshold implementations against side-channel attacks and glitches. In *Intl. Conf. on Information and Communications Security*, pages 529–545. Springer, 2006.

[68] U.S. Senate Committee on Armed Services. Inquiry into counterfeit electronic parts in the department of defence supply chain. https://www.armed-services.senate.gov/imo/media/doc/Counterfeit-Electronic-Parts.pdf, May 2012. Accessed on July 2022.

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

[70] M Pecht and S Tiku. Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics. *IEEE spectrum*, 43(5):37–46, 2006.

[71] T Popp, M Kirschbaum, T Zefferer, and S Mangard. Evaluation of the masked logic style MDPL on a prototype chip. In *CHES*, pages 81–94, 2007.

[72] T Popp and S Mangard. Masked dual-rail pre-charge logic: DPA-resistance without routing constraints. In *CHES*, pages 172–186, 2005.

[73] J-J Quisquater and D Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *CARDIS*, pages 200–210. Springer, 2001.

[74] JM Rabaey, AP Chandrakasan, and B Nikolic. *Digital integrated circuits*, volume 2. Prentice Hall Englewood Cliffs, 2002.

[75] M Robshaw and O Billet. *New stream cipher designs: the eSTREAM finalists*, volume 4986. Springer, 2008.

[76] U Rührmair, J Sölter, F Sehnke, X Xu, A Mahmoud, V Stoyanova, G Dror, J Schmidhuber, W Burleson, and S Devadas. PUF modeling attacks on simulated and silicon data. *TIFS*, 8(11):1876–1891, 2013.

[77] U Rührmair, X Xu, J Sölter, A Mahmoud, M Majzoobi, F Koushanfar, and W Burleson. Efficient power and timing side channels for physical unclonable functions. In *CHES*, pages 476–492. Springer, 2014.

[78] DP Sahoo, D Mukhopadhyay, RS Chakraborty, and PH Nguyen. A multiplexer-based arbiter PUF composition with enhanced reliability and security. *TC*, 67(3):403–417, 2017.

[79] DP Sahoo, PH Nguyen, D Mukhopadhyay, and RS Chakraborty. A case of lightweight PUF constructions: Cryptanalysis and machine learning attacks. *TCAD*, 34(8):1334–1343, 2015.

[80] DP Sahoo, S Saha, D Mukhopadhyay, RS Chakraborty, and H Kapoor. Composite puf: A new design paradigm for physically unclonable functions on fpga. In *HOST*, pages 50–55. IEEE, 2014.

[81] S Salvador and P Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[82] P Santikellur, A Bhattacharyay, and RS Chakraborty. Deep learning based model building attacks on arbiter PUF compositions. *Cryptology ePrint Archive*, 2019.

[83] J-M Schmidt and M Hutter. Optical and em fault-attacks on CRT-based RSA: Concrete results. In *Austrian Ws. Microelectronics*, pages 61–67, 2007.

[84] D Schuster and R Hesselbarth. Evaluation of bistable ring pufs using single layer neural networks. In *TRUST*, pages 101–109. Springer, 2014.

[85] W Stahnke. Primitive binary polynomials. *Mathematics of computation*, 27(124):977–980, 1973.

[86] K Stangherlin and M Sachdev. Design and implementation of a secure RISC-V microprocessor (code). https://github.com/cdrlabs-waterloo/2022-07-15.

[87] K Stangherlin and M Sachdev. Reliable strong puf enrollment and operation with temperature and voltage optimization. In *ISQED*, pages 529–534. IEEE, 2021.

[88] K Stangherlin and M Sachdev. Design and implementation of a secure risc-v microprocessor. *arXiv preprint arXiv:2205.05095*, 2022.

[89] K Stangherlin, Z Wu, H Patel, and M Sachdev. Secure and lightweight challenge obfuscation with keyed non-linear feedback shift register (RTL code). https://github.com/cdrlabs-waterloo/2022-07-22.

[90] K Stangherlin, Z Wu, H Patel, and M Sachdev. Design exploration and security assessment of puf-on-puf implementations. *arXiv preprint arXiv:2206.11840*, 2022.

[91] K Stangherlin, Z Wu, H Patel, and M Sachdev. Enhancing strong PUF security with non-monotonic response quantization. *arXiv preprint arXiv:2206.03440*, 2022.

[92] K Stangherlin, Z Wu, H Patel, and M Sachdev. High-level model for NMQ-RO strong PUF. https://doi.org/10.24433/CO.2685736.v3, 5 2022.

[93] K Stangherlin, Z Wu, H Patel, and M Sachdev. Secure and lightweight strong puf challenge obfuscation with keyed non-linear fsr. *arXiv preprint arXiv:2207.11181*, 2022.

[94] K Stangherlin, Z Wu, H Patel, and M Sachdev. Testchip measured CRPs of NMQ strong PUF. https://dx.doi.org/10.21227/y8e7-m164, 5 2022.

[95] E Strieder, C Frisch, and M Pehl. Machine learning of physical unclonable functions using helper data: Revealing a pitfall in the fuzzy commitment scheme. *TCHES*, pages 1–36, 2021.

[96] Y Su, J Holleman, and B Otis. A 1.6 pj/bit 96% stable chip-ID generating circuit using process variations. In *ISSCC*, pages 406–611. IEEE, 2007.

[97] GE Suh and S Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14. IEEE, 2007.

[98] V Suresh, R Kumar, M Anders, H Kaul, V De, and S Mathew. A 0.26% BER, 10 28 challenge-response machine-learning resistant strong-PUF in 14nm CMOS featuring stability-aware adversarial challenge selection. In *VLSI Symp.*, pages 1–2. IEEE, 2020.

[99] K Tiri, M Akmal, and I Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *E-SSCC*, pages 403–406. IEEE, 2002.

[100] K Tiri and I Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *DATE*, volume 1, pages 246–251. IEEE, 2004.

[101] TE Tkacik. A hardware random number generator. In *CHES*, pages 450–453. Springer, 2002.

[102] J Tobisch, A Aghaie, and GT Becker. Combining optimization objectives: New modeling attacks on strong PUFs. *TCHES*, pages 357–389, 2021.

[103] T Tong, HJ Ryu, Y Wang, W-H Chuang, J Huening, P Joshi, and Z Ma. Electron beam probing of active advanced FinFET circuit with fin level resolution. In *ISTFA*, pages 345–348. ASM, 2018.

[104] J Van den Herrewegen, D Oswald, FD Garcia, and Q Temeiza. Fill your boots: Enhanced embedded bootloader exploits via fault injection and binary analysis. *TCHES*, pages 56–81, 2021.

[105] Jasper GJ Van W, MF Witteman, and F Menarini. Practical optical fault injection on secure microcontrollers. In *FDTC*, pages 91–99. IEEE, 2011.

[106] JGJ van Woudenberg, MF Witteman, and B Bakker. Improving differential power analysis by elastic alignment. In *Cryptographers Track at the RSA Conf.*, pages 104–119. Springer, 2011.

[107] EI Vatajelu, G Di Natale, MS Mispan, and B Halak. On the encryption of the challenge in physically unclonable functions. In *IOLTS*, pages 115–120. IEEE, 2019.

[108] A Venkatesh, AB Venkatasubramaniyan, X Xi, and A Sanyal. 0.3 pj/bit machine learning resistant strong PUF using subthreshold voltage divider array. *TCAS II*, 67(8):1394–1398, 2019.

[109] H Wang, D Forte, MM Tehranipoor, and Q Shi. Probing attacks on integrated circuits: Challenges and research opportunities. *Design & Test*, 34(5):63–71, 2017.

[110] Q Wang, M Gao, and G Qu. A machine learning attack resistant dual-mode PUF. In *GLSVLSI*, pages 177–182, 2018.

[111] W-C Wang, Y Yona, Y Wu, SN Diggavi, and P Gupta. Slate: a secure lightweight entity authentication hardware primitive. *TIFS*, 15:276–285, 2019.

[112] Y Wang, X Xi, and M Orshansky. Lattice PUF: A strong physical unclonable function provably secure against machine learning attacks. In *HOST*, pages 273–283. IEEE, 2020.

[113] AF Webster and SE Tavares. On the design of S-boxes. In *Conf. on the theory and application of cryptographic techniques*, pages 523–534. Springer, 1985.

[114] PJ Windley. *Digital Identity: Unmasking identity management architecture (IMA)*. O'Reilly, 2005.

[115] N Wisiol, C Mühl, N Pirnay, PH Nguyen, M Margraf, J-P Seifert, M van Dijk, and U Rührmair. Splitting the interpose PUF: A novel modeling attack strategy. *TCHES*, pages 97–120, 2020.

[116] Z Wu, H Patel, M Sachdev, and MV Tripunitara. Strengthening PUFs using composition. In *ICCAD*, pages 1–8. IEEE, 2019.

[117] D Yamamoto, M Takenaka, K Sakiyama, and N Torii. Security evaluation of bistable ring PUFs on FPGAs using differential and linear analysis. In *FedCSIS*, pages 911–918. IEEE, 2014.

[118] K Yang, D Blaauw, and D Sylvester. An all-digital edge racing true random number generator robust against PVT variations. *JSSC*, 51(4):1022–1031, 2016.

[119] K Yang, Q Dong, D Blaauw, and D Sylvester. A physically unclonable function with BER$< 10^{-8}$ for robust chip authentication using oscillator collapse in 40nm CMOS. In *ISSCC*, pages 1–3. IEEE, 2015.

[120] Stephanie Yang. What's worse than a chip shortage? Buying fake ones. https://www.wsj.com/articles/chip-shortage-has-spawned-a-surplus-of-fraudsters-and-fake-parts-11626255002, Jul 2021. Accessed on July 2022.

[121] XM Zeng, Q Liu, JY Tay, KY Chew, J Cheah, and CL Gan. High resolution front-side visualization of charge stored in EEPROM with scanning nonlinear dielectric microscopy (SNDM). *Nanotechnology*, 32(48), 2021.

[122] J Zhang, C Xu, M-K Law, Y Jiang, X Zhao, P-I Mak, and RP Martins. A 4T/cell amplifier-chain-based XOR PUF with strong machine learning attack resilience. *TCAS*, 2021.

[123] R Zhang, X Wang, K Liu, and H Shinohara. A 0.186-pJ per bit latch-based true random number generator featuring mismatch compensation and random noise enhancement. *JSSC*, 2022.

[124] H Zhuang, X Xi, N Sun, and M Orshansky. A strong subthreshold current array PUF resilient to machine learning attacks. *TCAS I*, 67(1):135–144, 2019.